

Figure 5: A π -Pref net (1) and a CP-net (2) modeling of the same preference specification

first step considers the best instantiation for all the dependent variables, and, in a second step, completes the rest in all possible ways.

In [10, 15], attempts at representing a CP-net ordering using a possibilistic logic framework are reported. However, authors indicate that it may not be possible to build an exact logical representation due some paradoxical behavior of CP-nets. Besides, they show that Symmetric Pareto and Leximin orderings respectively lower and upper bound the CP-net ordering. These results can be exploited with a π -Pref-net since it is the graphical counterpart of a symbolic possibilistic logic base [3].

6.2 π -Pref nets vs. OCF-nets

Ordinal Conditional Functions (OCF) [26] are an uncertainty representation framework very close to possibility theory [14]. OCF-nets may also offer a semi-quantitative graphical model for preference modeling [16]. OCF-nets obey Markov property as Bayesian networks (and possibilistic networks). Indeed, they have the same structure and carry the same conditional independence namely, each node is independent from its descendant in the context of its parents. This strong resemblance raises the question of a possible transformation between OCF-nets and π -Pref nets.

Formally, an OCF-net κG has two components: (i) a graphical component, a directed graph $\mathcal{G} = (V, E)$ where V denotes the set of nodes and E denotes the set of edges representing the preferential dependencies; (ii) a quantitative component: each variable $A_i \in V$ is associated to a normalized⁵ conditional rank, a non-negative integer $\kappa(A_i|u_i)$, where u_i is an instantiation of the parents $Pa(A_i)$ of A_i .

The OCF relative to a solution ω , denoted by $\kappa(\omega)$ is the sum of the elementary ranks of the conditional rank tables such that:

$$\kappa(\omega) = \sum_{i=1}^N \kappa(A_i|u_i) \quad (3)$$

This expression parallels the product-based possibilistic chain rule, where weights are combined by the product operator. The best solution in an OCF-net has a cost equal to 0, while in the possibilistic framework it has a possibility degree equal to 1. However, this preference network with a rank interpretation has a close relationship with product-based π -Pref nets. Indeed, the cost of a solution induced by an OCF-net corresponds actually to a transformation of the possibility degree computed from a π -Pref net.

In [12], it was pointed out that the set-function $\pi_\kappa(A_i) = 2^{-\kappa(A_i)}$ is a possibility measure. The converse holds to some extent insofar as if $\pi(A_i) = \alpha$, the values $\kappa(A_i) = -\log_2(\alpha)$ are integer rank

weights. However, we can also extend the OCF framework to positive reals. Up to this proviso, the ordering induced by the product-based chain rule of π -pref nets is the same as the order induced by the corresponding rank function. In [3], it was proposed to use this transformation at the symbolic level, yielding a symbolic additive counterpart to π -pref nets.

Clearly, $\pi_{\Pi G}(\omega) = \alpha_1 \cdot \dots \cdot \alpha_N \Rightarrow \kappa_{\kappa G}(\omega) = -(\log_2(\alpha_1) + \dots + \log_2(\alpha_N))$ and $\kappa_{\kappa G}(\omega) = (\alpha_1 + \dots + \alpha_N) \Rightarrow \pi_{\Pi G}(\omega) = 2^{-\alpha_1} \cdot \dots \cdot 2^{-\alpha_N}$. Thus, after the logarithmic transformation, OCF-nets yield the same ordering on configurations as π -Pref nets. Note that π -pref nets with products cannot always be turned into OCF-nets with integer values. However, OCF-nets can be turned into π -pref nets with products.

Example 8 Let us consider the following conditional rank tables corresponding to an OCF-net of two binary variables A and B : $\kappa(a) = 3$, $\kappa(-a) = 0$, $\kappa(b|a) = 0$, $\kappa(b|-a) = 2$, $\kappa(-b|a) = 1$ and $\kappa(-b|-a) = 0$. This yields $\kappa(-a-b) = 0 < \kappa(-ab) = 2 < \kappa(ab) = 3 < \kappa(a-b) = 4$. Thus we have $-a-b \succ_{\kappa G} -ab \succ_{\kappa G} ab \succ_{\kappa G} a-b$. The transformation from this OCF-net to a numeric π -Pref net leads to the following possibilistic conditional tables: $\pi(a) = 0.125$, $\pi(-a) = 1$, $\pi(b|a) = 1$, $\pi(b|-a) = 0.25$, $\pi(-b|a) = 0.5$ and $\pi(-b|-a) = 1$ which yields $\pi(-a-b) = 1 > \pi(-ab) = 0.25 > \pi(ab) = 0.125 > \pi(a-b) = 0.0625$. Clearly the two models lead to the same ordering after this transformation.

Until now, OCF-nets have been used for dealing with numerical values only. However, the transformation of a *symbolic* possibilistic network leads to a *symbolic* OCF-net. Thus, the application of the different ordering relations defined above lead exactly to the same orderings induced by possibilistic networks. In fact, summation and product are handled similarly when we work in a symbolic setting.

Recently, numerical OCF-nets have been shown to “mimic” the CP-net ordering [16]. The proposed generation of an OCF-net from a CP-net leads to a total ordering, which contrasts with CP-nets. However, they proved that such an ordering is always consistent with the one induced by the corresponding CP-net. They also showed that the CP-net formalism is able to represent only a subclass of OCF-nets, which proves that OCF-nets are more expressive than CP-nets. These remarks can be immediately applied to *numerical* π -Pref nets as well.

7 Conclusion

This paper proposes a detailed study of π -Pref nets, which, if based on the product chain rule, are closer to Bayesian nets than CP-nets. This model proves to be flexible enough to support different readings leading to different orderings of solutions, and establishes the main relationships between them. π -Pref nets correctly reflect the elicited information in the sense that no further implicit priority is enforced like with CP-nets (e.g., in favor of parent nodes). π -Pref nets also offer a cautious way of modeling preferences without requiring numerical values, which should make them attractive for the same class of applications as CP-nets. In fact, precise numerical assessments are hard to get for conditional preferences that are qualitative in nature. Moreover, we have shown that symbolic possibilistic networks can handle additional qualitative information when available. Beside the fact that π -Pref nets can be put under an equivalent possibilistic logic form suitable for inference, they have another additive graphical counterpart under the form of OCF-nets.

⁵ $\forall u_i \in Pa(A_i), \exists j$ such that $\kappa(a_j|u_i) = 0$

REFERENCES

- [1] N. Ben Amor, S. Benferhat, and K. Mellouli, 'Anytime propagation algorithm for min-based possibilistic graphs', *Soft Computing*, **8**(2), 150–161, (2003).
- [2] N. Ben Amor, D. Dubois, H. Gouider, and H. Prade, 'Possibilistic networks: A new setting for modeling preferences', in *8th Int. Conf. SUM*, pp. 1–7. Springer, (2014).
- [3] N. Ben Amor, D. Dubois, H. Gouider, and H. Prade, 'Possibilistic conditional preference networks', in *Proc.ECSQARU'2015*, pp. 36–46, (2015).
- [4] S. Benferhat, D. Dubois, L. Garcia, and H. Prade, 'On the transformation between possibilistic logic bases and possibilistic causal networks', *Int. J. of Approximate Reasoning*, **29**(2), 135–173, (2002).
- [5] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole, 'Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements', *J. Artif. Intell. Res.(JAIR)*, **21**, 135–191, (2004).
- [6] G. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole, 'Preference-based constrained optimization with CP-nets', *Computational Intelligence*, **20**(2), 137–157, (2004).
- [7] C. Cayrol, D. Dubois, and F. Touazi, 'Ordres Partiels entre Sous-Ensembles d'un Ensemble Partiellement Ordonné', Research report RR–2014-02–FR, IRIT, Université Paul Sabatier, Toulouse, (february 2014).
- [8] L. De Raedt, K. Kersting, S. Natarajan, and D. Poole, *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2016.
- [9] D. Dubois, H. Fargier, and H. Prade, 'Refinements of the maximin approach to decision-making in fuzzy environment', *Fuzzy Sets and Systems*, **81**, 103–122, (1996).
- [10] D. Dubois, S. Kaci, and H. Prade, 'Approximation of conditional preferences networks "CP-nets" in possibilistic logic', in *Proc. 15th Int. Conf. on Fuzzy Systems (FUZZ-IEEE), Vancouver, July 16-21*, (2006).
- [11] D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, Plenum Press, 1988.
- [12] D. Dubois and H. Prade, 'Epistemic entrenchment and possibilistic logic', *Artif. Intell.*, **50**(2), 223–239, (1991).
- [13] D. Dubois and H. Prade, 'Possibilistic logic: a retrospective and prospective view', *Fuzzy Sets and Systems*, **144**(1), 3–23, (2004).
- [14] D. Dubois and H. Prade, 'Qualitative and semi-quantitative modeling of uncertain knowledge—a discussion', in *Computational Models of Rationality Essays Dedicated to Gabriele Kern-Isberner on the occasion of Her 60th Birthday*, pp. 280–292. College Publications, (2016).
- [15] D. Dubois, H. Prade, and F. Touazi, 'Conditional Preference-nets, possibilistic logic, and the transitivity of priorities', in *Proc. 33rd SGAI Int. Conf., Cambridge*, pp. 175–184. Springer, (2013).
- [16] C. Eichhorn, M. Fey, and G. Kern-Isberner, 'CP-and OCF-networks—a comparison', *Fuzzy sets and Systems*, **298**, 109–127, (2016).
- [17] C. Eichhorn and G. Kern-Isberner, 'Using inductive reasoning for completing OCF-networks', *J. of Applied Logic*, **13**(4), 605–627, (2015).
- [18] M. Goldszmidt and J. Pearl, 'Qualitative probabilities for default reasoning, belief revision, and causal modeling', *Artificial Intelligence*, **84**(1), 57–112, (1996).
- [19] C. Gonzales and P. Perny, 'GAI networks for utility elicitation', in *Proc. 9th int. conf. Principles of Knowledge Representation and Reasoning*, pp. 224–234, (2004).
- [20] C. Gonzales and P. Perny, 'GAI networks for decision making under certainty', in *IJCAI05—Workshop on Advances in Preference Handling*. Citeseer, (2005).
- [21] C. Gonzales, P. Perny, and S. Queiroz, 'Réseaux GAI pour la prise de décision', *Revue d'intelligence artificielle*, **21**(4), 555–587, (2007).
- [22] G. Kern-Isberner and C. Eichhorn, 'OCF-networks with missing values', in *Proceedings of the 4th Workshop on Dynamics of Knowledge and Belief*, pp. 46–60, (2013).
- [23] F. Koriche and B. Zanuttini, 'Learning conditional preference networks', *Artificial Intelligence*, **174**(11), 685–703, (2010).
- [24] J. Lang and J. Mengin, 'The complexity of learning separable ceteris paribus preferences.', in *IJCAI*, pp. 848–853, (2009).
- [25] J. Pearl, 'Bayesian networks: A model of self-activated memory for evidential reasoning', in *Proc. of Cognitive Science Society (CSS-7)*, (1985).
- [26] W. Spohn, 'Ordinal conditional functions: a dynamic theory of epistemic states', in *Causation in Decision, Belief Change, and Statistics*, eds., W. L. Harper and B. Skyrms, volume 2, 105–134, D. Reidel, (1988).
- [27] L. A. Zadeh, 'Fuzzy sets as a basis for a theory of possibility', *Fuzzy Sets and Systems*, **1**, 3–28, (1978).

Leveraging Stratification in Twitter Sampling

Vikas Joshi¹ Deepak P² L V Subramaniam¹

¹IBM Research - India ²Queen's University Belfast, UK
vijoshij@in.ibm.com deepaksp@acm.org lvsubram@in.ibm.com

Abstract. With Tweet volumes reaching 500 million a day, sampling is inevitable for any application using Twitter data. Realizing this, data providers such as Twitter, Gnip and Boardreader license sampled data streams priced in accordance with the sample size. Big Data applications working with sampled data would be interested in working with a large enough sample that is representative of the universal dataset. Previous work focusing on the representativeness issue has considered ensuring that global occurrence rates of key terms, be reliably estimated from the sample. Present technology allows sample size estimation in accordance with probabilistic bounds on occurrence rates for the case of uniform random sampling. In this paper, we consider the problem of further improving sample size estimates by leveraging stratification in Twitter data. We analyze our estimates through an extensive study using simulations and real-world data, establishing the superiority of our method over uniform random sampling. Our work provides the technical know-how for data providers to expand their portfolio to include stratified sampled datasets, whereas applications are benefited by being able to monitor more topics/events at the same data and computing cost.

1 Introduction

Microblogging sites have seen massive penetration over the last many years. The importance of microblogging as a social signal is immense in this age when Twitter has been shown to be useful in the context of natural disasters[17] and political uprisings[10]. The usefulness of the data has led to new business models to monetize social media data. Data providers like Twitter, Gnip and Boardreader provide access to data through different application programming interfaces (APIs). They constantly innovate with pricing models to sell data. With the number of tweets generated daily measuring as much as 500 million¹, massive computation infrastructure is needed to analyze such big-data. In order to expand the customer base to include small-scale Twitter intelligence applications who have limited compute infrastructure and capital, data providers offer (uniformly) *sampled* data streams. For example: Twitter provides three popular sampled APIs namely: *Powertrack API*, which returns all the data for the given keywords at a higher base cost, *Decahose API*, returns 10% of entire data (uniformly and randomly sampled) at lower cost than Powertrack API, *Free 1% API* which is 1% of the entire data stream and is free of cost. In the quest to enrich the sampling portfolio without compromising on probabilistic guarantees, we study the use of stratified sampling to improve sample size estimates. Leveraging stratification can improve the quality of the sample by providing one of (a) tighter bounds than uniform random sampling for the same

sample size, or (b) smaller sample sizes than uniform random sampling conforming to the same probabilistic bounds. We now look at usage of uniformly sampled streams in big data applications, and introduce the task of stratified sampling for Twitter.

1.1 Using Uniformly Sampled Data

For a big data application, it is of interest to ensure that the sampled data used is representative of the global data, given the topic of interest. Probabilistic bounding of large deviations from global mean values [5] has been a popular way to ensure the same. The intuition is that ensuring reliable estimation of global occurrence rates would help in reliable estimation of the global results for end applications too. It is desirable to obtain a sampled set, such that the end result of any application (such as finding trending hashtags, sentiment analysis, topic clustering, summarization, etc) be close to the end result of the same application applied on the universe. However, given the complexity and variety of analytics tasks, such bounds are application-specific and need to be analytically developed separately for each application. As an example, an attempt to bound the results of a simple sentiment analyzer was done in [16]. In the interest of providing generic bounds that are likely to benefit any application, bounding occurrence rates of key entities such as words/hashtags [4, 16] has been explored as a natural first step in probabilistically guaranteed sampling.

Results from deviation theory [5] suggest that sample sizes to ensure probabilistic bounds need to be increased as the rate of presence of the monitored topic decreases in the global dataset. For example, a Twitter application monitoring national elections in India can afford to sample much fewer than another application focusing on a regional film festival, to achieve the same probabilistic bounds on deviation. This is because the former topic has a higher rate of presence (e.g., the hashtags occur more frequently) as compared to the latter topic that generates interest within a smaller audience. This has obvious cost implications; the application can switch from the paid Decahose API to the Free 1% stream while shifting focus from a niche topic to a much more popular one. Usage of such results requires that the rate of occurrence (of hashtags of interest to) the event monitored is known before-hand, to determine the sample size. For practical scenarios, the occurrence rates are available with data providers who already maintain indexes on data to support search functionalities. In short, an application using Twitter data for day-to-day monitoring of a topic would first characterize the topic of interest by keyword/hashtags and then query the data provider for a sample with a specification of the desired probability (e.g., > 90%) and permissible deviation (e.g., < 10%). The data provider would internally use the occurrence rate statistics of the hashtags, and estimate the required

¹ <http://uk.businessinsider.com/twitter-tweets-per-day-appears-to-have-stalled-2015-6?r=US&IR=T>

uniform random sample size with a corresponding pricing. This may be done using Chernoff bounds formulae [5] that provide the minimum sample size required to ensure that the occurrence rate in the sample, for *each* hashtag, does not deviate by more than the specified deviation with the specified probability. A data provider offering such a probabilistic bounded sampling API is obviously attractive to the users since it allows them to be frugal on sample sizes especially while monitoring popular topics.

1.2 Why Stratified Sampling?

We now motivate using an example as to why stratified sampling would be of interest in this scenario. Consider an intelligence application looking to assess global opinion polarity on the *US Presidential Elections*. Given the geographic focus of the topic, it is conceivable that core hashtags for this topic are twice as frequent in tweets from *US* when compared to the rest of the world (*RoW*), even if the overall tweet volumes from the *US* and *RoW* are comparable. Geographic stratification is already performed by data providers for tasks such as geo-specific trends estimation, and is thus readily available with them. Uniform sampling would require us to sample as many tweets from *RoW* as from *US*. Due to the low occurrence rates of pertinent hashtags in *RoW*, marginal utility of a tweet from *RoW* in determining opinion on the event would be lower than that from *US*. However, since the task is to gauge global opinion, we cannot readily use results from a pure *US* sample; in particular, analogous to the uniform random sampling case, we would like to ensure that the sample would enable us to estimate global occurrence rates accurately. There exists an opportunity to exploit the knowledge of differential occurrence rates across *US* and *RoW* to work with smaller samples without compromising on the desired probabilistic guarantees.

However, to the best of our knowledge, there exists no technology to exploit differential occurrence rates across strata to derive smaller sample sizes that agree to probabilistic bounds (as given by Chernoff bounds [5] for uniform random sampling) within or outside the context of Twitter sampling. Our focus in this paper is to precisely develop that technology. Data providers would be able to leverage our method to provide a newer set of sampling APIs, stratified sampling APIs, that will output stratified samples agreeing to the same probabilistic bounds as in the uniform random sample case. The data consumer provides the same input to the data provider, a set of hashtags and the desired specification of probabilistic bounds; the data provider would then use our formulation and provide a smaller stratified sample to the user. The smaller sample sizes help the data user to monitor more topics for the same data cost.

1.3 Our Contributions

Our main contributions are as follows:

- For the first time, we consider the problem of bounding deviations in occurrence rate estimates of words/hashtags in stratified sampling in the context of Twitter, and outline methods to estimate sufficient sample sizes.
- We analyze the quantum of gains achieved using our method over corresponding estimates from uniform random sampling under the same setting, on simulations as well as real-world data.

It may be noted that even small reductions in sufficient sample size estimates are critical since procurement of tweets is practically the costliest aspect of maintaining a Twitter-based intelligence application. Data providers can leverage our technology to diversify

their API portfolio to include stratified sampling. On the other hand, the improved sample sizes enable big-data analytics applications to broaden their footprint at the same data procurement and compute costs. Thus, our work is squarely targeted at players in the big data space. ■

Roadmap: We start off with some background on probabilistic guarantees and occurrence rate bounding in Section 2. We will outline related work in Section 3, define the problem in Section 4 and describe our method in Section 5. This is followed by extensive simulation and experimental analysis in Section 6 and conclusions in Section 7.

2 Background

2.1 Probabilistic Guarantees

The data user/application would like the data provider to provide guarantees on the sampled set in representing the universe. A probabilistic guarantee on occurrence rate ensures that the global occurrence rate as estimated from the sample does not deviate from the actual global occurrence rate more than a specified tolerance, with at least a specified probability. Thus, such a guarantee is fully specified by a combination of tolerance, and probability threshold. Consider an example application seeking to summarize Twitterati's opinion on the US Presidential Election. If the hashtag *#HillaryClinton* appears in 20% of the tweets in the whole Twitter stream, the application designer might like to ensure that the frequency of the hashtag as estimated from the sample be within $20 \pm 2\%$ (i.e., 10% tolerance), with a high probability (say, 90%). This is so since the hashtag *#HillaryClinton* is central to the problem that the application is trying to address. If the estimated frequency of *#HillaryClinton* from the sample turns out to be 30%, it could mean that our application's opinion summary is skewed in favor of users who mention *#HillaryClinton* (and vice versa). As in previous work, we will work with relative bounds expressed as percentages. For each application domain, one may intuitively expect that there would be some such core hashtags, noun phrases, or words of interest whose frequencies as estimated from the sample be close to the dataset frequency. A probabilistic guarantee on the occurrence rate for a set of hashtags/words specifies that the occurrence rate of each word in the set be estimated to within the specified tolerance subject to the probabilistic bound. For a particular sample, the occurrence rate condition is said to fail even if the occurrence rate of one word deviates further than the tolerance bound.

2.2 Occurrence Rate Bounding

Chernoff bounds [5] are tailored to address the probabilistically guaranteed sample size estimation problem in Random Sampling; i.e., the task of bounding the probability of tail events, specifically that of large divergence of occurrence rate estimates (from the sample) from their values in the universe. While being generally applicable to get bounds of large deviations from the mean, they provide sufficient sample sizes to probabilistically bound the deviation of the occurrence rate or frequency of a word/hashtag. Consider a Binomial random variable X that is the sum of iid Bernoulli random variables, then, for any $0 < \epsilon < 1$, the following hold:

$$\mathbb{P}\{X < (1 - \epsilon)\mathbb{E}[X]\} \leq e^{-\frac{\epsilon^2 \mathbb{E}[X]}{2}}$$

$$\mathbb{P}\{X > (1 + \epsilon)\mathbb{E}[X]\} \leq e^{-\frac{\epsilon^2 \mathbb{E}[X]}{3}}$$

where $\mathbb{P}\{Y\}$ denotes the probability of event Y , and $\mathbb{E}[X]$ the expectation of the random variable X ; $\mathbb{E}[X] = s \times p$ where p is the success rate of the underlying Bernoulli random variable and s is the number of trials (i.e., the sample size in the sampling case). For the word occurrence rate scenario, the Bernoulli random variable (that X sums over) is one that has success probability equivalent to the occurrence rate of the word in the corpus. For example, if the word appears in 10% of the tweets in the corpus, the corresponding Bernoulli random variable would have $p = 0.1$. These bounds can be generalized to multiple words/hashtags and have been explored in AI for data-intensive applications such as mining; for example, previous work has addressed the task of preserving the status of objects as being θ -frequent (i.e., have a frequency more than θ) or not [4]. This has been adapted to the context of sampling in Twitter as well [16] with further extensions to derive bounds on preserving the dominant sentiment of a word. Thus, there has been recent interest in deriving sufficient sample size estimates towards preserving specific statistics in uniform random sampling within the data analytics community.

3 Related Work

There are a variety of applications for mining Twitter data, including ones for tweet summarization [11][18], topic analysis [3][20] and twitter sentiment analysis [13]. Since most such methods would need to analyze content and are thus computationally intensive, sampling would be an essential step for them to be applied to large scale twitter data.

There has been much empirical work on sampling such as analysis of sampled streams [15, 7, 12, 9, 1]. In [15] the authors compare Twitter sampling API's feed with the tweets obtained from Twitter Firehose API, which contains all the tweets. Authors empirically find that the analysis from the data obtained using Twitter's Streaming API (1% random sample) do not conform with Twitter's Firehose data (100% sample) for a set of end applications. In [7], the authors empirically compare sampling done with the help of human "experts" against random sampling. In [14], the authors analyze the bias in Twitter's API without using the costly Firehose data. In [12], the effects of using multiple streaming APIs is studied. The authors conclude that the Twitter's 1% Streaming API is rather biased than being random. All these studies empirically evaluate the quality of the Tweets for the existing Twitter APIs which mostly employ uniform random sampling. In contrast, we provide a theoretical treatment to determine the sample size needed to produce representative samples using stratified sampling. A recent work on Twitter sampling [16] looks at classifying words into frequent or infrequent using a threshold; it then builds upon ideas from work on frequent itemset mining [4] to bound the probability of words having a status in the sample different from their status in the whole. They also extend the bounds to derive necessary sample sizes for preserving the dominant sentiment of words in the sample. Our work, while related due to addressing sampling on Twitter, focuses on a different problem.

Stratified sampling has been studied extensively in the statistics literature [6, 8, 19]. The existing methods find the optimal sample size to minimize the variance of the estimates. However, they do not transcend into the probabilistic guarantees in bounding the estimates as done by Chernoff bounds [5]. We advance the state-of-art in stratified sampling, by deriving expressions to find the probability of bounding the estimates for the chosen sample size; these can in turn be used for arriving at sufficient sample sizes.

4 Problem Formulation

Consider a dataset \mathcal{D} of tweets that is stratified/split into two strata \mathcal{D}_1 and \mathcal{D}_2 ; we will consider two-strata stratification for narrative simplicity and will later show that the problem definition as well as our method easily generalizes to any number of strata. Now, consider a set of words/tags/phrases² of interest, $w = \{w^1, w^2, \dots\}$ for whom the occurrence rate is known in each stratum; \hat{x}_j^i denotes the rate of occurrence of w^i in \mathcal{D}_j whereas \hat{x}^i denotes the occurrence rate of w^i in the whole dataset \mathcal{D} . Occurrence rates measure the fraction of tweets from the stratum of interest, and are thus in $[0, 1]$. We also have a chosen tolerance level ϵ indicating the amount of fractional deviation from expected occurrence rate, and a probability h that bounds the probability of larger deviations.

The task of interest is to identify a stratified sampling strategy $[S_1, S_2]$ where S_1 and S_2 tweets be uniformly randomly sampled separately from \mathcal{D}_1 and \mathcal{D}_2 respectively, so that such a sample \mathcal{S} ($|\mathcal{S}| = S_1 + S_2$) confirms to the following:

$$\mathbb{P}\{\cup_i (X_{\mathcal{S}}^i \leq (1 - \epsilon)\hat{x}^i|\mathcal{D} \cup X_{\mathcal{S}}^i \geq (1 + \epsilon)\hat{x}^i|\mathcal{D})\} < h \quad (1)$$

where $X_{\mathcal{S}}^i$ is a random variable denoting the extrapolated frequency of w^i in \mathcal{D} from a sample \mathcal{S} generated using the stratified-sampling strategy $[S_1, S_2]$ and $\hat{x}^i \times |\mathcal{D}|$ denotes the actual frequency in the whole dataset. Preserving frequencies by a multiple of $\pm\epsilon$ is exactly the same as preserving occurrence rates by a multiple of $\pm\epsilon$, since occurrence rates is simply the frequency scaled down by the dataset size (on both sides of the inequality). Informally, we want to ensure that for any sample generated according to the strategy $[S_1, S_2]$, the probability of the estimated frequency of *any* word w^i (i.e., $X_{\mathcal{S}}^i$) deviating by more than $\pm\epsilon$ times the actual frequency be bounded by h . In particular, even one word not satisfying its condition would be a failure event. This can be generalized to k strata by changing the format of the strategy of interest from a pair to a k -length array $[S_1, \dots, S_k]$.

5 Our Method

We will first outline our method for two-strata stratified sampling, and later show how that could be generalized to more number of strata. Consider a stratified sampling strategy $[S_1, S_2]$ and a word w^i . Let x_j^i be the random variable corresponding to finding the word w^i in stratum \mathcal{D}_j ($j \in \{1, 2\}$). $X_{\mathcal{S}}^i$ is then a function of x_j^i 's as the following:

$$X_{\mathcal{S}}^i = \frac{|\mathcal{D}_1|}{S_1} \times \sum_{k=1}^{S_1} x_1^i + \frac{|\mathcal{D}_2|}{S_2} \times \sum_{k=1}^{S_2} x_2^i \quad (2)$$

$X_{\mathcal{S}}^i$ is the conventional stratified sampling variable denoting frequency of w^i in \mathcal{D} under the stratified sampling strategy $[S_1, S_2]$. The expected value of $X_{\mathcal{S}}^i$, which we will denote as μ^i , is independent of \mathcal{S} , and may be written as:

$$\mathbb{E}[X^i] = \mu^i = |\mathcal{D}_1| \times \hat{x}_1^i + |\mathcal{D}_2| \times \hat{x}_2^i = |\mathcal{D}| \times \hat{x}^i \quad (3)$$

The last condition holds since the extrapolation in $X_{\mathcal{S}}^i$ is in proportion to the strata sizes. We use μ^i and the union bound on Eq. 1 to write as:

² referred to generically as words hereon.

$$\mathbb{P}\{\cup_i ((X_S^i \leq (1-\epsilon)\mu^i) \cup (X_S^i \geq (1+\epsilon)\mu^i))\} < \left(\sum_i \mathbb{P}\{X_S^i \leq (1-\epsilon)\mu^i\} \right) + \left(\sum_i \mathbb{P}\{X_S^i \geq (1+\epsilon)\mu^i\} \right)$$

We will consider bounding the RHS of the above equation, to be lower than h . Going by conventions, we will refer to the first term in the RHS as the left-tail, and the second term as the right-tail. We first illustrate simplifying the left-tail expression for a particular w^i .

5.1 Left-Tail

We will now use a positive quantity t and multiply each side of the internal expression by $-t$ and exponentiate, with a corresponding inversion of the inequality. It may be noted that this is inspired from the classical derivation for Chernoff bounds; however, unlike Chernoff bounds, our random variable is not a Binomial random variable.

$$\mathbb{P}\{X_S^i \leq (1-\epsilon)\mu^i\} = \mathbb{P}\{\exp(-t(1-\epsilon)\mu^i) \geq \exp(-tX_S^i)\}$$

Using the Markov inequality, i.e., $\mathbb{P}\{A \geq a\} \leq \frac{\mathbb{E}[A]}{a}$, the above expression is upper bounded by:

$$\frac{\mathbb{E}[\exp(-tX_S^i)]}{\exp(-t(1-\epsilon)\mu^i)} \quad (4)$$

Let us now focus on the numerator, which we expand using the expression from Eq. 2 and re-write using $\exp(a+b) = \exp(a) \times \exp(b)$.

$$\begin{aligned} \mathbb{E}[\exp(-tX_S^i)] &= \\ \mathbb{E}[\exp(-t \times (\frac{|\mathcal{D}|}{S_1} \times \sum_{k=1}^{S_1} x_1^i)) \times \exp(-t \times (\frac{|\mathcal{D}|}{S_2} \times \sum_{k=1}^{S_2} x_2^i))] & \end{aligned} \quad (5)$$

x_1^i and x_2^i within the summation in the equation above are random variables. We can take the $\mathbb{E}[\cdot]$ and $\exp(\cdot)$ inward, assuming independence among the inner random variables.

$$= \left(\prod_{k=1}^{S_1} \mathbb{E}[\exp(-tx_1^i \frac{|\mathcal{D}_1|}{S_1})] \right) \left(\prod_{k=1}^{S_2} \mathbb{E}[\exp(-tx_2^i \frac{|\mathcal{D}_2|}{S_2})] \right) \quad (6)$$

Consider the internal expression $\mathbb{E}[\exp(-tx_j^i \frac{|\mathcal{D}_j|}{S_j})]$ (sub/superscripts generalized). The random variable x_j^i will be 1 with a probability of \hat{x}_j^i and 0 with a probability $(1 - \hat{x}_j^i)$. We can write the expectation as the sum of these two cases:

$$\begin{aligned} \mathbb{E}[\exp(-tx_j^i \frac{|\mathcal{D}_j|}{S_j})] &= \hat{x}_j^i \times \exp(-t \frac{|\mathcal{D}_j|}{S_j}) + (1 - \hat{x}_j^i) \times \exp(0) \\ &= 1 - \hat{x}_j^i \left(1 - \exp(-t \frac{|\mathcal{D}_j|}{S_j}) \right) \end{aligned}$$

We now use the inequality $1 - x < \exp(-x)$ to further upper bound the above expression as:

$$\mathbb{E}[\exp(-tx_j^i \frac{|\mathcal{D}_j|}{S_j})] < \exp\left(-\hat{x}_j^i \left(1 - \exp(-t \frac{|\mathcal{D}_j|}{S_j})\right)\right) \quad (7)$$

Re-writing and putting this back into Eq. 6,

$$\begin{aligned} \mathbb{E}[\exp(-tX_S^i)] &< \left(\prod_{k=1}^{S_1} \exp(\hat{x}_1^i (\exp(-t \frac{|\mathcal{D}_1|}{S_1}) - 1)) \right) \\ &\times \left(\prod_{k=1}^{S_2} \exp(\hat{x}_2^i (\exp(-t \frac{|\mathcal{D}_2|}{S_2}) - 1)) \right) \end{aligned} \quad (8)$$

Since $\exp(a) \times \exp(b) = \exp(a+b)$:

$$\begin{aligned} < \exp\left(\sum_{k=1}^{S_1} (\hat{x}_1^i (\exp(-t \frac{|\mathcal{D}_1|}{S_1}) - 1))\right) + \\ \sum_{k=1}^{S_2} (\hat{x}_2^i (\exp(-t \frac{|\mathcal{D}_2|}{S_2}) - 1)) \end{aligned} \quad (9)$$

Since the expression does not have random variables:

$$< \exp\left(\sum_{j \in \{1,2\}} S_j \hat{x}_j^i (\exp(-t \frac{|\mathcal{D}_j|}{S_j}) - 1)\right)$$

Replacing this upper bound in Eq. 4 and re-writing μ^i ,

$$\begin{aligned} \mathbb{P}\{X_S^i \leq (1-\epsilon)\mu^i\} &< \exp\left(t(1-\epsilon)(|\mathcal{D}| \times \hat{x}^i) \right. \\ &\left. + \sum_{j \in \{1,2\}} S_j \hat{x}_j^i (\exp(-t \frac{|\mathcal{D}_j|}{S_j}) - 1)\right) \end{aligned} \quad (10)$$

Using a similar sequence of steps for right-tail:

$$\begin{aligned} \mathbb{P}\{X_S^i \geq (1+\epsilon)\mu^i\} &< \exp\left(-t(1+\epsilon)(|\mathcal{D}| \times \hat{x}^i) \right. \\ &\left. + \sum_{j \in \{1,2\}} S_j \hat{x}_j^i (\exp(t \frac{|\mathcal{D}_j|}{S_j}) - 1)\right) \end{aligned} \quad (11)$$

We will refer to the expressions in the RHS of Eq. 10 and Eq. 11 as $LU(t, i, S_1, S_2, \epsilon)$ and $RU(t, i, S_1, S_2, \epsilon)$ respectively³. These upper bounds hold for any positive value of t ; the preferred value of t would be that which gives the tightest bound. Further, the expressions above can be easily generalized to any stratification of the dataset into k strata by letting the j variable iterate over as many values as there are strata.

5.2 Full Expression and Optimization

The full expression for upper bound would thus be:

$$\begin{aligned} \mathbb{P}\{\cup_i ((X_S^i \leq (1-\epsilon)\mu^i) \cup (X_S^i \geq (1+\epsilon)\mu^i))\} &< \\ \sum_i \left(LU(t_L^i, i, S_1, S_2, \epsilon) + RU(t_R^i, i, S_1, S_2, \epsilon) \right) \end{aligned} \quad (12)$$

If the RHS of the above expression evaluates to less than h , then the LHS would too (since LHS < RHS as above), and our task in Eq. 1 will be satisfied. We have added subscripts and superscripts to t within the expressions to indicate that the t s need not necessarily

³ Short for Left-tail Upper bound and Right-tail Upper bound

take the same value across expressions and are internal to the expression; the t used in the $LU(\cdot)$ for one word w_i could be different from that used in the $LU(\cdot)$ or $RU(\cdot)$ for the same or different words. It may be noted that the flexibility that we have is to alter S_1 , S_2 and the t 's (ϵ is part of the problem specification), since the data stratification is given and \hat{x}_j^i is deterministic in the sense that it is calculated from the stratified dataset. To re-iterate, if we can find values of S_1 , S_2 and the t s such that the following holds

$$\sum_i \left(LU(t_L^i, i, S_1, S_2, \epsilon) + RU(t_R^i, i, S_1, S_2, \epsilon) \right) < h \quad (13)$$

we can then claim to have a sampling strategy $[S_1, S_2]$ that addresses our task. However, simply addressing the task is not sufficient; for example, a sample size for uniform random sampling that addresses our task is easily available from Chernoff bounds. Our interest is in achieving the task using fewer samples than uniform random sampling by leveraging strata level occurrence rates (i.e., \hat{x}_j^i s), and thus the measure of interest is the total sample size, $S_1 + S_2$, which we will look to minimize. Thus, ideally, we look for values of S_1 , S_2 and the t s such that the above condition is satisfied and $S_1 + S_2$ is minimized.

Due to the complexity of the expression, a search in the possible values of S_1 , S_2 and the t s is a possibility to identify feasible sampling strategies. From an optimization perspective, it is useful to localize the search to a small region of the parameter space, in the interest of reducing computational expense. Since S_1 and S_2 are sizes of samples from \mathcal{D}_1 and \mathcal{D}_2 respectively, their ranges would respectively be $[1, |\mathcal{D}_1|]$ and $[1, |\mathcal{D}_2|]$. Though the extent of the search space for values of S_1 and S_2 are finite (due to bounds), the t s can take any positive value; we will now see how to localize the optimal t to limit the search.

5.3 Localizing the Optimal t

Consider the RHS in Eq. 10 which we are interested in minimizing (Ref. Eq. 13); we will focus on optimizing for t for chosen values of S_1 and S_2 . Since $\exp(x)$ increases with x , we can focus on minimizing the expression within the $\exp(\cdot)$:

$$fLU_i(t) = t(1 - \epsilon)(|\mathcal{D}| \times \hat{x}^i) + \sum_{j \in \{1,2\}} S_j \hat{x}_j^i (\exp(-t \frac{|\mathcal{D}_j|}{S_j}) - 1) \quad (14)$$

We outline some analytical observations about the behavior of $fLU_i(t)$ with varying t ; we omit detailed derivations for brevity. First, $fLU_i(t = 0) = 0$. This is evident from setting $t = 0$ in Equation 14. Secondly, there exists a positive value t' such that the following hold:

$$\frac{\partial fLU_i(0 < t < t')}{\partial t} < 0$$

$$\frac{\partial fLU_i(t = t')}{\partial t} = 0$$

$$\frac{\partial fLU_i(t > t')}{\partial t} > 0$$

In other words, $fLU_i(t)$ is a convex function in t in our region of interest (i.e., positive t or $t \in (0, \infty]$) with an optima at t' where $fLU_i(t')$ would evaluate to a negative value. Thus, if we can find

values t_l and t_u such that $\frac{\partial fLU_i(t=t_l)}{\partial t} < 0$ and $\frac{\partial fLU_i(t=t_u)}{\partial t} > 0$, we can localize the search for the optimal t to the range (t_l, t_u) since the optimal t is bound to be in that range, given the above observations.

We will show that $\left[\frac{\log(\frac{1}{1-\epsilon})}{\max\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}}, \frac{\log(\frac{1}{1-\epsilon})}{\min\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}} \right]$ is one such range.

Consider the slope of $fLU_i(t)$:

$$\frac{\partial fLU_i(t)}{\partial t} = (1 - \epsilon)(|\mathcal{D}| \times \hat{x}^i) + \sum_{j \in \{1,2\}} |\mathcal{D}_j| \hat{x}_j^i \exp(-t \frac{|\mathcal{D}_j|}{S_j})$$

Setting $t = \log(\frac{1}{1-\epsilon}) / \max\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}$ in the above expression and using Eq. 3 with some re-arrangements yields:

$$\sum_{j \in \{1,2\}} |\mathcal{D}_j| \hat{x}_j^i \left((1 - \epsilon) - (1 - \epsilon)^{\frac{|\mathcal{D}_j|}{S_j} \max\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}} \right)$$

The exponent of the second $(1 - \epsilon)$ is evidently less than 1.0 since its denominator is least as big as its numerator (if not bigger). Also, given that $(1 - \epsilon) < 1.0$ and due to the obvious result that $x^y > x$ when $x < 1.0$ and $y < 1.0$, the multiplier of each $|\mathcal{D}_j| \hat{x}_j^i$ term would be negative, leading to a negative value for the whole expression. Analogously, we now consider the slope at $t = \log(\frac{1}{1-\epsilon}) / \min\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}$:

$$\sum_{j \in \{1,2\}} |\mathcal{D}_j| \hat{x}_j^i \left((1 - \epsilon) - (1 - \epsilon)^{\frac{|\mathcal{D}_j|}{S_j} \min\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}} \right)$$

In this case, the exponent of the second $(1 - \epsilon)$ turns out to be greater than one. Thus, adapting the earlier argument, the multiplier of each $|\mathcal{D}_j| \hat{x}_j^i$ would be positive, leading to an overall positive value. Thus:

$$\arg \min_t fLU_i(t) \in \left[\frac{\log(\frac{1}{1-\epsilon})}{\max\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}}, \frac{\log(\frac{1}{1-\epsilon})}{\min\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}} \right]$$

The analogous result for the right-tail expression is:

$$\arg \min_t fRU_i(t) \in \left[\frac{\log(1 + \epsilon)}{\max\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}}, \frac{\log(1 + \epsilon)}{\min\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}} \right]$$

Though the optimal value of t would be different for expressions corresponding to different words, the bounds above are attractive in that they do not depend on any \hat{x}_j^i s and thus can be used across words. These bounds can be easily extended from two strata to multiple strata by changing the max and min to iterate over k entries instead of two.

It is computationally intensive to find a separate optimal value of t for each term in Eq. 13. Thus, one might fall back to search for a single value of t to be used across all expressions in Eq. 13; this single value could be chosen as that which minimizes the value of the whole expression in Eq. 13. For such a case, the search may be directed to within the union of the left-tail and right-tail bounds above, which would be:

$$\left[\frac{\log(1 + \epsilon)}{\max\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}}, \frac{\log(\frac{1}{1-\epsilon})}{\min\{\frac{|\mathcal{D}_1|}{S_1}, \frac{|\mathcal{D}_2|}{S_2}\}} \right] \quad (15)$$

Optimal value of t can be either obtained by searching through the values in the range given by Eq. 15, or by using a gradient descent approach where the update equation would be:

$$t^{new} = t^{old} - \eta \frac{\partial f LU_i(t)}{\partial t} \quad (16)$$

where η is the learning rate. t can be initialized to any value in the range given by Eq. 15. In our experiment, we have used grid-search approach to search for the optimal value of t in view of its simplicity for the optimization of a single dimensional variable.

5.4 Grid-Search

Algorithm 1 outlines an intuitive grid-search approach, *StratSam*, to discover a sampling strategy $[S_1, S_2]$; we resort to choosing a single value of t across terms in Eq. 13 for computational convenience as outlined earlier. The algorithm is largely self-explanatory with lines 6 and 7 checking for satisfaction of the task condition (Eq. 13). Line 4 avoids checking for strategies that are already worse on total sample size than the best strategy seen so far. It may be noted that *StratSam* allows for exploring the trade-off between computational expense and accuracy by tuning the step-size hyperparameters. Smaller step-sizes would allow to discover a better sampling strategy (i.e., smaller $(S_1 + S_2)$) whereas larger step-sizes lead to fast search completion. It is also worthy to point out that the condition may never be reached when the chosen h and ϵ values are very small for the dataset size; in such cases, we will choose the entire dataset as the sample. In large datasets such as those with Twitter, such cases are very rare.

Alg. 1 Grid-Search: *StratSam*

Input. 2-Strata Dataset Specs: $|\mathcal{D}_1|, |\mathcal{D}_2|, \forall w_i, (\hat{x}_1^i, \hat{x}_2^i)$ pairs

Problem Specs: ϵ, h

Hyper-Parameters. Step-sizes $s_1, s_2, \delta t$

Output. Sampling Strategy, i.e., a vector $[S_1, S_2]$.

1. Best Strategy, $BS = \phi$, Best Strategy Size, $BSS = \infty$
 2. For $S_1 = 1 \rightarrow |\mathcal{D}_1|$ in steps of s_1
 3. For $S_2 = 1 \rightarrow |\mathcal{D}_2|, s_2$
 4. If $(S_1 + S_2) > BSS$ continue;
 5. For $t = \frac{\log(1+\epsilon)}{\max\{\frac{|\mathcal{D}_1|}{s_1}, \frac{|\mathcal{D}_2|}{s_2}\}} \rightarrow \frac{\log(\frac{1}{1-\epsilon})}{\min\{\frac{|\mathcal{D}_1|}{s_1}, \frac{|\mathcal{D}_2|}{s_2}\}}, \delta t$
 6. Evaluate $v = \sum_i LU(i) + RU(i)$ with the choices of S_1, S_2 and t
 7. If $(v < h) \wedge (S_1 + S_2 < BSS)$
 8. $BS = [S_1, S_2], BSS = (S_1 + S_2)$
 9. Output BS as sampling strategy of choice.
-

5.5 Remarks

Better Sample Sizes: The total sample size from the above stratified approach would *always be equal or smaller* than that from a similar uniform random sampling approach (or that from the looser Chernoff bounds). This is so since the latter's sample size would also be a valid solution for the former, when split in proportion to strata sizes.

Speeding up the Search: Our proposed grid-search approach is quite feasible for a small number of strata and is very fast. It can be further speed-ed up by replacing the grid-search for t (Lines 5-8 in the algorithm) by a gradient descent approach, given the convexity observation from Section 5.3. In resource constrained scenarios or

to estimate sample sizes for fine-grained data stratification, conventional optimization methods may be employed over the entire search space of S_i s and t . For purposes of optimization, the objective function is simply $(\sum_j S_j)$ with the generalization of Eq. 13 to the required number of strata serving as an inequality constraint.

5.6 Uptake of Our Work

We now discuss considerations relating to uptake of our work. Analogous to the assumption of global occurrence rate availability for the uniform random sampling setting, we assume the availability of stratum-level occurrence rates. We will now outline why stratum-level occurrence rate availability is a feasible assumption in practical scenarios. Our target ecosystem is the emerging data economy that encompasses data providers and data consumers. Data providers maintain the entire dataset and provide various kinds of APIs for usage by data consumers with a pricing scheme. These APIs would include sampled streams, as well as search functionalities and various analytics features such as geo-trends, all of which require content indexing at the level of different granularities such as strata. The source of the data (e.g., the region), the type of the tweets (e.g., Twitter activity streams⁴) etc. provide straightforward stratifications that would be maintained at the data provider. Typical search functionalities are supported by inverted lists at the level of each word/tag; occurrence rates are then simply normalized inverted list sizes. Our method leverages the skew in occurrence rates of topical hashtags across strata to reduce required sample sizes as against those for uniform random sampling. Our results are generalizable to cover domains such as market-basket data mining where frequencies of specific items within transactions (as opposed to frequencies of words in tweets) are the measure of interest; in such cases, we can leverage existing stratification of customers such as *silver, gold* and *platinum* and/or stratification of stores such as *small* and *large* to collect stratum-level information. Uptake of our technology necessitates a few simple changes at the data user as well as the data provider.

Data User/Application: The sampled data request issued by the data user remains the same, i.e., a set of words and the specification of desired probabilistic bound. However, the data sample received from the provider would now be a stratified sample. Analogous to usage of uniform random samples where the results (e.g., sentiment frequencies) derived from the sample needs to be extrapolated to get to corpus-level estimates, results from the stratified samples need to be extrapolated in accordance with the sampling rates (as in Equation 2), to arrive at corpus-level estimates. This is the only difference required at the data user's side.

Data Provider: As outlined earlier, the data provider maintains multiple stratifications of Twitter data; while some of these may be maintained for purposes such as providing trends estimation and faceted search, some stratifications could be specifically targeted at supporting the new stratified sampling API. The data provider kicks off processing upon receiving a sampled data request from the user comprising of a set of words/tags, tolerance, and probability threshold. Next, the data provider runs our method against each stratification separately using respective occurrence rate statistics, each of which provide a different sample size estimate. The smallest sample size estimate is expected to be achieved for the stratification where the skew of occurrence rates for the provided set of words/tags is maximum. The data provider would then return the best sample, and charge the data user accordingly. In a competitive marketplace, it is in the inter-

⁴ <http://support.gnip.com/articles/activity-streams-intro.html>

est of the data provider to maintain a rich library of different stratifications. This would ensure that low sample sizes may be provided for data requests on a variety of topics, enhancing chances of repeat business.

6 Simulation and Experiments

We first describe the setup for our simulation and experimental studies followed by results and discussion.

6.1 Experimental Setup

We compare our method, *StratSam*, against uniform random sampling (US), the baseline method. Instead of using the final Chernoff bounds result that involves many approximations leading to looser (i.e., larger) sample size estimates, we do a similar derivation as in our case and use a grid search for fairness in comparison. For clarity, the US expression corresponding to RHS in Eq. 10 is:

$$\exp\left(t(1 - \epsilon)(|\mathcal{D}| \times \hat{x}^i) + S\hat{x}^i(\exp(-t\frac{|\mathcal{D}|}{S}) - 1)\right) \quad (17)$$

The comparison of interest would be that between the US sample size (*US.Size*) and the total sample size $S_1 + S_2$; we use the sample size ratio, $SSR = \frac{S_1+S_2}{US.Size}$, as the primary evaluation measure; $SSR \leq 1$ always holds (Sec. 5.5), and lower values of *SSR* are desirable. We perform extensive simulation analysis as well as experiments on real-world data to illustrate the savings achieved by *StratSam* over US. In the case of analysis on real-world datasets, we analyze another measure, the actual empirical failure rate (*StratSam* and US guarantee that to be bounded by h) as well. For the real-world dataset, we use a set of tweets crawled around the time of the Indian General Election, 2014⁵. In our *StratSam* implementation, we use 100 equal steps in each of the three parameters.

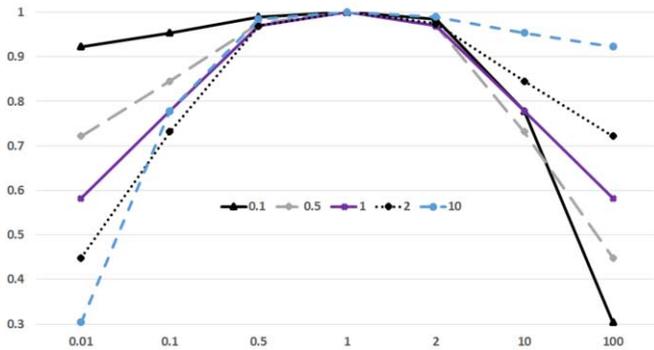


Figure 1. $SSR (\frac{S_1+S_2}{US.Size})$ on Y-axis vs. Occurrence Rate Ratio ($\frac{\hat{x}_1}{\hat{x}_2}$) plots for varying stratum size ratios ($\frac{|\mathcal{D}_1|}{|\mathcal{D}_2|}$)

6.2 Simulation Studies

We now use simulation studies to analyze the behavior of our approach. Two cases are considered: first, where there is only one core word for the topic of interest, and a second case involving two words.

6.2.1 Single Word Simulation

Figure 1 plots the *SSR* trends when the occurrence rate ratio of a word across the two strata (\hat{x}_1/\hat{x}_2) is varied keeping the dataset-level occurrence rate (i.e., \hat{x}^1) constant at 0.2. We use $\epsilon = 0.1$ and $h = 0.1$ for the plot in the figure; the trends were similar for other choices of ϵ and h too. Such trendlines are plotted for varying values of relative strata sizes ($\frac{|\mathcal{D}_1|}{|\mathcal{D}_2|}$) from 0.1 to 10. When each trendline is analyzed, it may be seen that *StratSam* is able to achieve smaller sample sizes as \hat{x}_1/\hat{x}_2 deviates from 1 on either side. When occurrence rates are equal, the strata are practically indistinguishable wrt w^1 and *StratSam* defaults to the US sample size, as is expected. It may be noted that *StratSam* is able to leverage the skew in occurrence rates under equal strata sizes to achieve > 40% reductions in sample sizes over *US*. On analyzing across trendlines (i.e., across stratum size ratios), it is evident that *StratSam* performs best when the occurrence rate is very high in a very small stratum; for example, the bottom-right point in the chart corresponds to the word being 100 times more frequent in the first stratum when it is $1/10^{th}$ of the second stratum in size. Thus, the chosen keywords being denser in the smaller stratum is favorable to *StratSam*.

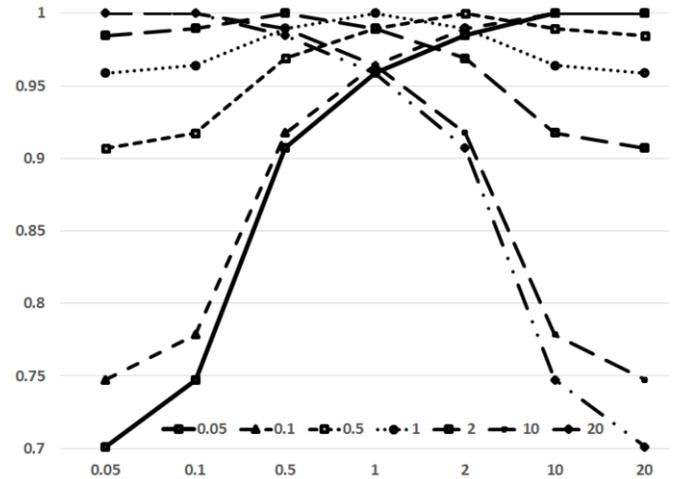


Figure 2. $SSR (\frac{S_1+S_2}{US.Size})$ on Y-axis vs. w^1 Occurrence Rate Ratio ($\frac{\hat{x}_1}{\hat{x}_2}$) plots for varying w^2 ratios ($\frac{\hat{x}_2}{\hat{x}_1}$)

6.2.2 Two Words Simulation

Figure 2 analyzes *SSR* trends for two words with equal sized strata (i.e., $|\mathcal{D}_1| = |\mathcal{D}_2|$). For the trendline where the second word is equally dense on either strata, deviations of \hat{x}_1/\hat{x}_2 shows similar trends as for the single-word case, though the quantum of savings is lower. Across trendlines, it may be seen that both the words being more skewed towards the same stratum (i.e., both occurrence rate ratios being low, or both being high) leads to maximum savings, with up to 30% savings recorded when occurrence rate ratios are both 0.05 (or equivalently, 20). Since typical sampling scenarios would be task focused (e.g., gauging sentiment on *US Elections*), it is intuitive to expect that words of interest are skewed towards the same stratum. The *SSR* trends were consistent with varying values of ϵ and h .

6.3 Experiments on Real-World Data

We use the tweet set from the Indian General Elections, 2014, and consider how *StratSam* performs on *SSR* under sets of words related

⁵ https://en.wikipedia.org/wiki/Indian_general_election,_2014

Words or Phrases	Universe size	Strata size	Strata Size Ratio	Word Skew (#N,#S)	Sample size (Strat-Sam)	Sample size (US)	SSR	Empirical Failure rate Strat-Sam	US
arvind, kejriwal, contribution	54501	north: 46808 south: 7693	6.08	(2,1)	14076 north:8993 south:5083	18638	0.75	0.01	0.008 (0.031)
bjp, modi, latestnew	87550	north: 31721 south: 55829	0.57	(2,1)	11997 north:1584 south:10413	16072	0.74	0.011	0.015 (0.028)
latestnew, sonia, varanasi, win, aap, kejriwal, firstpost, narendra, namo, exit, gandhi, arvind, bjp, vote, modi, poll,	265060	north: 98726 south: 166334	0.59	(11,6)	56199 north:15149 south:41050	58643	0.95	0.011	0.012 (0.018)

Table 1. Results on Real Data (North-South Stratification)

Words or Phrases	Universe size	Strata size	Strata Size Ratio	Word Skew (#E,#W)	Sample size (Strat-Sam)	Sample size (US)	SSR	Empirical Failure rate Strat-Sam	US
arvind, kejriwal, contribution	54501	east: 48959 west: 5542,	9.09	(2,1)	15641 west:3388 east:12252	18638	0.83	0.015	0.018 (0.028)
bjp, modi, latestnew	87550	east: 44589 west: 42961	1.03	(2,1)	10752 east:2603 west:8149	16072	0.66	0.01	0.014 (0.058)
latestnew, sonia, varanasi, win, aap, kejriwal, firstpost, narendra, namo, exit, gandhi, arvind, bjp, vote, modi, poll,	265060	east: 140378 west: 124682	1.12	(10,7)	44898 east:17379 west:27518	50691	0.88	0.008	0.011 (0.017)

Table 2. Results on Real Data (East-West Stratification)

to the election. We use the geo-stratification of tweets as *North* and *South*; *East* and *West* India using the location and time zone in the user profile. Twitter’s API was used to crawl tweets from May 12 to May 19, 2014, using topical keywords related to the election event. Table 1 and 2 summarize some representative results. Instead of using the entire set of tweets as the dataset, we wanted to experiment with varying dataset sizes too. Towards this, for every set of keywords, we filter out all tweets not containing even one of those keywords, to create the dataset for that keyword set. Thus, universe size represents the number of tweets obtained after such filtering. Strata size shows the number of tweets in the respective strata, with the strata size ratio indicating the ratio of the sizes of the strata. To provide a sense of the word skew, we look at each word in the set of interest, and assign it to the stratum in which it has a higher occurrence rate; thus, a word skew of (2, 1) indicates that 2 words have higher occurrence rates in the first stratum and the third word in the set occurs at a higher rate in the second stratum. While this does not capture the quantum of stratum-skew for each word, it is an indicator of the occurrence rate skew in the set of words of interest. We also report the sample size for *StratSam* and US, the SSR, and the empirical error rates obtained by repeatedly sampling (with 1000 Monte Carlo rounds) according to the respective strategy and measuring the fractional failure rate (which is analytically bounded above by $h = 0.1$). Results are obtained for $\epsilon = 0.1$ and $h = 0.1$. Empirical error rate within brackets in US column is obtained by uniform sampling with *StratSam* sample size. The trends are similar to that from the simulation and the experiments record gains up to 34% with significantly lower empirical error rates as well. The dataset was collected for the general elections, a pan-India event, thus mitigating the skew between various geographic strata within India; while this setting helps us observe that *StratSam* can achieve significant gains even in not-so-favorable scenarios, *StratSam* is expected to achieve much better gains when the stratification is more ‘aligned’ to the keyword set. It may be noted that in practical scenarios where millions of tweets

need to be sampled on a paid-basis, even $\approx 5\%$ gains are expected to result in large cost savings. Another noteworthy point is that most empirical failure rates are ≈ 10 times smaller than $h (= 0.1)$; this indicates potential for more empirical and/or theoretical work.

7 Conclusions and Future Work

In this paper, we considered the problem of using stratification in estimating sufficient sample sizes to reliably estimate the occurrence rates of specific words of interest, in sampling for Twitter. We exploit differential word occurrence rates across strata in a grid-search approach to significantly improve upon analogous estimates for uniform random sampling. We analyze our estimates through simulation studies as well as experiments on real-world data and illustrate that significant savings can be achieved over corresponding sample size estimates for uniform random sampling. We also outlined the context of big data applications that warrant superior sampling strategies for cost and computation reasons, and described how our method could be easily used by data providers and data users.

Translating the probabilistic bounds used in our approach to task-level bounds (e.g., bounds on deviation in sentiment analysis) would be an interesting direction for future research. Another direction is to see whether the sufficient sample sizes may be tightened in the context of our results in Section 6.3. Adapting the probabilistic bounds to time varying word occurrence rates and its application to online sampling streams and dynamic stratification derived from text clustering [2] could be considered in future. There are interesting engineering issues that are pertinent to the uptake of our method. For example, a data provider maintaining a library of different stratifications of data would benefit from heuristically choosing a subset of stratifications to run *StratSam* against; a heuristic that can choose geo-stratification when the set of keywords are to do with highly geo-focused events such as the UK EU Referendum would enable the data provider to achieve computational cost savings.

REFERENCES

- [1] Jisun An and Ingmar Weber, 'Whom should we sense in social sensing - analyzing which users work best for social media now-casting', *EPJ Data Science*, (2015).
- [2] Vipin Balachandran, P Deepak, and Deepak Khemani, 'Interpretable and reconfigurable clustering of document datasets by deriving word-based rules', *Knowledge and information systems*, **32**(3), 475–503, (2012).
- [3] David Blei, Andrew Ng, and Michael Jordan, 'Latent dirichlet allocation', in *Journal of Machine Learning Research*, (2003).
- [4] Venkatesan T. Chakaravarthy, Vinayaka Pandit, and Yogish Sabharwal, 'Analysis of sampling techniques for association rule mining', in *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pp. 276–283, (2009).
- [5] Herman Chernoff, 'A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations', *The Annals of Mathematical Statistics*, 493–507, (1952).
- [6] William G. COCHRAN, *Sampling Techniques*, Wiley, 1977.
- [7] Saptarshi Ghosh, Muhammad Bilal Zafar, Parantapa Bhattacharya, Naveen Sharma, Niloy Ganguly, and Krishna Gummadi, 'On sampling the wisdom of crowds: Random vs. expert sampling of the twitter stream', in *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, CIKM '13*, pp. 1739–1744, New York, NY, USA, (2013). ACM.
- [8] R.O. Gilbert, *Statistical Methods For Environmental Pollution Monitoring*, Van Nostrand, New York., 1987.
- [9] Sandra Gonzalez-Bailon, Ning Wang, Alejandro Riveroc, Javier Borge-Holthoefer, and Yamir Moreno, 'Assessing the bias in samples of large online networks', *Social Networks*, **38**, 16–27, (2014).
- [10] Carol Huang, 'Facebook and twitter key to arab spring uprisings: report', in *The National*, volume 6, (2011).
- [11] David Inouye and Jugal K Kalita, 'Comparing twitter summarization algorithms for multiple post summaries', in *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pp. 298–306. IEEE, (2011).
- [12] Kenneth Joseph, Peter M Landwehr, and Kathleen M Carley, 'Two 1% s dont make a whole: Comparing simultaneous samples from twitters streaming api', in *Social Computing, Behavioral-Cultural Modeling and Prediction*, 75–83, Springer, (2014).
- [13] Olga Kolchyna, Tharsis TP Souza, Philip Treleaven, and Tomaso Aste, 'Twitter sentiment analysis', *arXiv preprint arXiv:1507.00955*, (2015).
- [14] Fred Morstatter, Jürgen Pfeffer, and Huan Liu, 'When is it biased?: Assessing the representativeness of twitter's streaming api', in *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, pp. 555–556, (2014).
- [15] Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen M. Carley, 'Is the sample good enough? comparing data from twitter's streaming API with twitter's firehose', in *Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, July 8-11, 2013.*, (2013).
- [16] Deepan Subrahmanian Palguna, Vikas Joshi, Venkatesan T. Chakaravarthy, Ravi Kothari, and L. Venkata Subramaniam, 'Analysis of sampling algorithms for twitter', in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 967–973, (2015).
- [17] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo, 'Earthquake shakes twitter users: real-time event detection by social sensors', in *Proceedings of the 19th international conference on World wide web*, pp. 851–860. ACM, (2010).
- [18] Beaux Sharifi, Mark-Anthony Hutton, and Jugal K. Kalita, 'Experiments in microblog summarization', in *Proceedings of the 2010 IEEE Second International Conference on Social Computing, SOCIALCOM '10*, pp. 49–56, Washington, DC, USA, (2010). IEEE Computer Society.
- [19] S. K. Thompson, *Sampling*, Wiley, 2012.
- [20] Yu Wang, Eugene Agichtein, and Michele Benzi, 'Tm-lda: Efficient online modeling of latent topic transitions in social media', in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12*, pp. 123–131, New York, NY, USA, (2012). ACM.

A Minimization-Based Approach to Iterated Multi-Agent Belief Change

Paul Vicol¹ and James Delgrande¹ and Torsten Schaub^{2,3}

Abstract. We investigate minimization-based approaches to iterated belief change in multi-agent systems. A network of agents is represented by an undirected graph, where propositional formulas are associated with vertices. Information is shared between vertices via a procedure where each vertex minimizes disagreement with other vertices in the graph. Each iterative approach takes into account the proximity between vertices, with the underlying assumption that information from nearby sources is given higher priority than information from more distant sources. We have identified two main approaches to iteration: in the first approach, a vertex takes into account the information at its immediate neighbours only, and information from more distant vertices is propagated via iteration; in the second approach, a vertex first takes into account information from distance-1 neighbours, then from distance-2 neighbours, and so on, in a prioritized fashion. There prove to be three distinct ways to define the second approach, so in total we have four types of iteration. We define these types formally, find relationships between them, and investigate their basic logical properties. We also implemented the approaches in a software system called *Equibel*.

1 INTRODUCTION

We investigate several approaches for iterated belief change in multi-agent systems, each based on minimizing disagreements between agents in a prioritized manner. A problem instance is an undirected graph with formulas attached to vertices. Information is shared between vertices via a process of minimization over the graph. Previous work [2] dealt with one-shot belief change, where every vertex updates its beliefs through a *global* minimization process, with a weak notion of distance between vertices in a graph. We generalize this work to model iterated approaches, where distance between vertices is explicitly taken into account. One approach is for each vertex to repeatedly update its beliefs by taking into account only the beliefs of its immediate neighbours; another is for a vertex to take into account the beliefs of its distance-1 neighbours, and then take into account the beliefs of its distance-2 neighbours, and so on. We show that this second approach can be defined in three different ways, which lead to different behaviours. We implemented the approaches described in this paper in a software system called *Equibel*, available at <https://github.com/asteroidhouse/equibel>.

To motivate this work, we consider two interpretations of a graph: the first is where the graph represents a network of communicating agents, and the second is where the graph represents some general domain (such as a spatial domain), with local information contained

at each vertex. Iterated belief change in these settings can be understood as follows. In the multi-agent setting, each agent *consistently incorporates* information from other agents in a stepwise fashion, where the agent is more inclined to trust close acquaintances compared to more distant ones. In the general setting, the goal is to get an overall picture of the state of the world by combining information from multiple sources.

To illustrate one of the approaches to iterated belief change, consider a specific example in which a graph models a weather-sensing system, where vertices represent weather stations from which observations are made, and edges encode the adjacency of the spatial regions where the stations are located. The goal is to determine what information holds at a region by combining the information that is known to hold at that region (through local observations) with information coming from neighbouring regions. By the assumption of *spatial persistence* between adjacent regions, information from nearby regions should be prioritized over information from more distant regions. Informally, an approach to do this is as follows. We first consider the observations from directly adjacent regions (i.e., vertices at distance 1), and determine which observations minimize disagreement with the observations of the first region; then, we consider the observations from the next-nearest regions (i.e., vertices at distance 2), and we find observations which further minimize disagreement with the observations from distance 1, and so on. This can be seen as prioritized minimization of disagreements with respect to increasingly large *neighbourhoods* around a vertex of interest.

Multi-agent approaches in which an agent considers the beliefs of other agents in increasingly large neighbourhoods can be thought of as modeling a *group conversation* — an agent takes into account the beliefs of other agents, as well as the connections between those agents, in order to decide how to update its beliefs so as to minimize disagreement with the group. An agent tries to satisfy all protagonists to the greatest possible degree, in a prioritized manner (where nearby agents are given higher priority than distant agents).

The next section discusses related work in multi-source merging. Section 3 defines the iterative approaches we examine. Section 4 presents results concerning the relationships between the approaches. It proves to be the case that each of the approaches we propose yields non-comparable results when applied iteratively. Section 5 shows basic logical properties of the approaches. Sections 6 and 7 discuss our work and present the conclusion, respectively.

2 RELATED WORK

Most work on updating knowledge bases given new information stems from the AGM approach to belief revision [1, 7]. Belief merging [8, 5, 10] can be seen as an extension of belief revision to situ-

¹ Simon Fraser University, Burnaby, BC, {pvicol@sfu.ca, jim@cs.sfu.ca}

² University of Potsdam, Potsdam, Germany, {torsten@cs.uni-potsdam.de}

³ INRIA, Rennes, France

ations involving multiple belief bases, where the goal is to combine several, possibly conflicting, bases into a coherent whole.

Here we provide some background behind standard approaches to merging. Given a language $\mathcal{L}_{\mathcal{P}}$, a belief base K is a finite set of propositional formulas, and a belief profile $\mathcal{K} = \langle K_1, \dots, K_n \rangle$ is a finite vector consisting of n belief bases which are not necessarily pairwise different. A merging operator Δ is a function $\mathcal{L}_{\mathcal{P}} \times \mathcal{L}_{\mathcal{P}}^n \rightarrow \mathcal{L}_{\mathcal{P}}$ that associates a formula μ and a belief profile \mathcal{K} with a new formula $\Delta_{\mu}(\mathcal{K})$, which is called the merged belief state. The operator Δ aims at consistently merging the beliefs in \mathcal{K} under the integrity constraint given by μ . A set of nine postulates denoted **(IC0)**-**(IC8)** have been proposed to capture the notion of rational belief merging. These are called the Integrity Constraint (IC) merging postulates [9], listed below:

- (IC0)** $\Delta_{\mu}(\mathcal{K}) \vdash \mu$
- (IC1)** If $\mu \not\vdash \perp$, then $\Delta_{\mu}(\mathcal{K}) \not\vdash \perp$
- (IC2)** If $\bigwedge_{K \in \mathcal{K}} K \wedge \mu \not\vdash \perp$, then $\Delta_{\mu}(\mathcal{K}) \equiv \bigwedge_{K \in \mathcal{K}} K \wedge \mu$
- (IC3)** If $\mathcal{K}_1 \equiv \mathcal{K}_2$ and $\mu_1 \equiv \mu_2$, then $\Delta_{\mu_1}(\mathcal{K}_1) \equiv \Delta_{\mu_2}(\mathcal{K}_2)$
- (IC4)** If $K_1 \vdash \mu$, $K_2 \vdash \mu$ and $\Delta_{\mu}(\langle K_1, K_2 \rangle) \wedge K_1 \not\vdash \perp$, then $\Delta_{\mu}(\langle K_1, K_2 \rangle) \wedge K_2 \not\vdash \perp$
- (IC5)** $\Delta_{\mu}(\mathcal{K}_1) \wedge \Delta_{\mu}(\mathcal{K}_2) \vdash \Delta_{\mu}(\mathcal{K}_1 \sqcup \mathcal{K}_2)$
- (IC6)** If $\Delta_{\mu}(\mathcal{K}_1) \wedge \Delta_{\mu}(\mathcal{K}_2)$ is consistent, then $\Delta_{\mu}(\mathcal{K}_1 \sqcup \mathcal{K}_2) \vdash \Delta_{\mu}(\mathcal{K}_1) \wedge \Delta_{\mu}(\mathcal{K}_2)$
- (IC7)** $\Delta_{\mu_1}(\mathcal{K}) \wedge \mu_2 \vdash \Delta_{\mu_1 \wedge \mu_2}(\mathcal{K})$
- (IC8)** If $\Delta_{\mu_1}(\mathcal{K}) \wedge \mu_2$ is consistent, then $\Delta_{\mu_1 \wedge \mu_2}(\mathcal{K}) \vdash \Delta_{\mu_1}(\mathcal{K}) \wedge \mu_2$

Any operator Δ that satisfies these postulates is called an *IC merging operator*. Classical approaches to belief merging, such as [9] and [11], begin with a *set* of belief bases and produce a single, merged base. Our approach differs from these in that we deal with updating *multiple belief bases simultaneously*.

Distance-based merging operators $\Delta^{d,f}$ are characterized by a pseudo-distance d (that is, d does not have to satisfy the triangle inequality) between models and an aggregation function $f : \mathbb{R}^+ \times \dots \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$ [8]. Commonly used distances include the *drastic distance* (that is 0 if two models are equal and 1 otherwise), and the Hamming distance (the number of atoms on which two models differ). Our approach to minimization uses a set-theoretic distance between models, which is distinct from any of the standard distances used by IC merging operators.

[6] presents a framework for updating the beliefs of a group of agents via an iterated merge-and-revise procedure. The paper introduces *conciliation operators* which map a belief profile to a new belief profile in each step of the process. These operators are defined in terms of IC merging operators and the revision operators they induce. The authors describe two approaches: in the skeptical one, each agent gives priority to its previous beliefs over the merged beliefs of the group, while in the credulous one, each agent views the merged beliefs of the group as more important than its previous beliefs. The work in [6] focuses on the issue of updating a belief profile; it does not consider graphs. Since each agent considers the beliefs of all other agents simultaneously, the conciliation approach would correspond in our approach to connecting all agents in a complete graph.

More closely related to our approach is [13]. Belief Revision Games (BRGs) are games that model the dynamics of the beliefs of a group of communicating agents. Beliefs are associated with nodes in a directed graph. In each iterative step, each agent updates its be-

liefs by considering the beliefs of its neighbours. The authors define 18 different *revision policies* based on various IC merging operators that an agent can use to combine its beliefs with those of its neighbours. Each revision policy ascribes a different level of importance to the beliefs of an agent's neighbours compared to the current belief of the agent itself. The revision policies range from one in which an agent completely relinquishes its prior belief and replaces it with the merged beliefs of the group, to one where an agent does not give up its initial beliefs, but strengthens its opinion by incorporating consistent information from its neighbours.

The REV!GIS system [14] deals with belief revision in geographic information systems, using information at one location to revise adjacent locations. This conforms closely to the interpretation of a graph as representing a spatial-domain. BRELS [12] is a framework for integrating information from multiple sources, using an approach that combines merging, revision, and update.

In this paper, we build on the general framework for belief change introduced in [2] and [3]. Our approach to minimizing change between vertices in a graph is similar to that used in *belief extrapolation* [4], which can be seen as minimization of change in a chain graph where vertices represent successive points in time.

3 ITERATIVE APPROACHES

3.1 Preliminaries

We work with a propositional language \mathcal{L} defined over a finite alphabet $\mathcal{P} = \{p, q, r, \dots\}$ of atoms. We use the constants \top (resp. \perp) to represent formulas that are always true (resp. false), and the connectives $\neg, \wedge, \vee, \rightarrow$, and \leftrightarrow to construct formulas in the standard way. An interpretation of \mathcal{L} is an assignment of truth values to the atoms in \mathcal{P} . We represent an interpretation by the set of atoms that are *true* in the interpretation. For example, given $\mathcal{P} = \{p, q, r\}$, the interpretation where p is false but q and r are true is expressed by the set $\{q, r\}$. The set of all interpretations of \mathcal{L} is denoted \mathcal{W} . Given a formula $\alpha \in \mathcal{L}$ and an interpretation $w \in \mathcal{W}$, we write $w \models \alpha$ iff w makes α true in the usual truth-functional way; then we say that w is a *model* of α . We denote the set of models of α by $Mod(\alpha)$. If ω is a model over the finite alphabet \mathcal{P} , let $form(\omega) = \bigwedge_{p \in \omega} p \wedge \bigwedge_{p \in (\mathcal{P} \setminus \omega)} \neg p$. For a *set* of models $\Gamma \subseteq \mathcal{W}$, let $form(\Gamma) = \bigvee_{\omega \in \Gamma} form(\omega)$. For a set A , let $P(A)$ denote its *power set*, i.e., the set of all subsets of A .

3.2 Model Graphs

In this paper, we consider only connected, undirected graphs $G = \langle V, E \rangle$, where the vertices are identified by an initial sequence of natural numbers, e.g. for $|V| = n$, we have $V = \{1, \dots, n\}$.

Definition 3.1 (G-Scenario). Let $G = \langle V, E \rangle$ be a graph. A *G-scenario* is a function $\sigma : V \rightarrow \mathcal{L}$ that associates a propositional formula with each vertex in the graph. σ is *consistent* iff $\sigma(v)$ is consistent for all $v \in V$.

Next, we define a graph-theoretic representation for a graph G and an associated G -scenario σ , to make explicit the process by which information is shared between vertices. The idea is that a vertex $v \in V$ with formula $\sigma(v)$ is replaced by a *set* of vertices representing the models of $\sigma(v)$. For each edge $(v, w) \in E$, the vertices representing the models of $\sigma(v)$ and those representing the models of $\sigma(w)$ are connected in a complete bipartite graph. Each edge of the complete bipartite graph is given a label representing the level of disagreement between the models it connects. Various approaches for updating the

information at a vertex can be defined in terms of selecting one or more models corresponding to each original vertex, such that the labels of the edges involved are collectively minimal in some way. Next we formally define the *model graph* corresponding to a *base graph* G and an associated G -scenario.

Definition 3.2 (Model Graph). Let $G = \langle V, E \rangle$ be a graph, and let σ be a G -scenario. For a vertex $v \in V$, $\lambda(v) = \{(v, m) \mid m \in \text{Mod}(\sigma(v))\}$ is the set of *model vertices* corresponding to v . For an edge $(v, w) \in E$, $\delta(v, w) = \lambda(v) \times \lambda(w)$ is the set of *model edges* corresponding to (v, w) . The *model graph* of G under σ , denoted $\mathfrak{J}(G, \sigma)$, is the graph $\mathfrak{J}(G, \sigma) = \langle \bigcup_{v \in V} \lambda(v), \bigcup_{(v, w) \in E} \delta(v, w) \rangle$.

For a model vertex (v, m) , let $M((v, m)) = m$, so $M : V \times \mathcal{W} \rightarrow \mathcal{W}$ is a function that extracts the *model* from a model vertex. For a set Γ of model vertices, let $M(\Gamma) = \{M((v, m)) \mid (v, m) \in \Gamma\}$.

As an example, given the base graph shown in Figure 1, the corresponding model graph is shown in Figure 2. Here we have $\mathcal{P} = \{p, q, r\}$ and $\sigma(1) = p \wedge \neg q$, so $\text{Mod}(\sigma(1)) = \{\{p\}, \{p, r\}\}$. Thus, $\lambda(1) = \{(1, \{p\}), (1, \{p, r\})\}$. This is represented in Figure 2 by the rectangle labelled **1** that contains nodes labelled $\{p\}$ and $\{p, r\}$. Each edge in Figure 2 is labelled by the symmetric difference of the models at its endpoints.

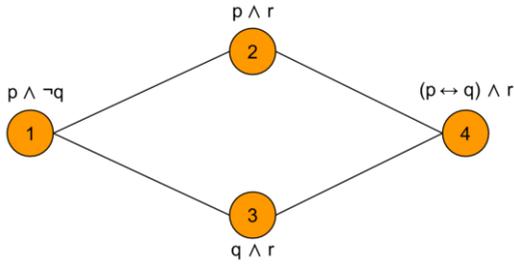


Figure 1. Base graph

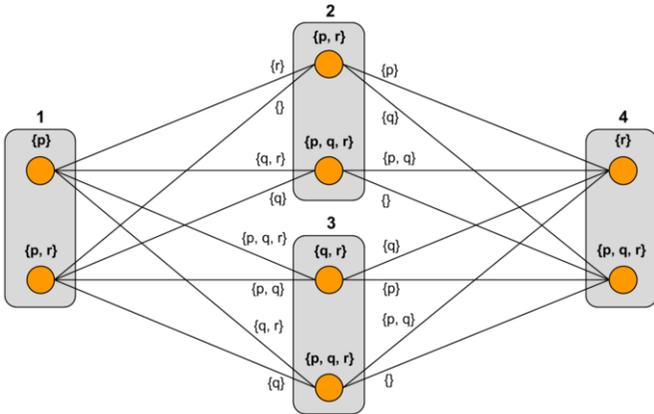


Figure 2. Model graph corresponding to the base graph

3.3 Types of Neighbourhoods

The iterative approaches we define in the next section are based on the notion of *distance prioritization*, which informally states that a vertex gives higher priority to information coming from nearby sources compared to more distant sources. In order to deal with

sources of information *within a certain distance* from a vertex of interest, we define *neighbourhoods* about vertices. In this section, we give formal definitions of the neighbourhoods we consider. In the following, let $\text{dist}(v, w)$ denote the length of the shortest path between nodes v and w , where we define $\text{dist}(v, v) = 0$.

Definition 3.3 (Neighbourhood, Pseudo-Neighbourhood). Let $G = \langle V, E \rangle$ be a graph.

- A *neighbourhood* of $v \in V$ is a connected subgraph $G' = \langle V', E' \rangle$ of G such that $v \in V'$.
- A *pseudo-neighbourhood* of $v \in V$ is a connected graph $G' = \langle V', E' \rangle$ where $v \in V' \subseteq V$ and $E' \subseteq V' \times V'$.

Note that a pseudo-neighbourhood is not a subgraph of G , i.e., there is no relationship between the edges in E and those in E' . Shortest-path trees are a natural starting point to define neighbourhoods consisting of nodes within a certain distance of a given node.

Definition 3.4 (Shortest-Path Tree). Given a graph $G = \langle V, E \rangle$, a *shortest-path tree* rooted at a vertex $v \in V$ is a spanning tree T of G , such that the path between the root v and any other vertex u in T is the shortest path between v and u in G . In graphs with un-weighted edges, shortest-path trees are equivalent to breadth-first search trees. Define a *shortest-path tree of radius r rooted at v* to be a breadth-first search tree of depth r from v .

Definition 3.5 (Cross-Edge). Let $G = \langle V, E \rangle$ be a graph, and let $v \in V$. A *cross-edge* for v is an edge $(s, t) \in E$ such that $\text{dist}(v, s) = \text{dist}(v, t)$.

Cross-edges are not included in shortest-path trees (because they break the tree property). Using the above definitions, we propose four types of neighbourhoods to be considered for iterative approaches. The following neighbourhood types are defined in alternative ways with respect to a *root* or *central* node v and a *radius* $r \in \mathbb{N}$:

1. Use Shortest-Path Trees (SPTs) to define neighbourhoods. There may be multiple SPTs from a vertex to a specified radius, because there may be multiple shortest paths between two nodes. Let $\{\text{SPT}_j(v, r) \mid j \in \{1, \dots, k\}\}$ be the set of all distinct SPTs of radius r rooted at v . One type of neighbourhood about v is an *arbitrary SPT of depth r* : $\text{SPT}_j(v, r)$.
2. Use the union of all SPTs to a specified radius from a certain node: $\text{USPT}(v, r) = \bigcup_j \text{SPT}_j(v, r)$. This removes the ambiguity involved in selecting an arbitrary SPT, and makes sense intuitively, since by considering all possible shortest paths from the root node to all other nodes in the neighbourhood, we look at all possible means of propagation of information to the root node. Note that a USPT neighbourhood does not contain cross-edges.
3. Use the union of SPTs together with all cross-edges; we call this a *complete neighbourhood*. Let $D(v, r) = \{w \mid \text{dist}(v, w) \leq r\}$. We define the *complete neighbourhood about v of radius r* to be the graph $W(v, r) = \langle D(v, r), \{(i, j) \mid (i, j) \in E, i \in D(v, r), j \in D(v, r)\} \rangle$. That is, the complete neighbourhood consists of all nodes within distance r of v , and all the edges that involve those nodes as endpoints.
4. Use the *ring* of nodes at distance r from the root node, i.e., find nodes at distance r and construct a star graph that connects those nodes directly to the root node. We define the *ring about v of radius r* to be the graph $R(v, r) = \langle T(v, r), S(v, r) \rangle$ where $T(v, r) = \{v\} \cup \{w \mid w \in V, \text{dist}(v, w) = r\}$ and $S(v, r) = \{(w, v) \mid w \in V, \text{dist}(v, w) = r\}$.

Note that alternatives 1-3 are *neighbourhoods* of v , while alternative 4 is a *pseudo-neighbourhood* of v . In the discussion of our approaches, we use the diameter of a graph and eccentricity of a node, defined as follows:

Definition 3.6 (Eccentricity, Diameter). Let $G = \langle V, E \rangle$ be a graph. The *eccentricity* of a node $v \in V$, denoted $\epsilon(v)$, is the *greatest shortest distance* between v and any other vertex: $\epsilon(v) = \max_{w \in V} \text{dist}(v, w)$. The *diameter* of the graph G , denoted $\text{diam}(G)$, is the maximum eccentricity of any vertex: $\text{diam}(G) = \max_{v \in V} \epsilon(v)$.

For simplicity and uniformity in the definitions of different approaches, for any neighbourhood type $N \in \{\text{SPT}_j, \text{USPT}, W, R\}$ and any vertex v , we *restrict* the radius of the neighbourhood so that it cannot exceed the eccentricity of v :

$$N(v, r) = \begin{cases} N(v, r) & \text{if } r < \epsilon(v) \\ N(v, \epsilon(v)) & \text{if } r \geq \epsilon(v) \end{cases}$$

By definition $\text{diam}(G) = \max_{v \in V} \epsilon(v)$, so for every node $v \in V$, $N(v, \text{diam}(G)) = N(v, \epsilon(v))$. Note that the complete neighbourhood is the only one with the property that if we look at a neighbourhood with a very large radius about any node, then that is equivalent to considering the original graph G . So $N(v, \text{diam}(G)) = G$ when $N = W$ (complete neighbourhood), but in general $N(v, \text{diam}(G)) \neq G$ when $N \neq W$ (for example, if N is a union of shortest-path trees).

3.4 Defining the Approaches

In this section, we define four approaches to iterated belief change, which we call the *simple*, *augmenting*, *expanding*, and *ring* approaches. Each approach involves a different procedure to *select model vertices* corresponding to each node in a graph. The following definition formalizes the notion of a *model selection*.

Definition 3.7 (Model Selection). Let $G = \langle V, E \rangle$ be a graph. A *model selection* is a function $s : P(V \times \mathcal{W}) \rightarrow V \times \mathcal{W}$ that selects exactly one model vertex from the set $\lambda(v)$ of model vertices for each $v \in V$. Let $S(G)$ be the *set of all possible model selections over G* .

- For a subgraph $G' = \langle V', E' \rangle$ of G and a model selection $s \in S(G)$, let $s(G')$ denote the selection s restricted to G' .
- For $N \in \{\text{SPT}_j, \text{USPT}, W, R\}$, let $S_N(v, r)$ be the *set of all model selections restricted to neighbourhood $N(v, r)$* . For a neighbourhood centered about a node v , and a model selection $s \in S_N(v, r)$, let $h(s) = s(\lambda(v))$, so that $h(s)$ is the model vertex selected at the central node v by the selection s . For a set $T \subseteq S_N(v, r)$ of model selections, let $h(T) = \{h(s) \mid s \in T\}$ be the *set of model vertices selected at v by any of the model selections in T* .
- For a model selection $s \in S_N(v, r)$, we define the *subselection of radius r'* , denoted $\text{sub}(s, r')$ for $r' \leq r$, to be the selection s restricted to the subgraph $N(v, r')$.

Next we define change sets induced by model selections. These change sets measure the *overall disagreement* between the models selected by a model selection at adjacent vertices.

Definition 3.8 (Change Set Induced by a Model Selection). The change set *induced* by a model selection $s \in S(G)$, denoted $\Delta(s)$, is defined as $\Delta(s) = \{(v, w), p \mid (v, w) \in E, s(\lambda(v)) = (v, a), s(\lambda(w)) = (w, b), p \in a \oplus b\}$, where $a \oplus b$ denotes the symmetric difference between sets a and b .

The following definition uses change sets to define a *preference relation* over model selections.

Definition 3.9 (Preferred Model Selections). Given two model selections $s, s' \in S$, we define $s \succeq s'$ iff $\Delta(s) \subseteq \Delta(s')$ and $s \succ s'$ iff $\Delta(s) \subset \Delta(s')$. We say that s is a *preferred or minimal model selection* iff $\nexists s' \in S : s' \succ s$. We denote the set of all preferred model selections by $\text{Pref}(S, \succeq)$.

Thus, s is preferred to s' iff the change set induced by s is included in the change set induced by s' . *Preferred* model selections are those that induce *inclusion-minimal* change sets, and therefore minimize disagreement between the models at adjacent vertices.

It is useful to distinguish between two classes of model selections: those that induce change sets *not induced by other model selections* (which we call *unique model selections*), and those that induce change sets that are *also induced by other model selections* (which we call *duplicated model selections*).

Definition 3.10 (Unique and Duplicated Model Selections). Let Γ be a set of model selections. Then, $q(\Gamma) = \{s \in \Gamma \mid \nexists s' \in \Gamma : s' \neq s \text{ and } \Delta(s') = \Delta(s)\}$ and $d(\Gamma) = \{s \in \Gamma \mid \exists s' \in \Gamma : s' \neq s \text{ and } \Delta(s') = \Delta(s)\}$. The functions q and d define *unique* and *duplicated* model selections, respectively, referring to the existence of other model selections that induce identical change sets. These functions *partition* Γ , so $\Gamma = q(\Gamma) \cup d(\Gamma)$ and $q(\Gamma) \cap d(\Gamma) = \emptyset$.

We are particularly interested in the case where $\Gamma = \text{Pref}(S(G), \succeq)$. For a model selection s to be *strictly preferred* to another selection s' , it must be the case that for *any* subgraph g of G , $s(g) \succeq s'(g)$ and for *some* subgraph g' of G , $s(g') \succ s'(g')$. A special case of this is when we consider subgraphs of G that are neighbourhoods of different radii about a node v . Then, in order for selection s to be strictly preferred to selection s' , we must have: $\forall r. 1 \leq r \leq \text{diam}(G) : s(N(v, r)) \succeq s'(N(v, r))$ and $\exists r. 1 \leq r \leq \text{diam}(G) : s(N(v, r)) \succ s'(N(v, r))$.

The following proposition states that a unique minimal model selection for a graph H must be contained within some minimal model selection for any supergraph I of H .

Proposition 1. *Let $G = \langle V, E \rangle$ be a graph, and let $H = \langle V', E' \rangle$ and $I = \langle V'', E'' \rangle$ be subgraphs of G such that H is a subgraph of I . If $s \in q(\text{Pref}(S(H), \succeq))$ then $\exists s' \in \text{Pref}(S(I), \succeq) : s'(H) = s$.*

Proof. Let $s \in q(\text{Pref}(S(H), \succeq))$. Consider the set of model selections $\mathcal{O} = \{s' \in S(I) \mid s'(H) = s\}$; that is, the set of all model selections over I that have subselections over H equal to s . \mathcal{O} is a partially-ordered set with respect to \succeq , and thus has minimal elements. Denote the set of minimal elements of \mathcal{O} by $\text{Min}(\mathcal{O}, \succeq)$. Take an arbitrary element $z \in \text{Min}(\mathcal{O}, \succeq)$. We want to show that $z \in \text{Pref}(S(I), \succeq)$. Suppose not. Then there must be a model selection $z' \in \text{Pref}(S(I), \succeq)$ such that $z' \succ z$. In order for z' to be strictly preferred to z we must have that for any subgraph g of G , $z'(g) \succeq z(g)$. In particular, we must have $z'(H) \succeq z(H) = s$. But since $s \in q(\text{Pref}(S(H), \succeq))$, if $z'(H) \succeq s$, then we must have $z'(H) = s$. Since $z' \in S(I)$ and $z'(H) = s$, we have $z' \in \mathcal{O}$. But $z \in \text{Min}(\mathcal{O}, \succeq)$, so $z' \not\succeq z$, which is a contradiction. Thus, $z \in \text{Pref}(S(I), \succeq)$ and $z(H) = s$, so we have constructed a model selection with the desired properties. This proves that $\exists s' \in \text{Pref}(S(I), \succeq) : s'(H) = s$. \square

Corollary 1.1. *If $x \in \{s(\lambda(v)) \mid s \in q(\text{Pref}(S(H), \succeq))\}$ then $x \in \{s(\lambda(v)) \mid s \in \text{Pref}(S(I), \succeq)\}$.*

A consequence of this is that $h(q(\text{Pref}(S_N(v, r), \succeq))) \subseteq h(\text{Pref}(S_N(v, r+1), \succeq))$. This states that if a model vertex is selected at a node v by a *unique minimal selection* s in a neighbourhood of radius r about v , then that model vertex must be selected at v in every neighbourhood of radius greater than r (i.e., that model vertex will not be eliminated when considering larger neighbourhoods). The next proposition gives an analogous result for *duplicated minimal selections*.

Proposition 2. *Let $G = \langle V, E \rangle$ be a graph, and let $H = \langle V', E' \rangle$ and $I = \langle V'', E'' \rangle$ be subgraphs of G such that H is a subgraph of I . If $s \in d(\text{Pref}(S(H), \succeq))$ then $\exists s' \in \text{Pref}(S(I), \succeq) : s'(H) \in \{t \in \text{Pref}(S(H), \succeq) \mid \Delta(t) = \Delta(s)\}$.*

Proof. This proof is nearly identical to the proof of Proposition 1; we omit it due to space constraints. \square

Corollary 2.1. *At least one of the model vertices selected at the central node v by a duplicated minimal model selection over H must be selected by a minimal model selection over I : $\{s(\lambda(v)) \mid s \in d(\text{Pref}(S(H), \succeq))\} \cap \{s(\lambda(v)) \mid s \in \text{Pref}(S(I), \succeq)\} \neq \emptyset$.*

A consequence of this for concentric neighbourhoods is:

$$h(d(\text{Pref}(S_N(v, r), \succeq))) \cap h(\text{Pref}(S_N(v, r+1), \succeq)) \neq \emptyset$$

Before defining the iterative approaches we investigate, we define the *global completion* operation studied previously [2], which performs *one-shot* (as opposed to iterative) belief change. This allows us to compare the global approach with our iterative approaches in Section 4.

Definition 3.11 (Global Completion). Let $G = \langle V, E \rangle$ be a graph, and let σ be an associated G -scenario. We define $C(v) = h(\text{Pref}(S_W(v, \text{diam}(G)), \succeq)) = \{s(\lambda(v)) \mid s \in \text{Pref}(S(G), \succeq)\}$. The *global completion* of σ , denoted σ_C , is the G -scenario such that $\forall v \in V : \sigma_C(v) = \text{form}(M(C(v)))$.

Now we define the *simple*, *augmenting*, and *expanding* approaches to iteration, by defining three functions F , A , and E , respectively, that *select specific model vertices* for each node that minimize disagreement between the beliefs of that node and those of its neighbours.

For a model graph \mathfrak{J} of $G = \langle V, E \rangle$ under σ , the set of model selections over \mathfrak{J} is denoted by $S_{\mathfrak{J}}$. Given an approach $\Omega \in \{F, A, E\}$ and neighbourhood type $N \in \{SPT_j, USPT, W\}$, for any iteration $i > 0$, $\Omega_N^i(v)$ is the *set of model vertices* selected for v by Ω in iteration i using neighbourhood N , and we define

$$\mathfrak{J}_{\Omega, N}^i = \left\langle \bigcup_{v \in V} \Omega_N^i(v), \bigcup_{(v, w) \in E} \Omega_N^i(v) \times \Omega_N^i(w) \right\rangle$$

to be the *model graph* that results from the i^{th} iteration of approach Ω using neighbourhood N . The set of all model selections over $\mathfrak{J}_{\Omega, N}^i$ is $S_{\mathfrak{J}_{\Omega, N}^i}$, which we abbreviate to $S_{\Omega, N}^i$. In this context, we let $S_{\Omega, N}^0$ stand for $S_{\mathfrak{J}}$, so that $S_{A, N}^0 = S_{E, N}^0 = S_{F, N}^0$. Then, $S_{\Omega, N}^i(v, r)$ is the set of model selections over $\mathfrak{J}_{\Omega, N}^i$, restricted to neighbourhood $N(v, r)$. Next, we define the following iterative approaches:

Definition 3.12 (Simple/Fixed-Radius Iteration). Let $G = \langle V, E \rangle$ be a graph, and let σ be an associated G -scenario. For $i \geq 1$, we define $F_N^i(v) = h(\text{Pref}(S_{F, N}^{i-1}(v, 1), \succeq))$. The G -scenario that results from the i^{th} iteration of the simple approach is denoted $\sigma_{F, N}^i$, and is defined as the G -scenario such that $\forall v \in V : \sigma_{F, N}^i(v) = \text{form}(M(F_N^i(v)))$.

In each iteration of the simple approach, a vertex changes its beliefs by considering only the beliefs of its immediate neighbours.

Definition 3.13 (Augmenting Iteration). Let $G = \langle V, E \rangle$ be a graph, and let σ be an associated G -scenario. Then for $i \geq 1$, we define $A_N^i(v) = h(T_N^i(v))$, where

$$T_N^i(v) = \{s \in \text{Pref}(S_{A, N}^{i-1}(v, \text{diam}(G)), \succeq) \mid \forall r. 1 \leq r \leq \text{diam}(G) : \text{sub}(s, r) \in \text{Pref}(S_{A, N}^{i-1}(v, r), \succeq)\}$$

The G -scenario that results from the i^{th} iteration of the augmenting approach is denoted $\sigma_{A, N}^i$, and is defined as the G -scenario such that $\forall v \in V : \sigma_{A, N}^i(v) = \text{form}(M(A_N^i(v)))$.

A model selection $s \in T_N^i(v)$ can be constructed stepwise by: 1) starting from v and considering preferred model selections over a radius 1 neighbourhood about v ; then 2) *augmenting* those preferred selections by considering *minimal model edges* that cross the boundary between the radius 1 and radius 2 neighbourhoods about v (as well as the model edges *between* distance-2 nodes, in the case of complete neighbourhoods), and so on. Once we find preferred model selections over a radius 1 neighbourhood, we take into account *only* the model vertices (and model edges) involved in those selections for future steps of the procedure; at each radius, we examine ways to *augment* preferred model selections from the preceding radius.

Definition 3.14 (Expanding Iteration). Let $G = \langle V, E \rangle$ be a graph, and let σ be an associated G -scenario. For $i \geq 1$, we define $E_N^i(v) = I_{i, N}^{\epsilon(v)}(v)$, where

$$I_{i, N}^1(v) = h(\text{Pref}(S_{E, N}^{i-1}(v, 1), \succeq))$$

and for $r > 1$,

$$I_{i, N}^r(v) = h(\text{Pref}(\{s \in S_{E, N}^{i-1}(v, r) \mid s(\lambda(v)) \in I_{i, N}^{r-1}(v)\}, \succeq))$$

Note that $I_{i, N}^{r+1}(v) \subseteq I_{i, N}^r(v)$. The G -scenario that results from the i^{th} iteration of the expanding approach is denoted $\sigma_{E, N}^i$, and is defined as the G -scenario where $\forall v \in V : \sigma_{E, N}^i(v) = \text{form}(M(E_N^i(v)))$.

The definition of expanding iteration captures the following intuition: at radius r , a subset of model vertices are selected at v that are associated with preferred model selections in the neighbourhood of radius r about v . When we expand to radius $r+1$, we select a subset of models at v associated with preferred model selections in the radius $r+1$ neighbourhood *from the models that remain after the selection at radius r* .

In addition to the three iterative approaches defined above, that are based on minimal model selections, we introduce another approach called the *ring method*, which is defined in terms of generic merging operators. The ring method involves *iterated merging* of beliefs from nodes in different “layers” about a particular node, where layer r consists of nodes at distance r from that node. In contrast to the other approaches, the ring method does not consider the topology of the graph between the central node and nodes at distance r .

Definition 3.15 (Ring Iteration). Let $G = \langle V, E \rangle$ be a graph, and let σ be an associated G -scenario. The G -scenario that results from the i^{th} iteration of the ring method is denoted σ_R^i , and is defined as follows. Let $C_i^r(v) = \{\sigma_R^{i-1}(w) \mid \text{dist}(v, w) = r\}$ be the *context* at distance r from v , in the i^{th} iteration of the ring method. Then, $\sigma_R^0 = \sigma$ and for $i \geq 1$, we have $R_i^0(v) = \sigma_R^{i-1}(v)$ and $R_i^t(v) = \Delta_{R_i^{t-1}(v)}(C_i^t(v))$. σ_R^i is defined as the G -scenario where $\forall v \in V : \sigma_R^i(v) = R_i^{\epsilon(v)}(v)$.

In the i^{th} step of the i^{th} ring iteration, we merge the belief profile $C_i^t(v)$ under the integrity constraint $R_i^{t-1}(v)$, which represents the result of the *previous step of the i^{th} iteration*. Δ can be any merging operator, but certain properties of the approach can be shown if Δ satisfies certain IC merging properties.

For simplicity, when we discuss the global completion and the ring approach together with the other approaches (simple, augmenting, and expanding), we do not explicitly denote the neighbourhood type, because the global and ring approaches are *not* defined to use different neighbourhoods. In this case, the neighbourhood type can be arbitrarily chosen.

Each of the four iterative approaches reaches a fixpoint, because each monotonically reduces the set of models at each node. This leads to the following observation:

Observation 1 (Monotonicity). *For $\Omega \in \{A, E, F, R\}$, we have: $\forall v \in V : \sigma_{\Omega}^{i+1} \models \sigma_{\Omega}^i$.*

For an approach $\Omega \in \{A, F, E, R\}$, denote by σ_{Ω}^* the fixpoint reached by iterating the approach, so that $\sigma_{\Omega}^* = \sigma_{\Omega}^t$ where t is the smallest integer for which $\sigma_{\Omega}^t = \sigma_{\Omega}^{t+1}$.

We observe that when Δ is the *consistency-based projection merging operator* [3], the ring method can be expressed as expanding iteration performed with respect to *ring neighbourhoods*, so that $\sigma_{R}^i = \sigma_{E,R}^i$.

4 RELATIONSHIPS BETWEEN APPROACHES

In the previous section we discussed one global approach to belief change (C), and four iterative approaches: simple iteration (F), expanding iteration (E), augmenting iteration (A), and ring iteration (R). Here we show how these approaches relate to one another (i.e., whether two approaches are comparable, and if so, which approach is logically stronger or weaker than the other). The notion of logical strength of G -scenarios is defined below:

Definition 4.1 (Equivalence and Logical Strength of G -Scenarios). Let G be a graph, and σ and σ' be two G -scenarios.

- We say that σ and σ' are *equivalent*, denoted $\sigma \equiv \sigma'$, iff for all $v \in V$ we have $\models \sigma(v) \equiv \sigma'(v)$.
- We say that σ is *at least as strong* as σ' , denoted $\sigma \models \sigma'$, iff $\models \sigma(v) \rightarrow \sigma'(v)$ for all $v \in V$.

Given $\Omega, \Omega' \in \{A, E, F, R, C\}$ and $r, t \geq 1$, we say that σ_{Ω}^r and $\sigma_{\Omega'}^t$ are *non-comparable* iff there exists a graph $G = \langle V, E \rangle$ and scenario σ such that $\sigma_{\Omega}^r \not\models \sigma_{\Omega'}^t$ and $\sigma_{\Omega'}^t \not\models \sigma_{\Omega}^r$.

First we give a result concerning the relationship between neighbourhoods. The following proposition states that adding a new edge between two nodes in a graph cannot logically strengthen the results of iteration, i.e., the results are either equivalent or weaker.

Proposition 3. *Let $G = \langle V, E \rangle$ be a graph, and σ be a G -scenario. Let $G' = \langle V', E' \rangle$ be a subgraph of G such that $\exists v, w \in V' : [(v, w) \in E \text{ and } (v, w) \notin E']$. Let (v, w) be such an edge, and let $G'' = \langle V', E' \cup \{(v, w)\} \rangle$ be a copy of G' that contains (v, w) . Then, $\text{Pref}(S(G'), \succeq) \subseteq \text{Pref}(S(G''), \succeq)$.*

Corollary 3.1. *For $\Omega \in \{A, E, F\}$, we have $\forall i > 0 : \sigma_{\Omega, SPT_j}^i \models \sigma_{\Omega, USPT}^i$ and $\forall i > 0 : \sigma_{\Omega, USPT}^i \models \sigma_{\Omega, W}^i$.*

That is, for any of the approaches A , E , or F , arbitrary SPT neighbourhoods yield the strongest results, followed by union-of-SPT neighbourhoods, and finally by complete neighbourhoods.

Propositions 4-11 compare the *first iterations* of all the approaches. Then, Corollaries 8.1 and 9.1 and Propositions 12-15 compare their *fixpoints*. For Propositions 4-7 and 10, let G be an arbitrary connected, undirected graph, let σ be an arbitrary G -scenario, and let $N \in \{SPT_j, USPT, W\}$ be an arbitrary neighbourhood.

Proposition 4. $\sigma_{A,N}^1 \models \sigma_{F,N}^1$

Proof. By definition, every model selection s found by the augmenting approach is preferred over restricted neighbourhoods of all radii $1 \leq r \leq \text{diam}(G)$; in particular, $\text{sub}(s, 1) \in \text{Pref}(S_{F,N}^0(v, 1), \succeq)$, which implies that $s(\lambda(v)) \in F_N^1(v)$. Thus, $\sigma_{A,N}^1 \models \sigma_{F,N}^1$. \square

Proposition 5. $\sigma_{A,N}^1 \models \sigma_{E,N}^1$

Proof. Due to space constraints, we just give an outline of this proof. Let $B_{A,N}(v, r) = \{s \in \text{Pref}(S_{A,N}^0(v, r), \succeq) \mid \forall r'. 1 \leq r' \leq r : \text{sub}(s, r') \in \text{Pref}(S_{A,N}^0(v, r'), \succeq)\}$ and let $B_{E,N}(v, 1) = \text{Pref}(S_{E,N}^0(v, 1), \succeq)$ and for $r > 1$, $B_{E,N}(v, r) = \text{Pref}(\{s \in S_{E,N}^0(v, r) \mid s(\lambda(v)) \in I_{1,N}^{r-1}(v)\}, \succeq)$. We prove by induction that every minimal model selection of radius r about an arbitrary node v found by the augmenting approach is also found by the expanding approach, so $B_{A,N}(v, r) \subseteq B_{E,N}(v, r)$. When $r = 1$, we have $S_{A,N}^0(v, 1) = S_{E,N}^0(v, 1)$, so $B_{A,N}(v, 1) = B_{E,N}(v, 1)$. For the inductive step, assume that $B_{A,N}(v, r) \subseteq B_{E,N}(v, r)$. Take $s \in B_{A,N}(v, r+1)$ and let $t = \text{sub}(s, r)$. By definition, $t \in B_{A,N}(v, r)$, and by the inductive assumption $t \in B_{E,N}(v, r)$. There are two cases: if $t \in q(B_{E,N}(v, r))$, then we use Proposition 1 to show that $s \in B_{E,N}(v, r+1)$; if $t \in d(B_{E,N}(v, r))$, then we use Proposition 2 to show that $s \in B_{E,N}(v, r+1)$. It follows by induction that $\sigma_{A,N}^1 \models \sigma_{E,N}^1$. \square

Proposition 6. $\forall i \geq 1 : \sigma_{A,N}^i \models \sigma_C$

Proof. If $m \in A_W^1(v)$, then by definition, $m \in h(\text{Pref}(S_W(v, \text{diam}(G)), \succeq))$, so $m \in C(v)$. Since $\forall v \in V : A_W^{i+1}(v) \subseteq A_W^i(v)$, we have $\forall v \in V, \forall i \geq 1 : A_W^i(v) \subseteq C(v)$. Thus, $\forall i \geq 1 : \sigma_{A,W}^i \models \sigma_C$. Since $\sigma_{A,USPT}^i \models \sigma_{A,W}^i$ and $\sigma_{A,SPT_j}^i \models \sigma_{A,USPT}^i$ we see that for $N \in \{SPT_j, USPT, W\}$, $\forall i \geq 1 : \sigma_{A,N}^i \models \sigma_C$. \square

Proposition 7. $\sigma_{E,N}^1 \models \sigma_{F,N}^1$

Proof. By the definition of expanding iteration, $\sigma_{E,N}^1(v) = \text{form}(M(E_N^1(v)))$, where $E_N^1(v) = I_{1,N}^{\epsilon(v)}(v)$. We have $I_{1,N}^1(v) = h(\text{Pref}(S_{E,N}^0(v, 1), \succeq)) = h(\text{Pref}(S_{F,N}^0(v, 1), \succeq)) = F_N^1(v)$, where $S_{E,N}^0 = S_{F,N}^0$ for any neighbourhood type. Since $I_{i,N}^{r+1}(v) \subseteq I_{i,N}^r(v)$, we have $E_N^1(v) = I_{1,N}^{\epsilon(v)}(v) \subseteq I_{1,N}^1(v) = F_N^1(v)$. This implies that $\sigma_{E,N}^1 \models \sigma_{F,N}^1$. \square

Proposition 8. $\forall i \geq 1 : \sigma_{E,N}^i$ and σ_C are in general non-comparable.

Proof. We begin with a counterexample to the statement that $\sigma_{E,N}^1$ and σ_C are always comparable: Let $G = \langle V, E \rangle$, where $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2), (2, 3), (3, 4), (4, 5)\}$, and let σ be a G -scenario such that $\sigma = \langle 1 : q \vee (r \wedge \neg s), 2 : p, 3 : (\neg p \wedge \neg q) \vee s, 4 : \top, 5 : \neg s \rangle$. Then, the first iteration of the expanding approach yields $\sigma_{E,N}^1$, where $\sigma_{E,N}^1(1) = \sigma_{E,N}^1(2) = \sigma_{E,N}^1(3) = p \wedge q \wedge s$ and $\sigma_{E,N}^1(4) = \sigma_{E,N}^1(5) = \neg p \wedge \neg q \wedge \neg s$, while the global completion yields $\sigma_C = \langle 1 : (p \wedge q \wedge s) \vee (p \wedge \neg q \wedge r \wedge \neg s), 2 : (p \wedge q \wedge s) \vee (p \wedge \neg q \wedge r \wedge \neg s), 3 : (p \wedge q \wedge s) \vee (\neg p \wedge \neg q \wedge r \wedge \neg s), 4 : (p \wedge q) \vee (\neg p \wedge \neg q \wedge r \wedge \neg s), 5 : (p \wedge q \wedge \neg s) \vee (\neg p \wedge \neg q \wedge r \wedge \neg s) \rangle$. We

see that $\sigma_{E,N}^1(4) \not\equiv \sigma_C(4)$ and $\sigma_C(4) \not\equiv \sigma_{E,N}^1(4)$. For this example, we have $\forall i > 1 : \sigma_{E,N}^i = \sigma_{E,N}^1$, so $\forall i \geq 1 : \sigma_{E,N}^i$ and σ_C are in general non-comparable. \square

Corollary 8.1. $\sigma_{E,N}^*$ and σ_C are non-comparable.

Proposition 9. $\forall i \geq 1 : \sigma_{F,N}^i$ and σ_C are in general non-comparable.

Proof. We begin with a counterexample to the statement that $\sigma_{F,N}^1$ and σ_C are always comparable: Let $G = \langle V, E \rangle$, where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (2, 3), (3, 4)\}$, and let σ be a G -scenario such that $\sigma = \langle 1 : \neg r, 2 : \neg p \vee r, 3 : (p \vee q) \wedge r, 4 : \neg q \rangle$. Then the result of the first simple iteration is $\sigma_{F,N}^1 = \langle 1 : \neg p \wedge \neg r, 2 : (p \wedge r) \vee (\neg p \wedge q), 3 : p \wedge \neg q \wedge r, 4 : p \wedge \neg q \wedge r \rangle$ while the result of the global completion is $\sigma_C = \langle 1 : (\neg p \vee \neg q) \wedge \neg r, 2 : (\neg p \wedge \neg r) \vee (p \wedge \neg q \wedge r), 3 : r \wedge (p \vee q) \wedge (\neg p \vee \neg q), 4 : \neg q \wedge r \rangle$. We see that $\sigma_{F,N}^1(2) \not\equiv \sigma_C(2)$ and $\sigma_C(2) \not\equiv \sigma_{F,N}^1(2)$. For this example, we have $\sigma_{F,N}^2 = \langle 1 : \neg p \wedge q \wedge \neg r, 2 : (\neg p \wedge q) \vee (p \wedge \neg q \wedge r), 3 : p \wedge \neg q \wedge r, 4 : p \wedge \neg q \wedge r \rangle$. We find that $\sigma_{F,N}^2(2) \not\equiv \sigma_C(2)$ and $\sigma_C(2) \not\equiv \sigma_{F,N}^2(2)$, so the second iteration of the simple approach is in general non-comparable with the global completion. In addition, $\forall i > 2 : \sigma_{F,N}^i = \sigma_{F,N}^2$, so $\forall i \geq 1 : \sigma_{F,N}^i$ and σ_C are non-comparable. \square

Corollary 9.1. $\sigma_{F,N}^*$ and σ_C are non-comparable.

Now, we show that the first iteration of the ring method produces logically stronger results than the first iteration of the simple approach.

Proposition 10. $\sigma_R^1 \models \sigma_{F,N}^1$ if the merging operator Δ used in the ring method is the consistency-based projection merging operator.

Proof. From the definition of the ring method, $R_1^0(v) = \sigma_R^0(v) = \sigma(v)$, and $R_1^1(v) = \Delta_{R_1^0(v)}(C_1^1(v)) = \Delta_{\sigma(v)}(C_1^1(v))$. Note that $\Delta_{\sigma(v)}(C_1^1(v)) = \sigma_{F,USPT}^1(v) = \sigma_{F,SPT}^1(v)$ and $\Delta_{\sigma(v)}(C_1^1(v)) \models \sigma_{F,W}^1(v)$, so we can say that $\Delta_{\sigma(v)}(C_1^1(v)) \models \sigma_{F,N}^1(v)$ regardless of the type of neighbourhood used for simple iteration. For all $i > 1$, $R_1^i(v) \models R_1^1(v)$; in particular, $R_1^{e(v)}(v) \models R_1^1(v)$, and since $R_1^1(v) \models \sigma_{F,N}^1(v)$, we have $\sigma_R^1 \models \sigma_{F,N}^1$. \square

The following proposition shows that the first iteration of the ring method is in general non-comparable to the first iterations of augmenting iteration, expanding iteration, or the global completion.

Proposition 11. σ_R^1 is in general not comparable with any of $\sigma_{A,N}^1$, $\sigma_{E,N}^1$, or σ_C .

Proof. All these statements can be proven by a single example for which augmenting iteration, expanding iteration, and the global completion produce the *same result*, which is different from, and not comparable to, the result produced by the ring method. Consider the graph $G = \langle V, E \rangle$ where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (2, 3), (3, 4)\}$ and let σ be a G -scenario such that $\sigma = \langle 1 : p \oplus q, 2 : p, 3 : \top, 4 : \neg p \rangle$. We find that $\sigma_{A,N}^1 = \sigma_{E,N}^1 = \sigma_C = \langle 1 : p \wedge \neg q, 2 : p \wedge \neg q, 3 : \neg q, 4 : \neg p \wedge \neg q \rangle$, while $\sigma_R^1 = \langle 1 : p \wedge \neg q, 2 : p \wedge \neg q, 3 : p \oplus q, 4 : \neg p \wedge q \rangle$. Note that $\sigma_{A,N}^1(4) = \sigma_{E,N}^1(4) = \sigma_C(4) = \neg p \wedge \neg q$, while $\sigma_R^1(4) = \neg p \wedge q$. Since $\neg p \wedge \neg q \not\equiv \neg p \wedge q$ and $\neg p \wedge q \not\equiv \neg p \wedge \neg q$, we conclude that the first iterations of expanding iteration, augmenting iteration, and the global completion are in general non-comparable with the first iteration of the ring method. \square

Propositions 12-15 compare the *fixpoints* of the augmenting, expanding, simple, and ring approaches.

Proposition 12. $\sigma_{E,N}^*$ and $\sigma_{F,N}^*$ are in general non-comparable.

Proof. Counterexample to the statement that $\sigma_{E,N}^*$ and $\sigma_{F,N}^*$ are always comparable: Let $G = \langle V, E \rangle$, where $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{(i, i+1) \mid 1 \leq i \leq 5\}$, and let σ be a G -scenario such that $\sigma = \langle 1 : \neg r, 2 : \top, 3 : \neg p \vee r, 4 : (p \vee q) \wedge r, 5 : \top, 6 : \neg q \rangle$. The fixpoint of simple iteration is $\sigma_{F,N}^* = \langle 1 : \neg p \wedge q \wedge \neg r, 2 : \neg p \wedge q \wedge \neg r, 3 : \neg p \wedge q \wedge r, 4 : p \wedge \neg q \wedge r, 5 : p \wedge \neg q \wedge r, 6 : p \wedge \neg q \wedge r \rangle$. The fixpoint of expanding iteration is $\sigma_{E,N}^* = \langle 1 : \neg p \wedge q \wedge \neg r, 2 : \neg p \wedge q \wedge \neg r, 3 : p \wedge \neg q \wedge r, 4 : p \wedge \neg q \wedge r, 5 : p \wedge \neg q \wedge r, 6 : p \wedge \neg q \wedge r \rangle$. We see that $\sigma_{F,N}^*(3) \not\equiv \sigma_{E,N}^*(3)$ and $\sigma_{E,N}^*(3) \not\equiv \sigma_{F,N}^*(3)$, so the simple and expanding approaches are not comparable in general. \square

Proposition 13. $\sigma_{E,N}^*$ and $\sigma_{A,N}^*$ are in general non-comparable.

Proof. Counterexample to the statement that $\sigma_{E,N}^*$ and $\sigma_{A,N}^*$ are always comparable: Let $G = \langle V, E \rangle$, where $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2), (2, 3), (3, 4), (4, 5)\}$, and let σ be a G -scenario such that $\sigma = \langle 1 : p, 2 : \top, 3 : (p \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r), 4 : \neg r, 5 : q \rangle$. Then the fixpoint of augmenting iteration is $\sigma_{A,N}^* = \langle 1 : p \wedge q \wedge r, 2 : p \wedge q \wedge r, 3 : \neg p \wedge \neg q \wedge \neg r, 4 : \neg p \wedge \neg r, 5 : \neg p \wedge q \wedge \neg r \rangle$, while the fixpoint of expanding iteration is $\sigma_{E,N}^* = \langle 1 : p \wedge \neg q \wedge r, 2 : p \wedge \neg q \wedge r, 3 : \neg p \wedge \neg q \wedge \neg r, 4 : \neg p \wedge \neg r, 5 : \neg p \wedge q \wedge \neg r \rangle$. Note that the beliefs produced by each approach at vertices 1 and 2 are not comparable, i.e., $\sigma_{E,N}^*(1) \not\equiv \sigma_{A,N}^*(1)$ and $\sigma_{A,N}^*(1) \not\equiv \sigma_{E,N}^*(1)$. \square

Proposition 14. $\sigma_{A,N}^*$ and $\sigma_{F,N}^*$ are in general non-comparable.

Proof. Counterexample to the statement that $\sigma_{A,N}^*$ and $\sigma_{F,N}^*$ are always comparable: Let $G = \langle V, E \rangle$, where $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2), (2, 3), (3, 4), (4, 5)\}$, and let σ be a G -scenario such that $\sigma = \langle 1 : p \wedge r, 2 : p, 3 : s, 4 : \neg r \vee s, 5 : \neg s \rangle$. Then the fixpoint of simple iteration is $\sigma_{F,N}^* = \langle 1 : p \wedge r \wedge s, 2 : p \wedge r \wedge s, 3 : p \wedge r \wedge s, 4 : p \wedge \neg r, 5 : p \wedge \neg r \wedge \neg s \rangle$, while the fixpoint of augmenting iteration is $\sigma_{A,N}^* = \langle 1 : p \wedge r \wedge s, 2 : p \wedge r \wedge s, 3 : p \wedge r \wedge s, 4 : p \wedge (r \leftrightarrow s), 5 : p \wedge \neg r \wedge \neg s \rangle$. Note that the beliefs produced by each approach at vertex 4 are not comparable, i.e., $\sigma_{F,N}^*(4) \not\equiv \sigma_{A,N}^*(4)$ and $\sigma_{A,N}^*(4) \not\equiv \sigma_{F,N}^*(4)$. \square

We also show that the fixpoint of the ring method is in general non-comparable to the fixpoints of augmenting iteration, expanding iteration, simple iteration, or the global completion.

Proposition 15. σ_R^* is in general not comparable with any of $\sigma_{A,N}^*$, $\sigma_{E,N}^*$, $\sigma_{F,N}^*$, or σ_C .

Proof. All these statements can be proven by a single example for which augmenting iteration, expanding iteration, simple iteration and the global completion produce the *same fixpoint*, which is different from, and not comparable to, the fixpoint produced by the ring method. Once again consider the graph and associated scenario shown in the proof of Proposition 11. We find that $\sigma_{A,N}^* = \sigma_{E,N}^* = \sigma_{F,N}^* = \sigma_C = \langle 1 : p \wedge \neg q, 2 : p \wedge \neg q, 3 : \neg q, 4 : \neg p \wedge \neg q \rangle$, while $\sigma_R^* = \langle 1 : p \wedge \neg q, 2 : p \wedge \neg q, 3 : p \wedge \neg q, 4 : \neg p \wedge q \rangle$. Note that $\sigma_{A,N}^*(4) = \sigma_{E,N}^*(4) = \sigma_{F,N}^*(4) = \sigma_C(4) = \neg p \wedge \neg q$, while $\sigma_R^*(4) = \neg p \wedge q$. Since $\neg p \wedge \neg q \not\equiv \neg p \wedge q$ and $\neg p \wedge q \not\equiv \neg p \wedge \neg q$, we conclude that the fixpoints of expanding iteration, augmenting iteration, simple iteration, and the global completion are in general non-comparable with the fixpoint of the ring method. \square

5 LOGICAL PROPERTIES

In this section we investigate logical properties that are satisfied by the approaches defined in Section 3. Observation 1 already noted the monotonicity of the iterative approaches. Here, we start with the basic property that if an initial scenario is consistent, then it remains consistent throughout every iteration of each approach.

Proposition 16 (Consistency Preservation). *Let G be a graph and σ be a G -scenario. If σ is consistent, then σ_{Ω}^i is consistent for all i and all $\Omega \in \{A, E, F, R, C\}$.*

It proves to be the case that if the conjunction of formulas at all vertices is consistent, then the result at each vertex will be just this conjunction. We formalize this as follows:

Definition 5.1 (Agreement Preservation (AP)). Let $G = \langle V, E \rangle$ be a graph, σ be a G -scenario, and σ' be the G -scenario that results from applying an approach to updating the information at the vertices in V . If $\bigwedge_{v \in V} \sigma(v)$ is consistent, then $\forall v \in V : \sigma'(v) \equiv \bigwedge_{v \in V} \sigma(v)$.

The following proposition states that the augmenting, expanding, simple, and global approaches satisfy Agreement Preservation.

Proposition 17. *For $\Omega \in \{A, E, F, C\}$, σ_{Ω}^* satisfies (AP).*

Proof. This was shown for $\Omega = C$ in [2]. If $\bigwedge_{v \in V} \sigma(v)$ is consistent, then $\bigcap_{v \in V} \text{Mod}(\sigma(v)) \neq \emptyset$. For each model $m \in \bigcap_{v \in V} \text{Mod}(\sigma(v))$, we can define a model selection $s \in S(G)$ such that for each $v \in V$, $s(\lambda(v)) = (v, m)$. Then, $\Delta(s) = \emptyset$. Clearly, s is minimal over any connected subgraph of G . So, given a vertex v , s is minimal over a neighbourhood of any radius about v . In particular, s is minimal in the radius-1 neighbourhood about v , so (v, m) is selected by the simple approach. s is minimal over neighbourhoods of all radii $1 \leq r \leq \text{diam}(G)$, so (v, m) is selected by the augmenting approach. For the expanding approach, s is minimal over a neighbourhood of radius 1, so (v, m) is selected in the first step. Since all other vertices agree on model m , (v, m) will continue to be part of the minimal selection s over increasingly large neighbourhoods. Thus, (v, m) is selected by the expanding approach. Finally, although additional models $m' \notin \bigcap_{v \in V} \text{Mod}(\sigma(v))$ may initially be selected by an approach (e.g. due to local minimization), such models are progressively eliminated when the approach is iterated, so that the fixpoint of each approach contains only the information agreed upon by all vertices. \square

We also show that the ring method satisfies the Agreement Preservation property under certain conditions.

Proposition 18. *The ring method satisfies (AP) iff Δ satisfies (IC2).*

Proof. Let $G = \langle V, E \rangle$ be a graph, and let $v \in V$ be an arbitrary vertex. Note that $C_1^1(v) \sqcup C_1^2(v) \sqcup \dots \sqcup C_1^{\epsilon(v)}(v) = \bigsqcup_{1 \leq r \leq \epsilon(v)} C_1^r(v) = \{\sigma(v) \mid v \in V\}$, where \sqcup denotes multiset union. Then, by using the definition of the ring method, we have $R_1^0(v) = \sigma(v)$ and

$$\begin{aligned} R_1^1(v) &= \Delta_{R_1^0(v)}(C_1^1(v)) = \bigwedge C_1^1(v) \wedge \sigma(v) \\ &\vdots \\ R_1^{\epsilon(v)}(v) &= \Delta_{R_1^{\epsilon(v)-1}(v)}(C_1^{\epsilon(v)}(v)) = \bigwedge_{v \in V} \sigma(v) \end{aligned}$$

So $\sigma_R^1(v) = \bigwedge_{v \in V} \sigma(v)$ for all $v \in V$. Since the beliefs of all nodes are equivalent after the first iteration of the ring method, clearly there will be no further change, so $\sigma_R^*(v) = \bigwedge_{v \in V} \sigma(v)$ for all $v \in V$. \square

The number of iterations for any approach to reach a fixpoint has an upper bound, given by the following proposition.

Proposition 19. *Given a graph $G = \langle V, E \rangle$ and a G -scenario σ , an upper bound on the number of iterations of any approach $\Omega \in \{A, E, F, R\}$ is $(\sum_{v \in V} |\text{Mod}(\sigma(v))|) - |V|$.*

Proof. By monotonicity, in each iteration the number of models at a node can either stay the same or decrease. Clearly, the smallest possible change from one iteration to the next is for a single model to be eliminated from the set of models at a specific node. The total number of models over all nodes in the graph is $\sum_{v \in V} |\text{Mod}(\sigma(v))|$. At the fixpoint (assuming σ is consistent), each node must have at least one model; the minimum number of models over the graph in any iteration is $|V|$. Thus, in the case that each iteration yields the minimal change, we see that there can be at most $(\sum_{v \in V} |\text{Mod}(\sigma(v))|) - |V|$ iterations before the scenario reaches a fixpoint. \square

6 DISCUSSION

This work calls for some comparisons with Belief Revision Games (BRGs) [13]. A BRG represents a network of communication agents, where each agent iteratively updates its beliefs by applying a *revision policy* that takes into account its current beliefs and the beliefs of its neighbours. By definition, each revision policy takes as input the current belief of an agent and the belief profile consisting of the beliefs of her neighbours, and returns a new set of beliefs. In this approach, agents do not take into account the topology of the graph beyond their immediate neighbours. The fixed-radius approach (using USPT neighbourhoods) introduced in this paper has a similar structure to BRGs, in that agents repeatedly update their beliefs by taking into account the beliefs of their immediate neighbours. However, our method of updating information by minimizing disagreements in a set-theoretic fashion is distinct from any of the IC merging operator-based revision policies. The expanding and augmenting approaches presented here further diverge from the BRG approach; they take as input an entire graph G , an associated scenario σ , and a node v , and yield a new set of beliefs for v . The expanding and augmenting approaches take into account the beliefs of agents throughout the graph, as well as the connections between those agents, to minimize disagreement within increasingly large neighbourhoods about a vertex of interest, in a prioritized manner.

7 CONCLUSION

In this paper, we have generalized the work presented in [2] by defining minimization-based approaches to iterated belief change that take into account the distance between nodes in a graph. We defined four iterated approaches which we call the simple, augmenting, expanding, and ring approaches. We showed that each approach is distinct, in that each produces results that are non-comparable with the results of the other approaches. We also compared these iterated approaches to the global approach studied previously. We found that the augmenting approach produces logically stronger results in each iteration than global completion, while the other approaches produce results that are non-comparable with global completion. We examined basic logical properties for the approaches, and stated an upper bound on the number of iterations needed to reach a fixpoint. Finally, we compared our approaches to related work.

We have also implemented the approaches to belief change described in this paper, in a software system called *Equibel*. This software is publicly available at <https://github.com/asteroidhouse/equibel>.

REFERENCES

- [1] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson, 'On the logic of theory change: Partial meet contraction and revision functions', *Journal of Symbolic Logic*, **50**(2), 510–530, (1985).
- [2] James P. Delgrande, Jérôme Lang, and Torsten Schaub, 'Belief change based on global minimisation', in *Proc. International Joint Conference on Artificial Intelligence (IJCAI'07)*, pp. 2468–2473, Hyderabad, India, (2007).
- [3] James P. Delgrande and Torsten Schaub, 'A consistency-based framework for merging knowledge bases', *Journal of Applied Logic*, **5**(3), 459–477, (2006).
- [4] Florence Dupin de Saint-Cyr and Jérôme Lang, 'Belief extrapolation (or how to reason about observations and unpredicted change)', in *Proc. of the Eighth International Conference on the Principles of Knowledge Representation and Reasoning (KR'02)*, pp. 497–508, Toulouse, France, (2002).
- [5] Patricia Everaere, Sébastien Konieczny, and Pierre Marquis, 'Quota and Gmin merging operators', in *Proc. International Joint Conference on Artificial Intelligence (IJCAI'05)*, pp. 424–429, Edinburgh, UK, (2005).
- [6] Olivier Gauwin, Sébastien Konieczny, and Pierre Marquis, 'Iterated belief merging as conciliation operators', in *Seventh International Symposium on Logical Formalizations of Commonsense Reasoning*, pp. 85–92, Corfu, Greece, (2005).
- [7] Hirofumi Katsuno and Alberto O. Mendelzon, 'Propositional knowledge base revision and minimal change', *Artificial Intelligence*, **52**(3), 263–294, (1991).
- [8] Sébastien Konieczny, Jérôme Lang, and Pierre Marquis, 'Distance-based merging: A general framework and some complexity results', in *Proc. of the Eighth International Conference on the Principles of Knowledge Representation and Reasoning (KR'02)*, pp. 97–108, San Francisco, USA, (2002).
- [9] Sébastien Konieczny and Ramón Pino Pérez, 'Merging information under constraints: A logical framework', *Journal of Logic and Computation*, **12**(5), 773–808, (2002).
- [10] Sébastien Konieczny and Ramón Pino Pérez, 'Logic based merging', *Journal of Philosophical Logic*, **40**(2), 239–270, (2011).
- [11] Paolo Liberatore and Marco Schaerf, 'Arbitration: A commutative operator for belief revision', in *Proc. of the Second World Conference on the Fundamentals of Artificial Intelligence (WOCFAI-95)*, pp. 217–228, Paris, France, (1995).
- [12] Paolo Liberatore and Marco Schaerf, 'BReLS: A system for the integration of knowledge bases', in *Proc. of the Seventh International Conference on the Principles of Knowledge Representation and Reasoning (KR'00)*, pp. 145–152, Breckenridge, CO, USA, (2000).
- [13] Nicolas Schwind, Katsumi Inoue, Gauvain Bourgne, Sébastien Konieczny, and Pierre Marquis, 'Belief revision games', in *Proc. of the Twenty-Ninth National Conference on Artificial Intelligence (AAAI-15)*, pp. 1590–1596, Austin, TX, USA, (2015).
- [14] Eric Würbel, Robert Jeansoulin, and Odile Papini, 'Revision: An application in the framework of GIS', in *Proc. of the Seventh International Conference on the Principles of Knowledge Representation and Reasoning (KR'00)*, pp. 505–515, Breckenridge, CO, USA, (2000).

Parameterised Model Checking for Alternating-Time Temporal Logic

Panagiotis Kouvaros¹ and Alessio Lomuscio¹

Abstract. We investigate the parameterised model checking problem for specifications expressed in alternating-time temporal logic. We introduce parameterised concurrent game structures representing infinitely many games with different number of agents. We introduce a parametric variant of ATL to express properties of the system irrespective of the number of agents present in the system. While the parameterised model checking problem is undecidable, we define a special class of systems on which we develop a sound and complete counter abstraction technique. We illustrate the methodology here devised on the prioritised version of the train-gate-controller.

1 Introduction

In the past 15 years there has been considerable interest in logics and techniques for reasoning about strategic play in multi-agent systems (MAS). Formalisms such as Alternating-Time Logic [4] enable us to capture precisely what agents may bring about by cooperating with one another. They also offer a game-theoretic angle to study interactions within MAS that was missing in previous mainstream formalisms, e.g., in the plain BDI approach [35].

Formalisms stronger than ATL have been put forward. The logic ATL* [4] extends the cooperation modalities to the full power of CTL* temporal modalities. More recently, a wide range of so called “strategy logics” [10, 32] have been introduced to increase the expressiveness even further. For example, in a variant of strategy logic, strategies can be explicitly named and bound to agents. This is useful in many scenarios and results in being able to express, for example, concepts such as Nash equilibria. Several underlying conditions on MAS have been explored in this context, including complete and incomplete information, perfect recall and memoryless strategies.

There is, however, a whole class of MAS that has not received attention so far in terms of *strategic reasoning*. This concerns MAS that are composed by an unbounded number of components. This is a natural and general class of MAS not only of theoretical interest, but of considerable relevance to applications. For example, open MAS where components enter and leave the system at runtime may be formalised by MAS with unbounded numbers of agents [7, 33]. Robotic swarms are also naturally assumed to be systems in which the number of agents is unbounded [27, 28, 29]. This tradition dates back to applications such as networking where protocols are often analysed by considering the unbounded nature of the processes in the system [8].

Specifications concerning strategic interplay naturally arise in these application areas. For example, in swarm analysis the engineer may be interested in establishing whether a small coalition in

a swarm can influence the behaviour of the whole system in a certain way (perhaps by forcing its direction of travel). In open MAS such as auctions it may be of interest to establish whether a small set of players can collude to force an unwanted state of affairs. Strategic interplay on games with infinitely many players is also of theoretical interest: many games are defined on arbitrarily many players including game cutting, diner’s dilemma, etc. In all these scenarios it is of interest to establish whether a *strategic property holds on the game irrespective of the number of players in the system*. By establishing the above we can deduce a strong property of the system, as the property will have been guaranteed to hold irrespective of the particular system instance under analysis.

In this paper we put forward a methodology to address this problem. We assume that our MAS is composed by arbitrarily many agents. We express specifications of interest in a novel variant of ATL, where we can naturally encode whether all players of a particular kind can, by establishing a coalition, force states of affairs expressed as LTL formulas on the rest of the system. We define the verification problem in terms of the verification of all realisations of the system, and show the problem is undecidable in general. Our key contribution is a technique based on counter abstraction that enables us, in several cases of interest, to draw conclusions on all possible system instances by analysing a single abstract model.

Related Work. We are not aware of any work addressing the problem above. Of course, the verification problem for ATL has received much attention over the years, including various implementations from the seminal jMocha model checker [2] to recent symbolic implementations such as Verics [24] and MCMAS [31]. However, all these techniques deal with a finite number of agents, and so cannot address the general problem. Closer to our approach is recent work on parameterised verification for MAS [25, 26, 30], where unbounded MAS are checked against epistemic specifications. In particular, [27, 28] present counter abstraction techniques to verify unbounded MAS against epistemic specifications. While our technique is inspired by [27, 28], the specification language supported here is based on strategies, and so it is not directly comparable.

Scheme of the Paper. In Section 2 we introduce parameterised concurrent game structures (PCGS), a novel form of concurrent game structures that can represent arbitrarily many agents; this is followed by the introduction of PATL, a parameterised version of ATL. In Section 3 we prove that the parameterised model checking problem for PCGS against PATL is undecidable and identify a class of systems for which we give a method for parameterised verification in Section 4. In Section 5 we encode an unbounded version of the train-gate-controller in PCGS and express relevant specifications in PATL. We conclude in Section 6, where we discuss possible future work.

¹ Department of Computing, Imperial College London, UK, email: {p.kouvaros,a.lomuscio}@imperial.ac.uk

2 Parameterised Concurrent Game Structures

To reason about strategic interplay in unbounded MAS, we introduce the formalism of *parameterised concurrent game structures* (PCGS). PCGS extend concurrent game structures [4], the traditional semantics for ATL, to represent an unbounded number of systems (or games) with an unbounded number of agents. The behaviour of the agents is given by templates which form part of the PCGS. We assume a finite number of templates from which an unbounded number of agents may be constructed. A vector of parameters specifies the number of agents in the system that are built from the templates. In other words, given a vector (n_1, \dots, n_k) and a PCGS of k templates, the concrete concurrent game structure is constructed by composing n_i agents for each template i .

We consider concrete structures where: (i) the agents have *incomplete information*, i.e., the agents are only aware of their private (local) state; (ii) the agents have *imperfect recall*, i.e., their strategies are given as functions from local states to actions; (iii) the agents' strategies are *uniform*, i.e., a strategy always specifies the same action in a given local state.

This is not the mainstream setting in which ATL was defined [4], but it is widely considered in the literature [1, 9, 22]. Observe, however, that assuming complete information, i.e., assuming that the agents are fully aware of the system's state, is problematic for unbounded systems. In fact, under complete information, each agent can be aware of the number of agents in each system instance. This immediately leads to undecidability of the parameterised model checking problem [17]. Similarly, observe that the plain model checking problem under incomplete information and perfect recall is undecidable [14]. It follows that the parameterised model checking problem is also undecidable. Finally, in the context of incomplete information and memoryless strategies we assume that strategies are *uniform* [23], i.e., when following a strategy the agents consistently select the same action in the same local state. This is an intuitive requirement that preserves the meaning of the ATL operators, only causing a jump to Δ_P^2 [22].

We define parameterised concurrent game structures. Then we define the concrete structures that these generate.

Definition 1 (Parameterised concurrent game structure) A parameterised concurrent game structure (PCGS) is a tuple $\hat{S} = \langle \hat{\Sigma}, \hat{Q}, \hat{q}_0, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \rangle$, where:

- $\hat{\Sigma} = \hat{\Sigma}_T \cup \Sigma_C$, where $\hat{\Sigma}_T = \{\hat{1}, \dots, \hat{k}\}$ is a finite, nonempty set of agent templates and $\Sigma_C = \{1, \dots, z\}$ is a finite set of concrete agents.
- \hat{Q} is a nonempty, and finite set of local states for the agents.
- $\hat{q}_0 = \hat{\Sigma} \rightarrow 2^{\hat{Q}}$ describes a set of initial local states for each of the agents.
- \hat{A} is a nonempty, and finite set of actions available to the agents.
- $\hat{\Pi}$ is a finite set of propositional variables.
- $\hat{\pi} : \hat{Q} \rightarrow 2^{\hat{\Pi}}$ is a labelling function on the states.
- $\hat{d} : \hat{\Sigma} \times \hat{Q} \rightarrow 2^{\hat{A}}$ determines the set of actions available to an agent given its local state.
- $\hat{\delta} : \hat{\Sigma} \times \hat{Q} \times \hat{A} \times 2^{\hat{A}} \rightarrow \hat{Q}$ gives the temporal evolution of an agent given its state, its action, and the projection of the joint action for the other agents into a set.

As the above definition suggests, to account for the unbounded nature of parameterised games, information regarding the identities of

the agents is abstracted away. In other words, as we will make precise with the definition of a concrete system, the temporal evolution of an agent depends on the agent's local state, its action, and the actions of the other agents, but it does not depend on how many and which agents performed each action.

Assume a PCGS \hat{S} defining k agent templates. Let $\bar{n} \in \mathbb{N}^k$ be a vector of parameters for the system. Consider $\bar{n}.i$ to be the i -th component in \bar{n} . We now define the \bar{n} -st concrete instantiation $S(\bar{n})$ of \hat{S} . $S(\bar{n})$ is a concurrent game structure resulting from the composition of the concrete agents Σ_C with $\bar{n}.i$ instantiations $\{(i, 1), \dots, (i, \bar{n}.i)\}$ of each agent template i . Let Σ denote the set of all concrete agents. The global states of $S(\bar{n})$ correspond to tuples of local states for all the agents in the system. For a global state q and an agent $ag \in \Sigma$, we write $q.ag$ for the local state of ag in q . The system's temporal evolution from a state depends on the joint action performed at the state. Given a joint action \bar{a} and an agent $ag \in \Sigma$, we write $\bar{a}.ag$ for the action of ag in \bar{a} . By $Actions(\bar{a})$, we denote the set $Actions(\bar{a}) = \{\bar{a}.ag : ag \in \Sigma\}$ of actions for all the agents in \bar{a} ; by $Actions_{-ag}(\bar{a})$, we denote the set $Actions_{-ag}(\bar{a}) = \{\bar{a}.ag' : ag' \in \Sigma \setminus \{ag\}\}$ of actions for all the agents other than ag in \bar{a} .

Definition 2 (Concrete concurrent game structure) Let $\hat{S} = \langle \hat{\Sigma}, \hat{Q}, \hat{q}_0, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \rangle$ be a PCGS of k agent templates and z concrete agents. Let $\bar{n} \in \mathbb{N}^k$. A concrete concurrent game structure (CCGS) is a tuple $S(\bar{n}) = \langle \Sigma, Q, Q_0, A, \Pi, \pi, d, \delta \rangle$, where:

- $\Sigma = \Sigma_T \cup \Sigma_C$, where $\Sigma_T = \{(i, j) : 1 \leq i \leq k, 1 \leq j \leq \bar{n}.i\}$ is the set of concrete agents instantiated from the templates.
- $Q = \hat{Q}^{|\Sigma|}$ is the set of concrete states.
- $Q_0 = (\hat{q}_0(\hat{1}))^{\bar{n}.1} \times \dots \times (\hat{q}_0(\hat{k}))^{\bar{n}.k} \times (\hat{q}_0(1)) \times \dots \times (\hat{q}_0(z))$ is the set of initial concrete states.
- $A = \hat{A}^{|\Sigma|}$ is the set of joint actions.
- $\Pi = \hat{\Pi} \times \Sigma$.
- $\pi : Q \rightarrow 2^{\Pi}$ is defined as $(p, ag) \in \pi(q)$ iff $p \in \hat{\pi}(q.ag)$.
- $d : Q \rightarrow 2^A$ gives the set of joint actions available at a state of the system: $\bar{a} \in d(q)$ iff

$$\bar{a}.i \in \hat{d}(i, q.i) \text{ for every agent } i \in \Sigma_C, \text{ and}$$

$$\bar{a}.(i, j) \in \hat{d}(i, q.(i, j)) \text{ for every agent } (i, j) \in \Sigma_T.$$

- $\delta : Q \times A \rightarrow Q$ is the temporal transition function of the system: $\delta(q, \bar{a}) = q'$ iff $\bar{a} \in d(q)$,

$$\hat{\delta}(q.i, \bar{a}.i, Actions_{-i}(\bar{a})) = q'.i \text{ for every agent } i \in \Sigma_C, \text{ and}$$

$$\hat{\delta}(q.(i, j), \bar{a}.(i, j), Actions_{-(i, j)}(\bar{a})) = q'.(i, j) \text{ for every agent } (i, j) \in \Sigma_T.$$

So a PCGS generates different CCGS depending on the vector of parameters for the system. Each CCGS is composed of a different number of agents. The propositional variables in a CCGS are indexed by each of the concrete agents. A propositional variable (p, ag) holds in a global state if the agent ag is at a local state labelled with the non-indexed propositional variable p by the template labelling function. These assumptions are crucial in expressing *collective* specifications that typically accommodate games with an unbounded number of agents such as cooperative games [36] or agreement protocols [34].

A path in a CCGS is a sequence $\lambda = q_0 q_1 \dots$ such that for every $i \geq 0$ there is a joint action \bar{a} with $\delta(q_i, \bar{a}) = q_{i+1}$. For a path λ and

$i \geq 0$, we write λ^i to denote the suffix $q_i q_{i+1} \dots$ of λ . We assume that the transition relation for CCGS is serial ².

Alternating-time temporal logic (ATL) generalises *branching-time temporal logic* (CTL) by allowing selective quantification over paths representing possible outcomes of a system [4]. Hence, ATL specifications can express strategic profiles of agents that can enforce a certain outcome. More specifically, in addition to customary temporal modalities, ATL includes *cooperation* formulas such as $\langle\langle \Gamma \rangle\rangle \varphi$, where Γ is a group of agents, expressing that Γ has a cooperative strategy to enforce an LTL formula φ irrespective of how the other agents act.

Let $VAR = VAR_1 \cup \dots \cup VAR_k$ be the union of disjoint sets of variable symbols, where each VAR_i is associated with agent template \hat{i} . To reason about unbounded groups of agents, we here define parameterised alternating-time temporal logic (PATL), a variation of ATL, where: (i) Γ is not a subset of agents from a predetermined set as in ATL, but it is a subset $\Delta \subseteq \Sigma_C \cup VAR$ of the union of the set of concrete agents Σ_C and the set of variables VAR ; (ii) the atomic propositions are indexed by the variables in VAR . The domain of a variable $v \in VAR_i$ appearing in a formula φ depends on the CCGS on which φ is evaluated; if φ is evaluated on $S(\bar{n})$, then the potential set of values for v is $\{(i, 1), \dots, (i, \bar{n}.i)\}$. Intuitively, PATL formulae quantify over the concrete agents. For instance, $\forall v \langle\langle \{v\} \rangle\rangle \varphi$, where $v \in VAR_i$, expresses that every concrete agent from template \hat{i} can enforce φ independently of the action performed by *however many* other agents are instantiated from each template.

Given a set Σ of concrete agents, a set $VAR = VAR_1 \cup \dots \cup VAR_k$ of variable symbols, and a set $\hat{\Pi}$ of propositional variables, PATL formulae are defined by the following BNF grammar:

$$\varphi ::= \top \mid (p, v) \mid \neg \varphi \mid \varphi \wedge \varphi \mid \langle\langle \Delta \rangle\rangle X \varphi \mid \langle\langle \Delta \rangle\rangle (\varphi U \varphi) \mid \langle\langle \Delta \rangle\rangle G \varphi \mid \forall \varphi$$

where p is a propositional variable, v is a variable symbol, and $\Delta \subseteq \Sigma \cup VAR$.

We abbreviate $\langle\langle \Delta \rangle\rangle (\top U \varphi)$ as $\langle\langle \Delta \rangle\rangle F \varphi$. The formula $\langle\langle \Delta \rangle\rangle X \varphi$ is read as “ $\langle\langle \Delta \rangle\rangle$ has a strategy to enforce φ in the next state”; $\langle\langle \Delta \rangle\rangle G \varphi$ denotes “ $\langle\langle \Delta \rangle\rangle$ has a strategy to enforce φ forever in the future”; and $\langle\langle \Delta \rangle\rangle \varphi_1 U \varphi_2$ represents “ $\langle\langle \Delta \rangle\rangle$ has a strategy to enforce that φ_2 holds at some point in the future and φ_1 holds until then”.

A variable appearing in a PATL formula is said to be *free* if it is not in the scope of a universal quantifier. A PATL formula is said to be a *sentence* if there are no free variables appearing in the formula. We here consider only PATL sentences. We say that a PATL sentence is an \bar{m} -indexed formula, where \bar{m} is a k -tuple of natural numbers, if there are precisely $\bar{m}.i$ variables from VAR_i appearing in a cooperation modality or a propositional variable in the formula.

The key difference of PATL with respect to plain ATL is that while a formula in ATL expresses what a concrete group can achieve irrespective of the actions of all other agents outside the group, a PATL formula quantifies over the groups of concrete agents that can enforce the property independently of how many concrete agents are actually instantiated; i.e., in any system of any size.

Assume a CCGS $S(\bar{n}) = \langle \Sigma, Q, Q_0, A, \Pi, \pi, d, \delta \rangle$. We now describe the evaluation of PATL sentences on $S(\bar{n})$. This uses the notion of a *strategy*. A strategy for a concrete agent $i \in \Sigma_C$ is a function $f_i: \hat{Q}^+ \rightarrow \hat{A}$ such that for each finite sequence $\hat{\lambda} \in \hat{Q}^+$ we have that $f_i(\hat{\lambda}) \in \hat{d}(i, \hat{q})$, where \hat{q} is the last state in $\hat{\lambda}$. Similarly, a strategy for

a concrete agent $(i, j) \in \Sigma_T$ is a function $f_{(i,j)}: \hat{Q}^+ \rightarrow \hat{A}$ such that for each finite sequence $\hat{\lambda} \in \hat{Q}^+$ we have that $f_{(i,j)}(\hat{\lambda}) \in \hat{d}(\hat{i}, \hat{q})$, where \hat{q} is the last state in $\hat{\lambda}$. A strategy is said to be *memoryless* if it only depends on the last state, i.e., $f_{ag}(\hat{q}_1, \dots, \hat{q}_x, \hat{q}) = f_{ag}(\hat{q})$, for any $x \in \mathbb{N}$. In the following we consider memoryless strategies.

Given a set Γ of concrete agents, a concrete state $q \in Q$, and an indexed set of strategies $F_\Gamma = \{f_{ag} : ag \in \Gamma\}$, the *outcomes* of F_Γ from q is defined as the set of infinite paths $out(q, F_\Gamma)$, where $q_0 q_1 \dots \in out(q, F_\Gamma)$ iff $q_0 = q$, and for every $x \geq 0$ there is a joint action \bar{a} such that: (i) $\bar{a}.ag = f_{ag}(q_0.ag \dots q_x.ag)$ for all agents $ag \in \Gamma$; (ii) $\delta(q_x, \bar{a}) = q_{x+1}$.

We now define the satisfaction relation. We write $(S(\bar{n}), q) \models \varphi$ to mean that a formula φ is true at state q in $S(\bar{n})$. If $S(\bar{n})$ is clear, then we simplify the notation to $q \models \varphi$.

Definition 3 (Satisfaction of PATL) *Let $S(\bar{n})$ be a CCGS, q a state in $S(\bar{n})$, (p, ag) a propositional variable, and $v \in VAR_i$ a variable symbol. The satisfaction relation \models is inductively defined as follows:*

$$\begin{aligned} q \models (p, ag) & \quad \text{iff } (p, ag) \in \Pi(q); \\ q \models \neg \varphi & \quad \text{iff } q \not\models \varphi; \\ q \models \varphi_1 \wedge \varphi_2 & \quad \text{iff } q \models \varphi_1 \text{ and } q \models \varphi_2; \\ q \models \langle\langle \Gamma \rangle\rangle X \varphi & \quad \text{iff for some } F_\Gamma \text{ and all } \lambda \in out(q, F_\Gamma) \text{ we have that } \lambda^1 \models \varphi; \\ q \models \langle\langle \Gamma \rangle\rangle \varphi_1 U \varphi_2 & \quad \text{iff for some } F_\Gamma \text{ and all } \lambda \in out(q, F_\Gamma), \text{ there is } i \geq 0 \text{ with } \lambda^i \models \varphi_2 \text{ and } \lambda^j \models \varphi_1 \text{ for all } 0 \leq j < i; \\ q \models \langle\langle \Gamma \rangle\rangle G \varphi & \quad \text{iff for some } F_\Gamma \text{ and all } \lambda \in out(q, F_\Gamma) \text{ and for all } i \geq 0 \text{ we have that } \lambda^i \models \varphi; \\ q \models \forall v \varphi & \quad \text{iff for all } ag \in \{(i, 1), \dots, (i, \bar{n}.i)\}, q \models \varphi[v \mapsto ag]. \end{aligned}$$

A PATL formula φ is said to be true in $S(\bar{n})$, denoted $S(\bar{n}) \models \varphi$, if $(S(\bar{n}), q) \models \varphi$ for every $q \in Q_0$.

PATL generalises indexed CTL [12], a parametric variant of CTL that introduces quantification operators over the system components. In addition to the next-time operator, the unrestricted nesting of the quantification operators can be used to represent the actual number of participants in the system [12], thereby making the parameterised model checking problem undecidable [11]. To circumvent this, indexed CTL typically excludes the next-time operator and is restricted to its prenex fragment in which all the quantifiers appear at the front of the formula [5]. In light of this, for the rest of the paper, we consider \bar{m} -indexed PATL formulae that comply to the following schema:

$$\forall v_{(1,1)} \dots \forall v_{(k,\bar{m}.k)} \left(\left(\bigwedge_{i,j} \neg(v_{(1,i)} = v_{(1,j)}) \wedge \dots \wedge \bigwedge_{i,j} \neg(v_{(k,i)} = v_{(k,j)}) \right) \rightarrow \varphi \right)$$

where φ is a PATL formula with no quantifiers and without the next-time operator that is built from precisely the variables $v_{(i,1)}, \dots, v_{(i,\bar{m}.i)}$, for each agent template \hat{i} . We simply write φ to denote an \bar{m} -indexed formula of the above schema.

The evaluation of an \bar{m} -indexed formula on a CCGS is determined by evaluating the conjunction of all its ground instantiations under any assignment for the variables. For example, assume a resource sharing protocol encoded as a PCGS with one

² A serial transition relation is induced by a given PCGS by assuming a *serial* action such that $serial \in \hat{d}(ag, q)$ and $\hat{\delta}(ag, q, serial, X) = q$, for each $q \in \hat{Q}$, $ag \in \hat{\Sigma}$, and $X \subseteq \hat{A}$.

agent template, and consider that want to check the property: “every agent has a strategy so that they eventually take the lock on the shared resource”. We can express this property in PATL by the 1-indexed formula $\langle\langle\{v\}\rangle\rangle F(lock, v)$. When evaluated on a concrete system with two participants, the formula denotes the formula $\langle\langle\{(1, 1)\}\rangle\rangle F(lock, (1, 1)) \wedge \langle\langle\{(1, 2)\}\rangle\rangle F(lock, (1, 2))$. Following the symmetric nature of these conjuncts, an \bar{m} -indexed formula can be evaluated by considering only its ground instantiation obtained by assigning pairwise distinct values to the variables in each VAR_i from the domain $\{(i, 1), \dots, (i, \bar{m}.i)\}$ [30]. For example, $\langle\langle\{v\}\rangle\rangle F(lock, v)$ can be evaluated by simply considering its ground instantiation $\langle\langle\{(1, 1)\}\rangle\rangle F(lock, (1, 1))$. For the rest of the paper, an \bar{m} -indexed formula equivalently refers to its aforementioned ground instantiation.

3 Parameterised Model Checking

We now introduce a procedure to assess the correctness of a MAS with respect to a given PCGS independently of the number of agents in the system. This decision problem is generally known as the *parameterised model checking problem* [8].

Definition 4 (PMCP) *Given a PCGS \hat{S} and an \bar{m} -indexed PATL formula φ , the parameterised model checking problem (PMCP) is the decision problem of determining whether the following holds:*

$$S(\bar{n}) \models \varphi \text{ for every } \bar{n} \geq \bar{m}^3.$$

The PCMP is known to be undecidable for arbitrary systems [6]. However, restrictions can be imposed on the systems leading to decidable problems; these have been exploited in a variety of applications ranging from networking to MAS against temporal or epistemic specifications [13, 18, 30].

We prove below that the PMCP for PCGS is undecidable. To achieve this, we show that PCGS can simulate broadcast protocols [16], whose PMCP has been shown undecidable for liveness properties [19]. This result shows the generality of systems that can be described by PCGS. Additionally, it will give us the intuition to define a special class of PCGS that enjoys a decidable PMCP.

A broadcast protocol is a tuple $B = (S, S_0, \Lambda, R)$, where S is a finite set of states, $S_0 \subseteq S$ is a set of initial states, $\Lambda = L \times \{!\} \cup L \times \{??\}$ is the union of finite sets of output broadcast labels ($L \times \{!\}$) and input broadcast labels ($L \times \{??\}$), and $R \subseteq S \times \Lambda \times S$ is a transition relation [16]. A concrete broadcast system $B(n)$ can be constructed from B and a parameter $n \in \mathbb{N}$ representing the number of processes in the system. The processes communicate via broadcast primitives, i.e., processes send messages that are received by all other processes. Formally, there is a transition from a global state g to a global state g' by means of label a if there is a process i such that $(g.i, a!!, g'.i) \in R$, and for all processes $j \neq i$, $(g.j, a??, g'.j) \in R$.

Theorem 1 *The PMCP for PCGS is undecidable.*

Proof sketch. Let $B = (S, S_0, \Lambda, R)$ be a broadcast protocol. We construct a PCGS $\hat{S} = \langle \hat{S}, \hat{Q}, \hat{q}_0, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \rangle$ that simulates B (see Figure 1):

- $\Sigma_C = \emptyset$, $\Sigma_T = \{\hat{1}\}$; $\hat{Q} = S$; $\hat{q}_0(\hat{1}) = S_0$; $\hat{A} = \Lambda$; $\hat{\Pi}$ and $\hat{\pi}$ can be arbitrarily defined.
- \hat{d} is defined as follows: for each $q \in S$ and $a \in \Lambda$ with $(q, a, q') \in R$, for some q' , we have that $a \in \hat{d}(\hat{1}, q)$.

³ $\bar{n} \geq \bar{m}$ is a shortcut for $\bar{n}.i \geq \bar{m}.i$, for each $1 \leq i \leq k$.

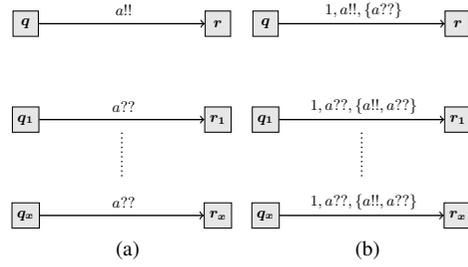


Figure 1: Simulation of broadcast protocols by PCGS.

- $\hat{\delta}$ is given by the following ⁴: for each $(q, a!!, q') \in R$, we have that $\hat{\delta}(\hat{1}, q, a!!, \{a??\}) = q'$; and for each $(q, a??, q') \in R$, we have that $\hat{\delta}(\hat{1}, q, a??, \{a!!, a??\}) = q'$.

It follows that for every $n \in \mathbb{N}$, and for every sequence of states $\lambda \in (S^n)^+$, λ is a path in $B(n)$ iff λ is a path in $S(n)$. But the PMCP for broadcast protocols against liveness properties is undecidable [16]. Since liveness can be expressed in PATL, it follows that the PMCP for PCGS against PATL specifications is also undecidable. ■

In the proof above note the crucial role of the transition $\hat{\delta}(\hat{1}, q, a!!, \{a??\}) = q'$. By the concrete semantics, the transition is enabled for a concrete agent if the agent exclusively performs $a!!$, and all other agents perform $a??$. Intuitively this encodes a situation where the agent takes the lock on the shared resource q' . We define the class of *mutual exclusion* PCGS on the basis of this interpretation.

We write $q \rightarrow_a q'$ to mean that there is a template transition $\hat{\delta}(\hat{a}g, q, a, X) = q'$ such that $a \in X$ for some agent template $\hat{a}g$. That is, if $q \rightarrow_a q'$, then an arbitrary number of agents from a certain template may perform the action a and move from state q to state q' . Assume $q \dashrightarrow_a q'$ to denote the latter, but with $a \notin X$. That is, if $q \dashrightarrow_a q'$, then precisely one agent following a certain template may perform the action a and move from state q to state q' . Call a template state q *initialisable* if for every template transition $\hat{\delta}(\hat{a}g, q, a, X) = q'$ we have that $q' \in \hat{q}_0(\hat{a}g)$.

Definition 5 (Resource state) *A template state q is said to be a resource state if: (i) $\nexists(q', a). q' \rightarrow_a q$; (ii) $\exists!(q', a). q' \dashrightarrow_a q$; (iii) q is initialisable.*

Definition 6 (Non-resource state) *A template state q is said to be a non-resource state if $q' \rightarrow_a q$ by means of agent template $\hat{a}g$ implies $q' \dashrightarrow_a q$ by means of $\hat{a}g$, and $q' \dashrightarrow_a q$ by means of an agent template $\hat{a}g$ implies $q' \rightarrow_a q$.*

Intuitively, a template state is a resource state if: by conditions (i) and (ii), exactly one agent (in a concrete system) can take the lock on the shared resource represented by the state at any given time-step; by condition (iii) an agent always releases the lock on a shared resource. Differently, a non-resource state is a template state in which an arbitrary number of agents can move into at any given time step. Clearly, there may be template states that are neither resource states nor non-resource states.

Definition 7 (Mutual exclusion property) *A PCGS is said to satisfy the mutual exclusion property if the following conditions hold: (i) every state is either a resource state or a non-resource state; (ii) every initial template state is a non-resource state.*

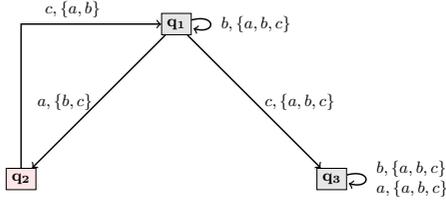


Figure 2: A PCGS violating the starvation freedom property. For clarity several transitions to non-resource states required by the mutual exclusion property are omitted in the figure.

A PCGS satisfying the mutual exclusion property may violate the so called *starvation freedom* property [21]. More specifically, the number of times an agent can access a shared resource may depend on the number of agents present in a concrete system. For example, consider the PCGS of one agent template $\hat{1}$ and zero concrete agents depicted in Figure 2. In a concrete system all agents are initially in state q_1 . Assume that the concrete agent ag wants to access the shared resource represented by the resource state q_2 . The agent has to perform the action a , and all other agents have to collectively perform the actions b and c . So, the agent ag goes to state q_2 and all the other agents move to states q_1 and q_3 . Note that the agents in q_3 remain forever in q_3 . To release the lock on the shared resource, the agent ag has to perform the action c and all the other agents have to collectively perform the actions a, b . Then, ag can take again the lock on the shared resource as described above. However, each time it does so it causes other agents to move from q_1 to q_3 . Thus, eventually, no agents are left in q_1 . At this point, ag cannot move from q_1 to q_2 since no agent can perform the action c . Hence, ag can access the shared resource only a finite number of times, the precise number depending on the number of agents in a concrete system.

The above behaviour is particularly problematic for parameterised model checking techniques. In counter abstraction it generates spurious paths that can be hard to identify [15, 28]. We thus restrict the analysis of PCGS to the class of PCGS that satisfy the mutual exclusion property and the starvation freedom property. The latter is defined as follows. Call a template transition $\hat{\delta}(\hat{i}, q, a, A) = q'$ bad on action b if $q \neq q'$, $b \in A$, and $\hat{\delta}(\hat{i}, q, a, A \setminus \{b\}) \neq q'$. In other words, a template transition from a state q to a different state is bad on an action b if a different action cannot be performed at the state without b being performed at the state. For example, the transition $\hat{\delta}(\hat{1}, q_1, a, \{b, c\}, q_2)$ of the PCGS shown in Figure 2 is bad on actions b and c . The starvation freedom property insists on these actions to be reflexive on the state they are performed, thereby “eliminating” the behaviour described above.

Definition 8 (Starvation freedom property) A PCGS is said to satisfy the starvation freedom property if the following condition holds: if a transition $\hat{\delta}(\hat{i}, q, a, A) = q'$ is bad on an action b and $\hat{\delta}(\hat{j}, r, b, A) = r'$, for any \hat{j} , r , then $r = r'$.

Definition 9 (MIE class of PCGS) The MIE class of PCGS is the class of all PCGS that satisfy the mutual exclusion property and the starvation freedom property.

Intuitively, the MIE class describes resource allocation protocols in which every agent can eventually access a shared resource after requesting to do so. The importance of the MIE class of systems is

particularly significant and goes well beyond these protocols. Indeed, in the next section we show that the PMCP for MIE MAS against PATL specifications is decidable.

4 Verifying the MIE Class of PCGS

In this section we show the significance of the MIE class of PCGS identified above. Specifically we give a procedure that solve the PMCP for MIE PCGS against PATL formulas. Given an MIE PCGS and an \bar{m} -indexed formula, the method involves checking a particular abstract model $\mathcal{S}(\bar{m})$ against the formula. We show below that the abstract model $\mathcal{S}(\bar{m})$ finitely encodes all concrete instantiations of a given PCGS. It follows that the satisfaction of the formula on the abstract model implies the satisfaction of the formula on every concrete instantiation of the original PCGS. Conversely, if the specification is not satisfied on the abstract model $\mathcal{S}(\bar{m})$, there exists a concrete instantiation of the original PCGS that falsifies the formula.

Fix a PCGS $\hat{S} = \langle \hat{\Sigma}, \hat{Q}, \hat{q}_0, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{a}, \hat{\delta} \rangle \in \text{MIE}$ of z concrete agents $\Sigma_C = \{1, \dots, z\}$ and k agent templates $\Sigma_T = \{\hat{1}, \dots, \hat{k}\}$. Let φ be an \bar{m} -indexed PATL formula. For $\bar{x} \in \mathbb{N}^k$, let $\Sigma_T(\bar{x})$ denote the set $\Sigma_T(\bar{x}) = \{(i, j) : 1 \leq i \leq k, 1 \leq j \leq \bar{x}.i\}$ of concrete agents. We now construct the abstract model $\mathcal{S}(\bar{m})$. An abstract state

$$\gamma = ((q_{(1,1)}, \dots, q_{(k, \bar{m}.k)}), q_1, \dots, q_z, (X_1, \dots, X_k))$$

consists of a concrete component $\gamma.c = (q_{(1,1)}, \dots, q_{(k, \bar{m}.k)}, q_1, \dots, q_z)$ and an abstract component $\gamma.ab = (X_1, \dots, X_k)$. The abstract state γ represents any concrete state g in an arbitrarily large system $S(\bar{n})$ in which: each concrete agent $(i, j) \in \Sigma_T(\bar{m})$ is at local state $q_{(i,j)}$; each concrete agent $i \in \Sigma_C$ is at local state q_i ; for each agent template \hat{i} , $X_i = \{g.(i, j) : (i, j) \in \Sigma_T(\bar{n}) \setminus \Sigma_T(\bar{m})\}$ is the set of states for the agents not in $\Sigma_T(\bar{m})$. We write $\gamma.c.ag$ for the local state of the agent ag in γ . By $\gamma.ab.i$ we denote the abstract component associated with the agent template \hat{i} in γ .

Definition 10 (Abstract model) Given a PCGS \hat{S} of k agent templates and z concrete agents, and an \bar{m} -indexed PATL specification φ , the abstract model $\mathcal{S}(\bar{m})$ is defined as the tuple $S = \langle \mathcal{G}, \mathcal{I}, \mathcal{T}, \mathcal{V} \rangle$, where:

- $\mathcal{G} \subseteq \hat{Q}^{|\Sigma_T(\bar{m})|} \times \hat{Q}^z \times (2^{\hat{Q}})^k$ is the set of abstract states.
- $\mathcal{I} = (\hat{q}_0(\hat{1}))^{\bar{m}.1} \times (\hat{q}_0(\hat{k}))^{\bar{m}.k} \times \hat{q}_0(1) \times \dots \times \hat{q}_0(z) \times 2^{\hat{q}_0(\hat{1})} \times \dots \times 2^{\hat{q}_0(\hat{k})}$ is the set of initial abstract states.
- $\mathcal{T} \subseteq \mathcal{G} \times \Lambda^k \times \mathcal{G}$ is the abstract transition relation, where $\Lambda = 2^{\hat{Q} \times \hat{A} \times 2^{\hat{A}} \times \hat{Q}}$ is a set of transition labels.
- $\mathcal{V} : \mathcal{G} \rightarrow 2^{(\hat{\Pi} \times \Sigma_C) \cup (\hat{\Pi} \times \Sigma_T(\bar{m}))}$ is the labelling function defined by $(p, ag) \in \mathcal{V}(\gamma)$ iff $p \in \hat{\pi}(\gamma.c.ag)$.

The concrete component $\gamma.c$ encodes the atomic propositions from which φ is built, whereas the abstract component $\gamma.ab$ encodes the ways an arbitrary number of agents may interfere with the state of $\gamma.c$. We make this precise with the definition of the abstract transition relation. Transitions from an abstract state represent transitions enabled from any concrete state represented by said abstract state. Each abstract transition is labelled by the template transitions taking place in a concrete representative transition. Specifically, for an abstract transition $(\gamma, (\Xi_1, \dots, \Xi_k), \gamma')$, a label $(q, a, A, q') \in \Xi_i$ indicates that in a global transition at least one agent in $\Sigma_T(\bar{n}) \setminus \Sigma_T(\bar{m})$ is in state q ; the agent performs the action a and moves to the state q' , and all other agents perform the set of actions A .

⁴ Note that the undecidability result in [19] is drawn under the assumption that for each $q \in S$, $a \in \Lambda$, there is exactly one state q' with $(q, a, q') \in R$.

Let $\Xi = \Xi_1 \cup \dots \cup \Xi_k$. We write $\text{Actions}(\Xi) = \{a : \exists q, q', A. (q, a, A, q') \in \Xi\}$ for the set of actions performed by Ξ . We now define the abstract transition relation.

Definition 11 (Abstract transition relation) *The abstract transition relation \mathcal{T} is defined as $(\gamma, (\Xi_1, \dots, \Xi_k), \gamma') \in \mathcal{T}$ iff there is a joint action $\bar{a} \in \hat{A}^{|\Sigma_T(\bar{m})|+|\Sigma_C|}$ such that:*

- for every concrete agent $ag = (i, j) \in \Sigma_T(\bar{m})$, we have $\bar{a}.ag \in \hat{d}(\hat{i}, \gamma.c.ag)$ and $\hat{\delta}(\hat{i}, \gamma.c.ag, \bar{a}.ag, \text{Actions}_{-ag}(\bar{a}) \cup \text{Actions}(\Xi)) = \gamma'.c.ag$.
- for every concrete agent $i \in \Sigma_C$, we have $\bar{a}.i \in \hat{d}(\hat{i}, \gamma.c.i)$ and $\hat{\delta}(\hat{i}, \gamma.c.i, \bar{a}.i, \text{Actions}_{-i}(\bar{a}) \cup \text{Actions}(\Xi)) = \gamma'.c.i$.
- for each agent template \hat{i} , for all $l = (q, a, A, q') \in \Xi_i$, we have $q \in \gamma.ab.i$, $q' \in \gamma'.ab.i$, $a \in \hat{d}(\hat{i}, q)$, $\hat{\delta}(\hat{i}, q, a, A) = q'$, and either $A = \text{Actions}(\bar{a})$ or $A = \text{Actions}(\bar{a}) \cup \text{Actions}(\Xi \setminus \{l\})$;
- for each agent template \hat{i} , for every $q \in \gamma.ab.i$, there is a transition label in Ξ_i to a state $q' \in \gamma'.ab.i$, and for every $q' \in \gamma'.ab.i$, there is a transition label in Ξ_i from a state $q \in \gamma.ab.i$.

A path in $\mathcal{S}(\bar{m})$ is a sequence $\gamma_0\gamma_1\dots$ such that for every $i \geq 0$ there is a k -tuple of transition labels (Ξ_1, \dots, Ξ_k) with $(\gamma_i, (\Xi_1, \dots, \Xi_k), \gamma_{i+1}) \in \mathcal{T}$.

The satisfaction of an \bar{m} -indexed formula φ on $\mathcal{S}(\bar{m})$ implies the satisfaction of φ on every concrete system $S(\bar{n})$ with $\bar{n} \geq \bar{m}$.

Lemma 1 *Let $\hat{S} \in \mathbb{MIE}$ be a PCGS of k agent templates and z concrete agents. Let φ be an \bar{m} -indexed formula. Then, $\mathcal{S}(\bar{m}) \models \varphi$ implies $S(\bar{n}) \models \varphi$ for every \bar{n} with $\bar{n} \geq \bar{m}$.*

Proof sketch. Let $\bar{n} \geq \bar{m}$. Define a mapping ζ from the set of concrete states in $S(\bar{n})$ to the set of abstract states in $\mathcal{S}(\bar{m})$:

$$\zeta(q) = (q.(1, 1), \dots, q.(k, \bar{m}.k), q.1, \dots, q.z, \Xi_1, \dots, \Xi_k),$$

where $\Xi_1 = \{q.(1, j) : \bar{m}.1 + 1 \leq j \leq \bar{n}.1\}, \dots, \Xi_k = \{q.(k, j) : \bar{m}.k + 1 \leq j \leq \bar{n}.k\}$. For each agent template \hat{i} , define a mapping ξ_i from concrete transitions to abstract transition labels as follows: for a concrete transition $\delta(q, \bar{a}) = q'$, $(r, a, A, r') \in \xi_i(q, \bar{a}, q')$ iff there is a j with $\bar{m}.i + 1 \leq j \leq \bar{n}.i$ such that $q.(i, j) = r$, $\bar{a}.(i, j) = a$, $A = \text{Actions}_{-(i, j)}(\bar{a})$, and $q'.(i, j) = r'$.

We show that if $\zeta(q) = \gamma$ and $\gamma \models \varphi$, for an \bar{m} -indexed formula φ , then $q \models \varphi$. We do this by induction on φ .

Assume $(p, ag) \in (\hat{\Pi} \times \Sigma_C) \cup (\hat{\Pi} \times \Sigma_T(\bar{m}))$ and $\gamma \models (p, ag)$. Then, $(p, ag) \in \mathcal{V}(\gamma)$. Therefore, $(p, ag) \in \hat{\pi}(\gamma.c.ag)$, and therefore $(p, ag) \in \pi(q)$. Hence, $q \models p$.

The cases of $\varphi = \neg\varphi_1$, and $\varphi_1 \wedge \varphi_2$ are straightforward.

Suppose that $\varphi = \langle\langle \Gamma \rangle\rangle(\varphi_1 U \varphi_2)$ for a group of agents $\Gamma \subseteq \Sigma_C \cup \Sigma_T(\bar{m})$. Let $\gamma \models \varphi$. Consider a strategy $F_\Gamma = \{f_{ag} : ag \in \Gamma\}$ such that for all $\sigma \in \text{out}(\gamma, F_\Gamma)$, there is $i \geq 0$ with $\sigma^i \models \varphi_2$ and $\sigma^j \models \varphi_1$ for all $0 \leq j < i$. Choose the same strategy F_Γ for the group Γ of agents in $S(\bar{n})$. Let $\lambda = q_0q_1\dots \in \text{out}(q, F_\Gamma)$, where $q_0 = q$. Assume a sequence $\bar{a}_0\bar{a}_1\dots$ of joint actions enabling the path λ ; i.e., for each $i \geq 0$, $\delta(q_i, \bar{a}_i) = q_{i+1}$. It can be checked that for each $i \geq 0$, $(\delta(q_i), (\xi_1(q_i, \bar{a}_i, q_{i+1}), \dots, \xi_k(q_i, \bar{a}_i, q_{i+1})), q_{i+1}) \in \mathcal{T}$. Therefore, $\delta(\lambda) = \delta(q_0)\delta(q_1)\dots$ is a path in $\mathcal{S}(\bar{m})$. Moreover, since $\lambda \in \text{out}(F_\Gamma, q)$, it follows that $\delta(\lambda) \in \text{out}(F_\Gamma, \gamma)$. Thus, there is $i \geq 0$ with $\delta(\lambda)^i \models \varphi_2$ and $\delta(\lambda)^j \models \varphi_1$ for all $0 \leq j < i$. By the inductive hypothesis, $\lambda^i \models \varphi_2$ and $\lambda^j \models \varphi_1$ for all $0 \leq j < i$. Hence, $q \models \varphi$.

The case of $\varphi = \langle\langle \Gamma \rangle\rangle G\varphi_1$ can be shown similarly to the case of $\varphi = \langle\langle \Gamma \rangle\rangle(\varphi_1 U \varphi_2)$.

Therefore, we have shown that $\delta(q) = \gamma$ and $\gamma \models \varphi$ implies that $q \models \varphi$. As for every initial state q in $S(\bar{n})$, $\delta(q)$ is an initial state in $\mathcal{S}(\bar{m})$, it follows that $\mathcal{S}(\bar{m}) \models \varphi$ implies $S(\bar{n}) \models \varphi$. ■

Conversely, the falsification of an \bar{m} -indexed formula on $\mathcal{S}(\bar{m})$ implies the existence of a concrete system falsifying the formula.

Lemma 2 *Let $\hat{S} \in \mathbb{MIE}$ be a PCGS of k agent templates and z concrete agents. Let φ be an \bar{m} -indexed formula. Then, $\mathcal{S}(\bar{m}) \not\models \varphi$ implies that there is $\bar{n} \geq \bar{m}$ with $S(\bar{n}) \not\models \varphi$.*

Proof sketch. Given an agent template \hat{i} , let $\hat{\delta}_i = \{(q, a, A, q) : \hat{\delta}(\hat{i}, q, a, A) = q\}$. For a tuple $t = (q, a, A, q)$, $t \in \hat{\delta}_i$, let λ_t be a finite concrete path such that there is an agent in state q in the last state of the path. Denote the concrete system in which λ_t occurs by $S(\bar{n}_t)$; denote the agent that is in state q in λ_t by ag_t . Let $\bar{n} = \bar{m} + \sum_{t \in \hat{\delta}_1 \cup \dots \cup \hat{\delta}_k} \bar{n}_t$, where $\bar{n}_{t1} + \bar{n}_{t2}$ denotes $(\bar{n}_{t1}.1 + \bar{n}_{t2}.1, \dots, \bar{n}_{t1}.k + \bar{n}_{t2}.k)$. Given an agent template \hat{i} , let $\bigcup_t Ag_{i,t}$ be a partition of the set $\{(i, \bar{m}.i + 1), \dots, (i, \bar{n}.i)\}$ of agents in $S(\bar{n})$ such that $|Ag_{i,t}| = \bar{n}_t.i$ for each $t \in \hat{\delta}_1 \cup \dots \cup \hat{\delta}_k$. Assume a bijective mapping $\zeta_{i,t} : Ag_{i,t} \rightarrow \{(i, 1), \dots, (i, \bar{n}_t.k)\}$ from the set $Ag_{i,t}$ of agents in $S(\bar{n})$ to the set $\{(i, 1), \dots, (i, \bar{n}_t.k)\}$ of agents in $S(\bar{n}_t)$. Define a relation R between the states in $\mathcal{S}(\bar{m})$ and the states in $S(\bar{n})$ as follows: $(\gamma, q) \in R$ if: (i) $\gamma.c.(i, j) = q.(i, j)$ for each $(i, j) \in \Sigma_C \cup \Sigma_T(\bar{m})$; (ii) for each \hat{i} , t , every agent $ag \in Ag_{i,t}$ is in local state equal to the local state of the agent $\zeta_{i,t}(ag)$ in the last state of λ_t .

We show that if $(\gamma, q) \in R$ and $\gamma \not\models \varphi$, for an \bar{m} -indexed formula φ , then $q \not\models \varphi$. We do this by induction on φ . The atomic case and the cases of $\varphi = \neg\varphi_1$, and $\varphi_1 \wedge \varphi_2$ are straightforward. Suppose that $\varphi = \langle\langle \Gamma \rangle\rangle(\varphi_1 U \varphi_2)$ for a group of agents $\Gamma \subseteq \Sigma_C \cup \Sigma_T(\bar{m})$. Let $\gamma \not\models \varphi$. Then, for every strategy $F_\Gamma = \{f_{ag} : ag \in \Gamma\}$, there is $\sigma \in \text{out}(\gamma, F_\Gamma)$ that falsifies $\varphi_1 U \varphi_2$. Choose a strategy F_Γ and a path $\sigma = \gamma_0\gamma_1\dots \in \text{out}(\gamma_0, F_\Gamma)$, where $\gamma_0 = \gamma$, falsifying $\varphi_1 U \varphi_2$. Choose the same strategy F_Γ for the group Γ of agents in $S(\bar{n})$. We show that there is a path $\lambda = q_0q_1\dots$, where $q_0 = q$, in $\text{out}(F_\Gamma, q)$ such that $(\gamma_i, q_i) \in R$, for every $i \geq 0$. For each $i \geq 0$, consider $(\gamma_i, (\Xi_1^i, \dots, \Xi_k^i), \gamma_{i+1}) \in \mathcal{T}$ by means of the joint action \bar{a}_i for the agents in $\Sigma_C \cup \Sigma_T(\bar{m})$. Then, λ results from the following joint actions at every time step i :

- Every agent ag in $\Sigma_C \cup \Sigma_T(\bar{m})$ performs the action $\bar{a}_i.ag$.
- Let $\Xi = \Xi_1^i \cup \dots \cup \Xi_k^i$ and $A = \text{Actions}(\Xi) \cup \text{Actions}(\bar{a}_i)$. Consider $\text{Bad} = \{a : \exists t \in \Xi. t \text{ is bad on } a\}$. Then, for each $a \in \text{Bad}$, choose a transition label $t = (q, a, A, q) \in \hat{\delta}_i$ (since $a \in \text{Bad}$, by the starvation freedom property, such a transition label exists for a state q and an agent template \hat{i}), and let the agent $\zeta_{i,t}^{-1}(ag_t)$ perform the action a .
- All the rest of the agents perform the *serial* action.

It can be checked the λ is as required. It follows that λ falsifies $\varphi_1 U \varphi_2$. Thus, $q \not\models \varphi$. The case of $\varphi = \langle\langle \Gamma \rangle\rangle G\varphi_1$ can be shown similarly to the case of $\varphi = \langle\langle \Gamma \rangle\rangle(\varphi_1 U \varphi_2)$.

Therefore, we have shown that $(\gamma, q) \in R$ and $\gamma \not\models \varphi$ implies $q \not\models \varphi$. Now for every initial state γ in $\mathcal{S}(\bar{m})$, there is an initial state q in $S(\bar{n})$ where: $\gamma.c.(i, j) = q.(i, j)$ for each $(i, j) \in \Sigma_C \cup \Sigma_T(\bar{m})$; for each \hat{i} , t , every agent $ag \in Ag_{i,t}$ is in local state equal to the local state of the agent $\zeta_{i,t}(ag)$ in the initial state of λ_t . From q there is a path to a state q' such that $(\gamma, q') \in R$. This path is defined as $\circ_{t \in \hat{\delta}_1 \cup \dots \cup \hat{\delta}_k} \lambda'_t$, where \circ denotes string concatenation, and λ'_t is the path in which: for every agent template \hat{i} , each agent ag in $Ag_{i,t}$

performs the sequences of actions that the agent $\zeta_{i,t}(ag)$ performs in $S(\bar{n}_t)$; every other agent performs the *serial* action. Consequently, as the agents in $\Sigma_C \cup \Sigma_T(\bar{m})$ stutter at their initial states in the path from q to q' , $S(\bar{m}) \not\models \varphi$ implies $S(\bar{n}) \not\models \varphi$. ■

Theorem 2 Let $\hat{S} \in \mathbb{MIE}$ be a PCGS and φ an \bar{m} -indexed formula. Then, the PMCP returns true iff $S(\bar{m}) \models \varphi$.

Proof (\Rightarrow) If $S(\bar{m}) \not\models \varphi$, then, from Lemma 2, the PMCP returns false, which is a contradiction. (\Leftarrow) From Lemma 1.

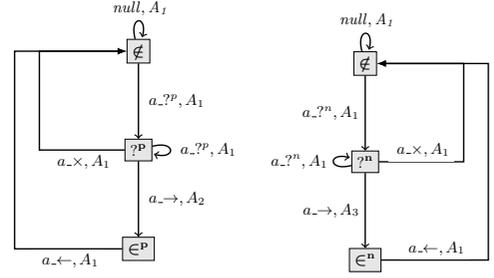
Theorem 2 is the main result of the paper. It states that, for a given \bar{m} -indexed PATL formula φ , the abstract model from Definition 10 is powerful enough to capture every possible instantiation of the PCGS that it is defined from. Since Definition 10 is entirely constructive, this gives us an immediate methodology for verifying any \mathbb{MIE} PCGS against any PATL specification: we simply construct the abstract model following Definition 10 and verify it against the specification φ . The result of this check is the result of the PMCP for the given PCGS against the PATL φ .

5 Prioritised Train-Gate-Controller

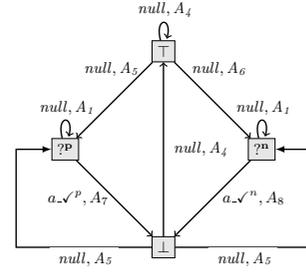
We exemplify the use of the methodology introduced in this paper on the prioritised train-gate controller (PTGC) [30], a parametric variant of the untimed version of the train-gate-controller [4, 20]. The system of PTGC is composed of a controller and an arbitrary number of two types of trains: *prioritised trains* and *normal trains*. Each train runs along a circular track and all tracks pass through a narrow tunnel. The tunnel can accommodate only one train to be in it at any time. Both sides of the tunnel are equipped with traffic lights, which can be either green or red. The controller operates the colour of the traffic lights to let the trains enter and exit the tunnel while complying with the following protocol: a normal train is given permission to enter the tunnel only when there is no pending request from a prioritised train to enter the tunnel.

We now model the PTGC as a PCGS $\mathfrak{G} = \langle \hat{\Sigma}, \hat{Q}, \hat{q}_0, \hat{A}, \hat{\Pi}, \hat{\pi}, \hat{d}, \hat{\delta} \rangle$, where (see Figure 3):

- $\Sigma_C = \{1\}$ and $\Sigma_T = \{\hat{1}, \hat{2}\}$. Template agent $\hat{1}$ represents the prioritised trains, template agent $\hat{2}$ represents the normal trains, and concrete agent 1 represents the controller.
- $\hat{Q} = \{\hat{\zeta}, ?^p, ?^n, \in^p, \in^n, \perp, \top\}$, where:
 - $\hat{\zeta}$ represents that a train is not in the tunnel and it has not requested to enter the tunnel.
 - $?^p$ ($?^n$, respectively) represents that a prioritised (normal, respectively) train has requested to enter the tunnel.
 - \in^p (\in^n , respectively) represents that a prioritised (normal, respectively) train is in the tunnel.
 - \perp represents the traffic lights having colour red.
 - \top represents the traffic lights having colour green.
- $\hat{q}_0 = \{\hat{1} \mapsto \{\hat{\zeta}\}, \hat{2} \mapsto \{\hat{\zeta}\}, 1 \mapsto \{\top\}\}$.
- $\hat{A} = \{null, a_{-?^p}, a_{-?^n}, a_{-\sqrt{p}}, a_{-\sqrt{n}}, a_{-\times}, a_{-\rightarrow}, a_{-\leftarrow}\}$, where:
 - $a_{-?^p}$ ($a_{-?^n}$, respectively) represents a prioritised (normal, respectively) train making a request to enter the tunnel.
 - $a_{-\times}$ represents a train relinquishing its request to enter the tunnel.



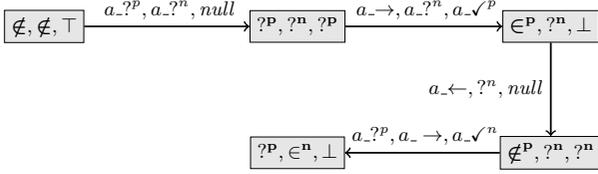
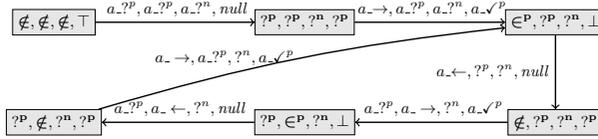
(a) Template prioritised train. (b) Template normal train.



(c) Controller.

Figure 3: The PCGS of the PTGC.

- $a_{-\sqrt{p}}$ ($a_{-\sqrt{n}}$, respectively) represents the controller accepting a prioritised (normal, respectively) train's request to enter the tunnel.
- $a_{-\rightarrow}$ ($a_{-\leftarrow}$, respectively) represents a train entering (exiting, respectively) the tunnel.
- $\hat{\Pi} = \{tp, np\}$, $\hat{\pi}(\in^p) = \{tp\}$, and $\hat{\pi}(\in^n) = \{tn\}$. A prioritised (normal, respectively) train has entered the tunnel.
- $\hat{d}(\hat{1}, \hat{\zeta}) = \{null, a_{-?^p}\}$. Whenever a prioritised train is not in the tunnel, it can choose to do nothing or it can make a request to enter the tunnel.
- $\hat{d}(\hat{1}, ?^p) = \{a_{-?^p}, a_{-\times}, a_{-\rightarrow}\}$. The train has already made a request to enter the tunnel but the request has not yet been granted. The train can relinquish its request, or make a new request, or, if the controller grants access to the tunnel, enter the tunnel.
- $\hat{d}(\hat{1}, \in^p) = \{a_{-\leftarrow}\}$. Whenever a train is in the tunnel, it can only exit the tunnel.
- \hat{d} is similarly defined for normal trains.
- $\hat{d}(1, \top) = \hat{d}(1, \perp) = \{null\}$.
- $\hat{d}(1, ?^p) = \{null, a_{-\sqrt{p}}\}$, $\hat{d}(1, ?^n) = \{null, a_{-\sqrt{n}}\}$. The controller can either delay a request from a train to enter the tunnel, or it can accept it.
- $\hat{\delta}(\hat{1}, \hat{\zeta}, null, A_1) = \hat{\zeta}$ and $\hat{\delta}(\hat{1}, \hat{\zeta}, a_{-?^p}, A_1) = ?^p$, where $A_1 \subseteq \hat{A}$. A prioritised train may choose to remain out of the tunnel or it may choose to request to enter the tunnel irrespectively of the other agents' actions. Similarly, $\hat{\delta}(\hat{1}, ?^p, a_{-?^p}, A_1) = ?^p$, $\hat{\delta}(\hat{1}, ?^p, a_{-\times}, A_1) = \hat{\zeta}$, and $\hat{\delta}(\hat{1}, \in^p, a_{-\leftarrow}, A_1) = \hat{\zeta}$. These transitions are similarly defined for normal trains.
- $\hat{\delta}(\hat{1}, ?^p, a_{-\rightarrow}, A_2) = \in^p$, where $\{a_{-\sqrt{p}}\} \subseteq A_2 \subseteq \hat{A} \setminus \{a_{-\sqrt{n}}\}$. A prioritised train can enter the tunnel if the controller accepts a prioritised request and no other train is entering the tunnel

Figure 4: Fragment of the CCGS $\mathfrak{G}((1, 1))$.Figure 5: Fragment of the CCGS $\mathfrak{G}((2, 1))$.

- at the same time. Similarly, $\hat{\delta}(\hat{2}, ?^n, a_- \rightarrow, A_3) = \in^n$, where $\{a_- \sqrt^n\} \subseteq A_3 \subseteq \hat{A} \setminus \{a_- \sqrt^p\}$.
- $\hat{\delta}(1, \top, null, A_4) = \top$, where $A_4 \subseteq \hat{A} \setminus \{a_- \sqrt^p, a_- \sqrt^n\}$.
 - $\hat{\delta}(1, \top, null, A_5) = ?^p$, where $\{a_- \sqrt^p\} \subseteq A_5 \subseteq \hat{A}$. If both prioritised and normal trains have requested to enter the tunnel, then the controller can only accept prioritised requests. Otherwise, if only normal requests have been made, then the controller can accept normal requests: $\hat{\delta}(1, \top, null, A_6) = ?^n$, where $\{a_- \sqrt^n\} \subseteq A_6 \subseteq \hat{A} \setminus \{a_- \sqrt^p\}$.
 - $\hat{\delta}(1, ?^p, null, A_1) = ?^p$, $\hat{\delta}(1, ?^n, null, A_1) = ?^n$. The controller can delay handling a request irrespectively of the other agents' actions.
 - $\hat{\delta}(1, ?^p, a_- \sqrt^p, A_7) = \perp$ and $\hat{\delta}(1, ?^n, a_- \sqrt^n, A_7) = \perp$, where $\{a_- \rightarrow\} \subseteq A_7 \subseteq \hat{A}$.
 - $\hat{\delta}(1, \perp, null, A_4) = \top$, $\hat{\delta}(1, \perp, null, A_5) = ?^p$, and $\hat{\delta}(1, \perp, null, A_6) = ?^n$.

Any concrete system is generated by considering copies of the templates above. For example, a fragment of the concrete system of one prioritised train and one normal train is depicted in Figure 4; a fragment of the concrete system of two prioritised trains and one normal train is shown in Figure 5. In the figures each concrete state represents, from left to right, the local states of the prioritised trains, the local state of the normal train, and the local state of the controller. The tuples of actions are similarly read.

We now exemplify the use of PATL to reason about MAS with arbitrary many agents. We refer to [3, 4, 20] for several specifications of interest in plain ATL concerning the train-gate-controller. Differently from these works, where the overall system is composed of two trains, we can here represent specifications concerning the strategic properties of an unbounded number of trains. In particular, we are interested in formulating whether exactly one prioritised train can ensure that no normal trains can enter the tunnel irrespectively of how many other trains (normal or prioritised) compose the system. This is expressed by the PATL formula

$$\varphi_{PTGC1} = \forall v.1\forall u.2\langle\{v\}\rangle G\neg(tn, u),$$

where $v.i$ indicates that the variable v is in VAR_i . Further, we would like to check whether at least two prioritised trains have a joint strategy to enforce the above property. This is expressed by the PATL formula

$$\varphi_{PTGC2} = \forall v.1\forall u.1\forall z.2\langle\{v, u\}\rangle G\neg(tn, z).$$

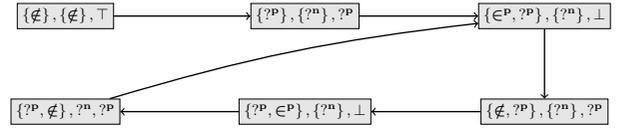


Figure 6: Fragment of the abstract model for the PTGC.

Finally, we would like to assess whether the controller has a strategy to ensure that precisely one train can be in the tunnel at any time, as expressed by the following PATL formula:

$$\begin{aligned} \varphi_{PTGC3} = & \forall v.1\forall u.1\forall z.2\forall w.2G\langle\{1\}\rangle \\ & ((tp, v) \rightarrow (\neg(tp, u) \wedge \neg(tn, z))) \wedge \\ & (tn, z) \rightarrow (\neg(tp, u) \wedge \neg(tn, w)) \end{aligned}$$

It is easy to check that \mathfrak{G} satisfies the mutual exclusion property and the starvation freedom property. Therefore, $\mathfrak{G} \in \mathbb{MEE}$, and therefore we can assess the correctness of the PTGC against the formulae φ_{PTGCi} , $1 \leq i \leq 3$, as per Theorem 2. A fragment of the abstract model that represents the fragment of the concrete system $\mathfrak{G}((2, 1))$ (Figure 5) is shown in Figure 6. A state in the figure depicts, from left to right, the abstract component of the prioritised train, the abstract component of the normal train, and the local state of the controller. For brevity, the concrete components, of both the trains, and the transition labels are omitted in the figure.

The abstract model and the specifications can be checked by an ATL model checker; this would return *false* for φ_{PTGC1} , and *true* for φ_{PTGC2} and φ_{PTGC3} . Indeed, as it can be observed in Figures 5 and 6, any pair of prioritised trains can ensure that no normal trains can enter the tunnel: they can simply request access to the tunnel; since they are prioritised over normal trains, one of them will enter the tunnel while the other continues on requesting access to the tunnel, thus preventing a normal train to enter the tunnel. A single train, however, does not have a strategy to ensure this property, as it can be observed by the counterexample shown in Figure 4.

6 Conclusions

In this paper we put forward a technique for the parameterised verification of MAS against specifications expressing strategic behaviour of the agents. To achieve this we introduced PCGS, a novel parameterised extension of CGS (the usual semantics of ATL); PATL, a parameterised version of ATL; and defined the parameterised model checking problem for MAS given in PCGS against specifications expressed in PATL. The proof of undecidability of the PMCP for PCGS against PATL specifications enabled us to identify an expressive class of PCGS for which a decidable counter abstraction methodology could be given. This enabled us to verify unbounded MAS against strategic specifications, as the example of the prioritised train-gate-controller demonstrated.

In future work we would like to extend the specifications supported by the technique here introduced. For example [26, 27] address parameterised verification against epistemic specifications. It would be of interest to develop a technique that can account for both epistemic and strategic specifications such as those discussed here. However, the counter abstraction technique presented in [27] cannot seemingly be extended to ATL modalities. Moreover, the one here devised for ATL cannot be applied to epistemic modalities. More work is required to solve this problem.

Acknowledgments. This research was funded by the EPSRC under grant EP/I00520X/1 and a Doctoral Prize Fellowship.

REFERENCES

- [1] T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge, 'Knowledge and ability', in *Handbook of Logics for Knowledge and Belief*, College Publications, (2015).
- [2] R. Alur, L. de Alfaro, R. Grosu, T. Henzinger, A. Thomas, M. Kang, C. Kirsch, R. Majumdar, F. Mang, and B.-Y. Wang, 'jMocha: A model checking tool that exploits design structure', in *Proceedings of the 23rd International Conference on Software Engineering (ICSE01)*, pp. 835–836. IEEE, (2001).
- [3] R. Alur, L. de Alfaro, T. Henzinger, S. Krishnan, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran, 'MOCHA user manual', Technical report, University of California at Berkeley, (2000).
- [4] R. Alur, T. A. Henzinger, and O. Kupferman, 'Alternating-time temporal logic', *Journal of the ACM*, **49**(5), 672–713, (2002).
- [5] B. Aminof, S. Jacobs, A. Khalimov, and S. Rubin, 'Parameterized model checking of token-passing systems', in *Proceedings of the 15th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI14)*, volume 8318 of *Lecture Notes in Computer Science*, pp. 262–281. Springer, (2014).
- [6] K.R. Apt and D. C. Kozen, 'Limits for automatic verification of finite-state concurrent systems', *Information Processing Letters*, **22**(6), 307–309, (1986).
- [7] F. Belardinelli, D. Grossi, and A. Lomuscio, 'Finite abstractions for the verification of epistemic properties in open multi-agent systems', in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI15)*, pp. 854–860. AAAI Press, (2015).
- [8] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder, *Decidability of Parameterized Verification*, Morgan and Claypool Publishers, 2015.
- [9] N. Bulling and W. Jamroga, 'Comparing variants of strategic ability: how uncertainty and memory influence general properties of games', *Autonomous Agents and Multi-Agent Systems*, **28**(3), 474–518, (2014).
- [10] K. Chatterjee, T. Henzinger, and N. Piterman, 'Strategy logic', in *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR07)*, volume 4703, pp. 59–73, (2007).
- [11] E. Clarke, M. Talupur, T. Touili, and H. Veith, 'Verification by network decomposition', in *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR04)*, volume 3170 of *Lecture Notes in Computer Science*, 276–291, Springer, (2004).
- [12] E.M. Clarke, O. Grumberg, and M.C. Browne, 'Reasoning about networks with many identical finite state processes', *Information and Computation*, **81**(1), 13–31, (1989).
- [13] E.M. Clarke, M. Talupur, and H. Veith, 'Proving ptolemy right: The environment abstraction framework for model checking concurrent systems', in *Proceedings of 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS08)*, volume 4963 of *Lecture Notes in Computer Science*, pp. 33–47. Springer, (2008).
- [14] C. Dima and F. Tiplea, 'Model-checking ATL under imperfect information and perfect recall semantics is undecidable', *CoRR*, **abs/1102.4225**, (2011).
- [15] E. A. Emerson and K. S. Namjoshi, 'Automatic verification of parameterized synchronous systems', in *Proceedings of the 8th International Conference on Computer Aided Verification (CAV96)*, volume 1102 of *Lecture Notes in Computer Science*, pp. 87–98. Springer, (1996).
- [16] E. A. Emerson and K. S. Namjoshi, 'On model checking for non-deterministic infinite-state systems', in *Proceedings of 13th International Symposium on Logic in Computer Science (LICS98)*, pp. 70–80. IEEE, (1998).
- [17] E.A. Emerson and V. Kahlon, 'Model checking guarded protocols', in *Proceedings of the 14th International Symposium on Logic in Computer Science (LICS03)*, pp. 361–370. IEEE, (2003).
- [18] E.A. Emerson and K.S. Namjoshi, 'Reasoning about rings', in *Proceedings of the 22nd Annual Sigact-Aigplan on Principles of Programming Languages (POPL95)*, pp. 85–94. Pearson Education, (1995).
- [19] J. Esparza, A. Finkel, and R. Mayr, 'On the verification of broadcast protocols', in *Proceedings of the 10th International Symposium on Logic in Computer Science (LICS99)*, pp. 352–359. IEEE, (1999).
- [20] W. van der Hoek and M. Wooldridge, 'Tractable multiagent planning for epistemic goals', in *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS02)*, pp. 1167–1174. ACM Press, (2002).
- [21] M. R. A. Huth and M. D. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems (2nd edition)*, Cambridge University Press, Cambridge, England, 2004.
- [22] W. Jamroga and J. Dix, 'Model checking abilities under incomplete information is indeed δ_p^2 -complete', in *Proceedings of the 4th European Workshop on Multi-Agent Systems EUMAS'06*, pp. 14–15. Citeseer, (2006).
- [23] G. Jonker, *Feasible Strategies in Alternating-time Temporal Epistemic Logic*, Master's thesis, University of Utrecht, The Netherlands, 2003.
- [24] M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Woźna, and A. Zbrzezny, 'Verics 2007 - a model checker for knowledge and real-time', *Fundamenta Informaticae*, **85**(1), 313–328, (2008).
- [25] P. Kouvaros and A. Lomuscio, 'Automatic verification of parametrised interleaved multi-agent systems', in *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS13)*, pp. 861–868. IFAAMAS, (2013).
- [26] P. Kouvaros and A. Lomuscio, 'A cutoff technique for the verification of parameterised interpreted systems with parameterised environments', in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI13)*, pp. 2013–2019. AAAI Press, (2013).
- [27] P. Kouvaros and A. Lomuscio, 'A counter abstraction technique for the verification of robot swarms', in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI15)*, pp. 2081–2088. AAAI Press, (2015).
- [28] P. Kouvaros and A. Lomuscio, 'Verifying emergent properties of swarms', in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI15)*, pp. 1083–1089. AAAI Press, (2015).
- [29] P. Kouvaros and A. Lomuscio, 'Formal verification of opinion formation in swarms', in *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS16)*, pp. 1200–1209. IFAAMAS, (2016).
- [30] P. Kouvaros and A. Lomuscio, 'Parameterised verification for multi-agent systems', *Artificial Intelligence*, **234**, 152–189, (2016).
- [31] A. Lomuscio, H. Qu, and F. Raimondi, 'MCMAS: A model checker for the verification of multi-agent systems', *Software Tools for Technology Transfer*, (2015). <http://dx.doi.org/10.1007/s10009-015-0378-x>.
- [32] F. Mogavero, A. Murano, and L. Sauro, 'On the boundary of behavioral strategies', in *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS2013)*, pp. 263–272. IEEE, (2013).
- [33] Marco Montali, Diego Calvanese, and Giuseppe De Giacomo, 'Verification of data-aware commitment-based multiagent system', in *Proceedings of the 14th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS14)*, pp. 157–164. IFAAMAS, (2014).
- [34] C. Nardini, B. Kozma, and A. Barrat, 'Whos talking first? consensus or lack thereof in coevolving opinion formation models', *Physical review letters*, **100**(15), 158701, (2008).
- [35] A. S. Rao and M. P. Georgeff, 'Decision procedures for BDI logics', *Journal of Logic and Computation*, **8**(3), 293–343, (1998).
- [36] E. Semsar-Kazerooni and K. Khorasani, 'Multi-agent team cooperation: A game theory approach', *Automatica*, **45**(10), 2205–2213, (2009).

Interpretable Encoding of Densities Using Possibilistic Logic

Ondřej Kuželka¹ and Jesse Davis² and Steven Schockaert³

Abstract. Probability density estimation from data is a widely studied problem. Often, the primary goal is to faithfully mimic the underlying empirical density. Having an interpretable model that allows insight into why certain predictions were made is often of secondary importance. Using logic-based formalisms, such as Markov logic, can help with interpretability, but even in Markov logic it can be difficult to gain insight into a model's behavior due to interactions between the logical formulas used to specify the model. This paper explores an alternative approach to representing densities that makes use of possibilistic logic. Concretely, we propose a novel way to transform a learned density tree into a possibilistic logic theory. An advantage of our transformation is that it permits performing both MAP and, surprisingly, marginal inference, with the converted possibilistic logic theory. At the same time, we still retain the benefits conferred by using possibilistic logic, such as the ability to compact the theory and the interpretability of the model.

1 INTRODUCTION

A key machine learning task is learning to compactly represent a probability density from a given set of examples. The resulting distributions can be useful for answering a large variety of queries. While many sophisticated approaches exist for this task [7, 21, 27, 23, 25, 26] usually the focus is on ensuring that the model as accurately as possible captures the empirical distribution. Often, this comes at the expense of interpretability, which makes it difficult to gain insight into a model's predictions or to refine it based on feedback from users or experts.

A natural approach for representing densities in an interpretable way is to describe them using logical formulas (or in a framework which is close to logic such as Bayesian networks). Figure 1(a) shows the idea underlying Markov logic [26], one of the most popular logic-based frameworks for modelling densities. This example approximates the density using the propositional formulas $\alpha_1, \dots, \alpha_{10}$, each of which has a corresponding weight w_i . In Figure 1(a), the height of each box is related to the weight associated with the corresponding formula, and the width represents its set of models. The probability of a given possible world is defined to be proportional to the exponentiated sum of the weights of the satisfied formulas. Still, it can be difficult to grasp the intuitive meaning of the weights associated with the formulas. In the considered example, for instance, α_1 has

one the highest weights, which could incorrectly give the impression that models of α_1 are highly probable.

This paper explores an alternative approach, also based on weighted logical formulas, which is illustrated in Figure 1(b). Here, a weight's meaning is clear as there is a more explicit relationship between it and the probability of the corresponding possible worlds: a formula β with weight $1 - p$ means that its models can have at most a probability of p . Thus, weighted formulas are seen as constraints that act on the set of possible worlds. In Figure 1(b), the most probable worlds are exactly those that satisfy $\neg\beta_1, \dots, \neg\beta_9$. Similarly, if the weight associated with formulas β_5 and β_6 is $1 - p_1$, then the worlds whose probability is higher than p_1 are exactly those who satisfy $\neg\beta_1, \dots, \neg\beta_4, \neg\beta_7, \dots, \neg\beta_9$.

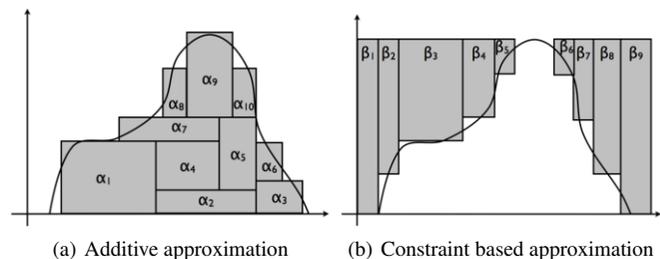


Figure 1. Two ways of approximating a density using logical formulas.

To the best of our knowledge, we present the first such constraint-based representation of probability densities in a logical setting. Specifically, the approach we propose consists of estimating a density tree [25] from a given set of examples, and then converting it into a possibilistic logic theory [8]. Possibilistic logic is a well-known representation and reasoning formalism that supports non-monotonic inferences, which is a requirement if we are to model probability densities. At the same time, it remains close to classical logic, which helps with interpretability and means that off-the-shelf SAT solvers enable highly efficient reasoning. We present several novel ways to perform this transformation. We show that it is possible to perform both MAP and, surprisingly, marginal inference, with the converted possibilistic logic theory. Using possibilistic logic confers several advantages. First, because possibilistic logic remains close to classical logic, we can exploit logical inference to reduce the size of the learned theories, sometimes leading to theories that are exponentially smaller than the corresponding density trees. Second, we can improve the quality of learned possibilistic logic theories by taking into account feedback or input from experts. To this end, we provide an

¹ School of Computer Science and Informatics, Cardiff University, UK, email: KuzelkaO@cardiff.ac.uk

² Department of Computer Science, KU Leuven, Belgium, email: jesse.davis@cs.kuleuven.be

³ School of Computer Science and Informatics, Cardiff University, UK, email: SchockaertS1@cardiff.ac.uk

algorithm for retraining the weights of the possibilistic logic theory that preserves the expert’s modifications. Finally, the resulting theories should be more interpretable.

An implementation of the methods described in this paper is available online⁴.

2 BACKGROUND

2.1 Possibilistic logic

Syntax A theory in possibilistic logic is a set of formulas $\{(\alpha_1, \lambda_1), \dots, (\alpha_n, \lambda_n)\}$ with each α_i a propositional formula and λ_i a certainty weight in $]0, 1]$. The standard inference relation in possibilistic logic follows the principle of the weakest link, i.e. $\{(\alpha_1, \lambda_1), \dots, (\alpha_n, \lambda_n)\} \vdash (\alpha^*, \lambda^*)$ if the following entailment holds in classical logic: $\{\alpha_i \mid \lambda_i \geq \lambda^*\} \models \alpha^*$. In other words, we can derive (α^*, λ^*) iff we can classically derive α^* without using formulas whose certainty weight is strictly less than λ^* .

The λ -cut of a possibilistic logic theory Θ is defined as the classical theory $\Theta^\lambda = \{\gamma \mid (\gamma, \mu) \in \Theta \text{ and } \mu \geq \lambda\}$. A non-monotonic consequence relation \vdash_{poss} can be defined for possibilistic logic as follows. Consider a possibilistic logic theory Θ and formula α . Let λ^* be the highest certainty value for which $\{\alpha\} \cup \Theta^{\lambda^*}$ is inconsistent (and $\lambda^* = 0$ if there is no such certainty value). Then $(\Theta, \alpha) \vdash_{\text{poss}} \beta$ iff the entailment $\{\alpha\} \cup \{\alpha_i \mid (\alpha_i, \lambda_i) \in \Theta, \lambda_i > \lambda^*\} \models \beta$ holds in classical logic. Note that all formulas whose certainty weight is at most λ^* are ignored, even if they are unrelated to any inconsistency in Θ . This is known as the drowning effect.

Semantics The semantics of possibilistic logic are defined in terms of possibility distributions. A possibility distribution, in this context, is a mapping π from the set of possible worlds Ω to $[0, 1]$. A possibility distribution induces two uncertainty measures: the possibility measure Π and its dual N , defined for $A \subseteq \Omega$ as [30, 10]:

$$\Pi(A) = \max_{a \in A} \pi(a) \quad N(A) = 1 - \Pi(\Omega \setminus A)$$

We will also write $N(\alpha)$ as an abbreviation for $N(\llbracket \alpha \rrbracket)$, where α is a propositional formula and $\llbracket \alpha \rrbracket$ is its set of models. Intuitively, $\Pi(\alpha)$ reflects the degree to which α is compatible with our available beliefs, while $N(\alpha)$ reflects the degree to which α is considered certain. At the semantic level, the possibilistic logic theory $\Theta = \{(\alpha_1, \lambda_1), \dots, (\alpha_n, \lambda_n)\}$ corresponds to the possibility distribution π defined by $(\omega \in \Omega)$:

$$\pi(\omega) = 1 - \max\{\lambda_i \mid \omega \not\models \alpha_i\} \quad (1)$$

where we assume $\max \emptyset = 0$. It is easy to see that, for N the necessity measure induced by π , it holds that $N(\alpha_i) \geq \lambda_i$. Moreover, it can be shown that $\Theta \vdash (\alpha, \lambda)$ iff $N(\alpha) \geq \lambda$ [20].

2.2 Density estimation trees

A density estimation tree⁵ is a rooted directed binary tree in which internal nodes are labelled by propositional variables (attributes) and leaves are labelled by real numbers (“densities”). $\text{Nodes}(T)$ denotes the set of the nodes in tree T and $\text{Leaves}(T)$ the set of its leaves. Edges are labelled by 0 (false) or 1 (true). A path from the root

to a leaf is called a branch. We will represent branches by conjunctions. For instance, let $a_1, 0, a_{i_2}, 1, \dots, 1, a_{i_{k-1}}, 0$ be the sequence of labels of internal nodes and edges corresponding to a branch from the root N_1 , labelled with propositional variable a_1 , to a leaf N_{i_k} . Then the conjunction corresponding to this branch is $\neg a_1 \wedge a_{i_2} \wedge \dots \wedge \neg a_{i_{k-1}}$. We call branches paths or conjunctions interchangeably. A density estimation tree defines a probability distribution on possible worlds where the probability of a world ω is given by the label of the leaf of the unique branch B which is consistent with ω (i.e. such that $\omega \models B$). Importantly, density estimation trees can be learned efficiently from data [25].

3 REPRESENTING DENSITY TREES IN POSSIBILISTIC LOGIC

Next, we show how we can transform a density tree into a possibilistic logic theory. Surprisingly, this transformation permits computing marginal probabilities from the possibilistic logic theory.

3.1 Transforming density trees

We start with a basic transformation which is similar in spirit to transforming a decision tree into a CNF formula.

Transformation 1. Let T be a density estimation tree. Let $\mathcal{B} = \{B_1, \dots, B_k\}$ be the set of all branches of the tree, represented as conjunctions, and let p_1, \dots, p_k be the estimated probabilities of worlds consistent with the respective branches, i.e. if $\omega \models B_i$ then $p(\omega) = p_i$. We define the possibilistic logic theory corresponding to T as $\Theta_T = \{(\neg B_i, 1 - p_i) \mid B_i \in \mathcal{B}\}$.

Proposition 1. If T is a density estimation tree and Θ_T is its possibilistic logic theory constructed by Transformation 1, then for all possible worlds ω , it holds that $p(\omega) = \pi(\omega)$, where $p(\cdot)$ is the probability given by T and $\pi(\cdot)$ is the possibility distribution associated with Θ_T .

Proof. Let ω be a possible world, T be a density estimation tree and $\mathcal{B} = \{B_1, \dots, B_k\}$ be the set of all its branches, represented as conjunctions. Let $B^* \in \mathcal{B}$ be a branch consistent with ω and p^* be the respective density, i.e. $\omega \models B^*$. Clearly, there can be only one such branch as all the branches in T are mutually exclusive. Likewise, the only rule $(\alpha', \lambda') \in \Theta$ which is not satisfied in ω is $(\neg B^*, 1 - p^*)$. By (1), we therefore have $\pi(\omega) = 1 - (1 - p^*) = p^*$. It follows that for any $\omega \in \Omega$ we have $\pi(\omega) = p(\omega)$. \square

Remark 1. Transformation 1 works in time $O(|\text{Nodes}(T)|^2)$. A possibilistic logic theory constructed by Transformation 1 from a density estimation tree T has at most $|\text{Leaves}(T)|$ rules and its size, i.e. the sum of the lengths of its rules, is bounded by $|\text{Nodes}(T)|^2$. Figure 2 shows an example of a tree, whose possibilistic logic representation is quadratic. Specifically, the size of the possibilistic logic theory constructed from such a tree of depth k (which has size $S = 2k - 1$ nodes) by Transformation 1 is of order $O(k^2)$ which is also of order $O(S^2)$.

In Section 4.1 we will show how the size of the constructed possibilistic logic theories can be reduced, often significantly. As we will see, there are cases where the reduced possibilistic logic theories are exponentially smaller than the trees from which they were created. Next we give an example of applying Transformation 1.

⁴ <https://github.com/supertweety/>

⁵ In this paper we only consider density estimation trees involving Boolean variables. In general, density estimation trees can also define probability densities in continuous domains.

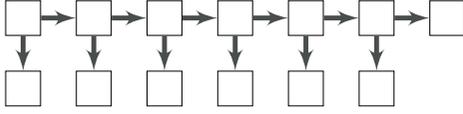


Figure 2. A tree whose possibilistic logic representation is quadratic.

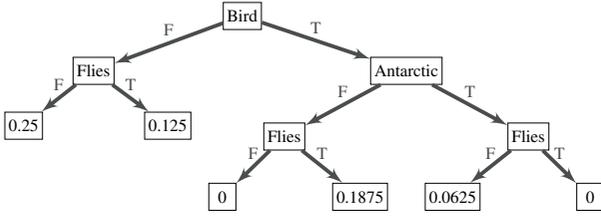


Figure 3. A density estimation tree from Example 1.

Example 1. Applying Transformation 1 to the density estimation tree T in Figure 3 yields the following possibilistic logic theory

$$\Theta = \{(\neg \text{bird} \vee \text{antarctic} \vee \text{flies}, 1), \\ (\neg \text{bird} \vee \neg \text{antarctic} \vee \neg \text{flies}, 1), \\ (\neg \text{bird} \vee \neg \text{antarctic} \vee \text{flies}, 0.9375), \\ (\text{bird} \vee \neg \text{flies}, 0.875), \\ (\neg \text{bird} \vee \text{antarctic} \vee \neg \text{flies}, 0.8125), \\ (\text{bird} \vee \text{flies}, 0.75)\}.$$

Notice that we have, e.g. $(\Theta, \emptyset) \vdash_{\text{poss}} \neg \text{bird}$, $(\Theta, \{\text{bird}\}) \vdash_{\text{poss}} \text{flies} \wedge \neg \text{antarctic}$.

Remark 2. In the possibilistic logic theory obtained by Transformation 1, the formula with the lowest weight always drowns, i.e. it is inconsistent with the other formulas and will thus never play a role in the evaluation of \vdash_{poss} . In many cases, it is therefore possible to remove this formula from the theory. However, if we want to use the possibilistic logic theory for probabilistic inference, as in Section 3.2, then we must keep the lowest level in the theory (although we can replace it by \perp) in order to keep the information about the numerical value of the probability of the most probable worlds.

Transformation 1 differs from the standard probability-possibility transformation [11]. For completeness, we present the syntactical counterpart of the standard probability-possibility transformation in Appendix A.1. In fact, both transformations induce the same ranking of possible worlds (since both are identical to the ranking induced by the probability distribution). Therefore, any classical formula α which can be derived using the possibilistic entailment operator \vdash_{poss} from a theory obtained by one of the transformations can also be derived from the theory obtained by the other transformation.

It is beneficial for scalability to simplify the possibilistic theory as much as possible already while performing the transformation, or at least without having to check logical entailment. To this end, we describe a simple modification of Transformation 1 which results in smaller possibilistic logic theories.

Transformation 2. Let T be a density estimation tree. Let $\mathcal{B} = \{B_1, \dots, B_k\}$ be the set of all branches of the tree, represented

as conjunctions, and let p_1, \dots, p_k be the estimated probabilities of worlds consistent with the respective branches, i.e. if $\omega \models B_i$ then $p(\omega) = p_i$. Let us assume w.l.o.g. that $p_i \geq p_{i+1}$. The resulting possibilistic logic theory then consists of possibilistic logic formulas $(\neg B'_i, 1 - p_i)$ where each B'_i is a conjunction and is obtained as follows:

- Without loss of generality, let $B_i = b_1^i \wedge \dots \wedge b_{j_i}^i$ where its conjuncts are ordered so that the node labeled by b_s^i is closer to the root of T than the node labeled by b_t^i whenever $s < t$.
- Let $B'_i = b_1^i \wedge \dots \wedge b_r^i$ be the minimal prefix of B_i such that there is no $j < i$ such that B_j contains B'_i as a prefix.

Proposition 2. Transformation 1 and Transformation 2 produce equivalent possibilistic logic theories.

Proof. Let us denote by Θ_A the result of Transformation 1 and by Θ_B the result of Transformation 2. To prove this proposition it is enough to show that Θ_A and Θ_B correspond to the same possibility distribution. Let B_1^A, \dots, B_k^A be defined as in Transformation 1 and let B_1^B, \dots, B_k^B be defined as in Transformation 2. Let ω be a possible world and let $\neg B_{i^*}^A$ be the only formula from Θ_A not satisfied in ω (it follows from the proof of Proposition 1 that there is only one such formula in Θ_A). Clearly the respective formula $\neg B_{i^*}^B$ cannot be satisfied in ω (because $\neg B_{i^*}^B$ implies $\neg B_{i^*}^A$). Let $1 - p_{i^*}$ be the weight of $\neg B_{i^*}^A$ in Θ_A (equal to the weight of $\neg B_{i^*}^B$ in Θ_B). What we need to show is that $\neg B_{i^*}^B$ has the highest weight among the formulas from Θ_B which are not satisfied in ω . It follows from the way Transformation 2 works that any such formula would necessarily have to be a prefix of $\neg B_{i^*}^B$ (i.e. a clause consisting of the first r literals of $\neg B_{i^*}^B$). This is because every clause in Θ_B is a prefix of some clause in Θ_A and at most one clause from Θ_A can be falsified in any possible world ω at the same time. But then it follows that $\neg B_{i^*}^B$ must have the highest weight among the falsified formulas because, by construction, there is no clause $\neg B_j^B$ with $j > i^*$ which is a prefix of $\neg B_{i^*}^B$. It follows that $\pi_A(\omega) = \pi_B(\omega)$, where π_A and π_B are the possibility distributions corresponding to Θ_A and Θ_B , respectively. \square

The next example illustrates the use of Transformation 2.

Example 2. Let us consider the same density estimation tree as in Example 1 (shown in Figure 3). The branches of the tree, represented as conjunctions, are: $B_1 = \neg \text{bird} \wedge \neg \text{flies}$, $B_2 = \text{bird} \wedge \neg \text{antarctic} \wedge \text{flies}$, $B_3 = \neg \text{bird} \wedge \text{flies}$, $B_4 = \text{bird} \wedge \text{antarctic} \wedge \neg \text{flies}$, $B_5 = \text{bird} \wedge \text{antarctic} \wedge \text{flies}$, and $B_6 = \text{bird} \wedge \neg \text{antarctic} \wedge \neg \text{flies}$. The corresponding conjunctions B'_i are then (we omit B'_1 considering Remark 2):

$$\begin{aligned} B'_2 &= \text{bird}, & B'_3 &= \neg \text{bird} \wedge \text{flies} \\ B'_4 &= \text{bird} \wedge \text{antarctic}, & B'_5 &= \text{bird} \wedge \text{antarctic} \wedge \text{flies} \\ B'_6 &= \text{bird} \wedge \neg \text{antarctic} \wedge \neg \text{flies} \end{aligned}$$

Applying Transformation 2 yields the following possibilistic logic theory: $\Theta' = \{(\neg \text{bird} \vee \text{antarctic} \vee \text{flies}, 1), (\neg \text{bird} \vee \neg \text{antarctic} \vee \neg \text{flies}, 1), (\neg \text{bird} \vee \neg \text{antarctic}, 0.9375), (\text{bird} \vee \neg \text{flies}, 0.875), (\neg \text{bird}, 0.8125)\}$.

3.2 Answering queries

In this section, we discuss how different types of queries about a given density can be answered by using the possibilistic logic theory obtained from Transformation 1 or 2. The most natural kinds of

queries to consider, in the context of possibilistic logic, are maximum a posteriori (MAP) queries, as these only depend on the ordering of the possible worlds. In particular, we consider the following MAP inference relation [13]:

$$(T, \alpha) \vdash_{MAP} \beta \quad \text{iff} \quad \forall \omega \in \max(T, \alpha) : \omega \models \beta$$

where T is a density tree, α and β are propositional formulas and $\max(T, \alpha)$ is the set of most probable models of α , w.r.t. the probability distribution induced by T . The next proposition shows that, for possibilistic logic theories obtained using the introduced transformations, whatever can be derived using the \vdash_{MAP} relation from the tree can also be derived using \vdash_{poss} from the respective possibilistic logic theory, and vice versa.

Proposition 3. *If T is a density estimation tree and Θ_T is the possibilistic logic theory constructed using Transformation 1 or 2, or using Transformation 4 described in the appendix, then for any formulas α and β it holds that*

$$(T, \alpha) \vdash_{MAP} \beta \quad \text{iff} \quad (\Theta_T, \alpha) \vdash_{poss} \beta.$$

Proof. It is sufficient to show that the ranking of possible worlds induced by the probability $p(\cdot)$ defined by the density estimation tree is the same as the rankings of possible worlds induced by the possibility distributions $\pi(\cdot)$ corresponding with the theories that are obtained by the three transformations. For Transformation 1 this follows from Proposition 1. For Transformation 2, this follows from Proposition 1 and Proposition 2. For Transformation 4, this follows from the so-called order preservation principle, which is known to hold for the probability-possibility transformation (see [12]). \square

One important advantage of Transformation 1 and 2 over the standard probability-possibility transformation is that it permits computing marginal probabilities directly from the possibilistic logic theory. In particular, if Θ_T is a possibilistic logic theory obtained by Transformation 1 or 2 then the probability of a formula α is

$$P(\alpha) = \sum_{\omega: \omega \models \alpha} \pi(\omega). \quad (2)$$

The convenience of the possibilistic logic encoding lies in the fact that the sum in (2) can easily be computed using model counting as follows.

- Let Θ be a possibilistic logic theory obtained by Transformation 1 or 2 and let $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_k\}$ be the set of weights appearing in Θ , sorted in increasing order. Let us set $\lambda_{k+1} = \infty$ (for convenience of notation below). For every $\lambda \in \Lambda$ let us define $M_\lambda^\alpha = \{\omega \mid \omega \models \Theta^\lambda \cup \{\alpha\}\}$, with Θ^λ the λ -cut of Θ as before. In other words, $|M_\lambda^\alpha|$ is the “model count” of $\Theta^\lambda \cup \{\alpha\}$.
- We find:

$$P_{\Theta}(\alpha) = \sum_{i=1}^k (1 - \lambda_i) \cdot (|M_{\lambda_{i+1}}^\alpha| - |M_{\lambda_i}^\alpha|). \quad (3)$$

Proposition 4. *If Θ is a possibilistic logic theory whose corresponding possibility distribution π can be interpreted as a probability distribution (i.e. $\sum_{\omega} \pi(\omega) = 1$). Let P be the probability measure induced by π . It holds that $P(\alpha) = P_{\Theta}(\alpha)$.*

Proof. We need to show that $P_{\Theta}(\alpha) = \sum_{\omega: \omega \models \alpha} \pi(\omega)$. Let Λ and λ_{k+1} be defined as above. We have

$$\begin{aligned} \sum_{\omega: \omega \models \alpha} \pi(\omega) &= \sum_{\omega: \omega \models \alpha} 1 - \max\{\lambda \mid (\gamma, \lambda) \in \Theta \text{ and } \omega \not\models \gamma\} \\ &= \sum_{i=1}^k (1 - \lambda_i) \cdot |\{\omega \in \Omega \mid \omega \models \Theta^{\lambda_{i+1}} \cup \{\alpha\}\}| \\ &= \sum_{i=1}^k (1 - \lambda_i) \cdot (|M_{\lambda_{i+1}}^\alpha| - |M_{\lambda_i}^\alpha|) \end{aligned}$$

where the last equality follows from the fact that $M_{\lambda_i}^\alpha \subseteq M_{\lambda_{i+1}}^\alpha$. \square

If α is just a conjunction of literals, and the possibilistic logic theory Θ is the direct transformation of a density estimation tree, then performing inference in the tree will likely be more straightforward. If the theory Θ has been modified (e.g., refined by an expert) or α is a more complicated formula than a conjunction of literals, then performing inference directly in the possibilistic logic theory may be more efficient.

Using possibilistic logic inference, we can also easily characterize what is true in all worlds whose probability is above a given threshold θ . In particular, it is easy to see that:

$$\llbracket \{\alpha \mid (\alpha, \lambda) \in \Theta, \lambda > 1 - \theta\} \rrbracket = \{\omega \mid p(\omega) \geq \theta\}$$

where p is the probability distribution associated with the density tree T that was used to construct the possibilistic logic theory Θ , using Transformation 1 or 2. It follows that the set of formulas $\{\alpha \mid (\alpha, \lambda) \in \Theta, \lambda > 1 - \theta\}$ exactly characterizes what is true for all worlds whose probability is at least θ . Along similar lines, using model counting as in Eq. 3, we can easily characterize what is true for the $x\%$ most probable worlds, or even the $x\%$ most probable worlds in which some formula α is true.

4 IMPROVING LEARNED THEORIES

Learned possibilistic logic theories may be improved in multiple ways, some of which we describe in this section.

4.1 Pruning

Exact pruning An important advantage of encoding density estimation trees in possibilistic logic is that the resulting theories can be simplified based on logical inference. In particular, a weighted formula (α, λ) can be removed from a theory Θ if $\Theta^\lambda \setminus \{\alpha\} \models \alpha$. We can iteratively identify and remove such formulas, until the theory Θ is free from redundancies. To further simplify the theory, note that each of the proposed transformations results in a weighted set of clauses. Clearly, we can replace the weighted clause $(a_1 \vee \dots \vee a_k, \lambda)$ by the sub-clause $(b_1 \vee \dots \vee b_l, \lambda)$, with $\{b_1, \dots, b_l\} \subset \{a_1, \dots, a_k\}$, if $\Theta^\lambda \models b_1 \vee \dots \vee b_l$. This often results in substantially smaller theories, while yielding the same MAP predictions and probability estimates as the initial density trees.

Example 3. *Consider a density tree that assigns a uniform non-zero probability to worlds satisfying the formula $(a_1 \vee a_2) \wedge (a_3 \vee a_4) \wedge \dots \wedge (a_{n-1} \vee a_n)$, and zero probability to the remaining worlds, then such a density tree will be exponentially larger than the respective possibilistic encoding after pruning, which only contains the formulas $(a_1 \vee a_2, \lambda), \dots, (a_{n-1} \vee a_n, \lambda)$.*

Approximate pruning It is possible to further simplify the possibilistic logic theories if we drop the requirement that the associated possibility distribution should be identical to the probability distribution encoded by the density tree. A particularly convenient way of reducing possibilistic logic theories is to iteratively merge levels with consecutive weights, each time simplifying the newly created level using the exact pruning method outlined above. One possibility is to iteratively merge the levels with the highest weights, which is especially useful for MAP inference, as this reduction only affects the relative ordering of the least probable worlds. Moreover, whatever can be derived from the reduced theory by \vdash_{poss} can also be derived from the original theory, although the converse does not hold in general. For marginal inference, it is necessary to recompute the weight of the new level but that is straightforward.

Pruning default rules If we only care about the ordering of the possible worlds, a possibilistic logic theory Θ may be seen as a compact representation of a default theory, where a default “if α then typically β ” is in this theory if and only if $(\Theta, \{\alpha\}) \vdash_{\text{poss}} \beta$. In some cases, e.g. if we want to explain the theories to people without training in logic, it may be preferable to explain what is captured by a given possibilistic logic theory by presenting these default rules instead. A set of short default rules which are implicitly encoded by the possibilistic logic theory can be extracted using a method described in [19]. The resulting set of defaults is usually too large, however. We describe a practically efficient and theoretically sound method, which we also use in the experiments, for pruning sets of default rules in Appendix A.2.

4.2 Parameter reestimation

Recall that experts can easily modify a possibilistic logic theory. However, manually modifying a theory obtained using either Transformation 1 or 2 would require us to reestimate the theory’s weights if we wanted to use it for computing marginal probabilities (performing MAP inferences does not require retraining the weights). However, we do not want this retraining to override an expert’s modifications. Therefore, we require that the reestimated weights have the same relative ordering as the original weights. This ensures that everything that can be derived by MAP from the original theory can also be derived from the theory with the reestimated parameters (but the probabilities of the possible worlds will be different). Imposing this restriction makes reestimating the parameters a non-trivial problem, for which we present a solution in this subsection.

Let E be a multiset of examples which we want to use to reestimate the parameters. Let Θ be a possibilistic logic theory, let $\Lambda = \{\lambda_1, \dots, \lambda_k\}$ be the set of weights in Θ , ordered increasingly, let $\lambda_{k+1} = \infty$ and let P_Θ be given by Eq. 3. A maximum likelihood estimate of the parameters is a solution of the following optimization problem:

- Variables: $\lambda'_1, \lambda'_2, \dots, \lambda'_{|\Lambda|}$.
- Maximize: $\prod_{\omega \in E} P(\omega) = \prod_{\lambda_i \in \Lambda} (1 - \lambda'_i)^{|E_{\lambda_i}|}$ where $E_\lambda = \{\omega \in E \mid \lambda = \max\{\lambda' \mid (\alpha, \lambda') \in \Theta \text{ and } \omega \not\models \alpha\}\}$.
- Subject to:

$$\lambda'_1 < \lambda'_2 < \dots < \lambda'_{|\Lambda|} \quad (4)$$

$$\sum_{i=1}^k (1 - \lambda'_i) \cdot (|M_{\lambda_{i+1}}^\top| - |M_{\lambda_i}^\top|) = 1 \quad (5)$$

where $M_{\lambda_1}^\top, \dots, M_{\lambda_j}^\top$ are as in Eq. 3 where we set $\alpha := \top$ (i.e. α is a tautology).

This is a nonlinear optimization problem which can be solved using off-the-shelf⁶ techniques of *geometric programming* [5]. In particular, the general *geometric programming problem* is:

- Minimize: $g_0(x)$
- Subject to:

$$g_i(x) \leq 1, i = 1, 2, \dots, m \quad (6)$$

$$x > 0 \quad (7)$$

where $x = (x_1, \dots, x_m) \in \mathbb{R}^m$ and g_i is a *posynomial*, i.e. $g_i(x) = \sum_{j=1}^{T_i} c_{ij} \prod_{k=1}^N x_k^{a_{ijk}}$ with $c_{ij} \geq 0$ and $a_{ijk} \in \mathbb{R}$.

We can follow the same strategy used for maximum likelihood parameter estimates of a multinomial distribution with order constraints in [6]. In order to formulate the problem as a geometric programming problem, we first substitute $\lambda'_i := 1 - x$. Then we replace $|E_{\lambda_i}|$ by the relative frequency $|E_{\lambda_i}|/|E|$ and change maximization to minimization by replacing the terms in the maximized product by their reciprocals. We also replace the strict inequalities in Eq. 4 by nonstrict. A solution close to the optimal but with the strict inequalities satisfied can then later be obtained by simply adding and subtracting suitably tiny numbers from the weights. We rewrite each of the nonstrict inequalities $x_i \geq x_{i+1}$ as a posynomial inequality $x_i^{-1} \cdot x_{i+1} \leq 1$. Finally, we also need to replace the equality in Eq. 5 by an inequality ≤ 1 , which clearly does not change the solution in this case.

4.3 Ensembles of predictors

Oftentimes, using a model ensemble, which aggregates the predictions of multiple different models, improves modelling performance. Given an ensemble of density estimation trees T_1, T_2, \dots, T_n , we can apply Transformation 1 or Transformation 2 to obtain a possibilistic encoding of the ensemble by constructing possibilistic logic theories $\Theta_1, \Theta_2, \dots, \Theta_n$ and combining them. We first show how to construct a *weighted* combination of two possibilistic logic theories

Transformation 3 (Weighted combination of two theories). *Let Θ_A and Θ_B be two possibilistic logic theories and let a and b , $a + b = 1$, be positive real numbers. The weighted combination of Θ_A and Θ_B with weights a and b (denoted by $a \cdot \Theta_A \oplus b \cdot \Theta_B$) is the possibilistic logic theory Θ_{AB} constructed as follows:*

- For every $(\alpha, \lambda) \in \Theta_A$, add $(\alpha, a \cdot \lambda)$ to Θ_{AB} .
- For every $(\beta, \mu) \in \Theta_B$, add $(\beta, b \cdot \mu)$ to Θ_{AB} .
- For every pair $(\alpha, \lambda) \in \Theta_A$, $(\beta, \mu) \in \Theta_B$ add $(\alpha \vee \beta, a\lambda + b\mu)$ to Θ_{AB} .

This transformation actually corresponds to a special case of a combination operator for possibilistic logic theories [3], from which we immediately obtain the following result.

Proposition 5. *Let Θ_A and Θ_B be possibilistic logic theories and π_A and π_B be the corresponding possibility distributions. Let p_A and p_B be two probability distributions on possible worlds and let a , b , $a + b = 1$ be positive real numbers. If for all $\omega \in \Omega$, $p_A(\omega) = \pi_A(\omega)$ and $p_B(\omega) = \pi_B(\omega)$ then also $a \cdot p_A(\omega) + b \cdot p_B(\omega) = \pi_{AB}(\omega)$ where π_{AB} is possibility distribution corresponding with $a \cdot \Theta_A \oplus b \cdot \Theta_B$.*

⁶ Geometric programming problems can be solved using, e.g. the CVX toolkit [15, 14].

To construct a uniform combination of possibilistic logic theories $\Theta_1, \Theta_2, \dots, \Theta_k$, we can iteratively apply the merging operator $((\dots (\frac{2}{3} \cdot (\frac{1}{2} \cdot \Theta_1 \oplus \frac{1}{2} \cdot \Theta_2) \oplus \frac{1}{3} \cdot \Theta_3) \oplus \dots))$. The caveat is that the size of the produced theory may grow exponentially with the number of combined theories. However, this can be mitigated if we allow some imprecision and apply the approximate pruning procedure from Section 4.1 while iteratively building the combination.

5 EXPERIMENTS

In this section we experimentally evaluate the proposed methods. We first provide some examples of learned possibilistic logic theories, after which we present a quantitative evaluation in Section 5.2.

5.1 Illustrative examples

We start by contrasting the interpretability of a learned possibilistic logic theory with a corresponding MRF.⁷ Then we illustrate how interpretability in some cases can be further improved by approximating the possibilistic logic theory using default rules.

Possibilistic logic and MRFs First, we use a credit-default dataset [29], where we only consider a subset of the variables for readability. Using the method proposed in this paper, we obtain the following possibilistic logic theory⁸:

$$\begin{aligned} &(\neg single, \lambda_0), (\neg gradSchool, \lambda_1), (male, \lambda_2), (single \vee male, \lambda_3), \\ &(male \vee \neg gradSchool, \lambda_4), (single \vee university, \lambda_5), \\ &(\neg highSchool, \lambda_6), (married \vee \neg highSchool, \lambda_7), \\ &(\neg otherMaritalStatus, \lambda_8), (\neg otherSchool, \lambda_8) \end{aligned}$$

As well as a number of integrity constraints such as $(\neg single \vee \neg married, 1)$. The theory contains several interesting rules, which capture the properties that hold for typical people who default on their credit. These people typically are not single, did not go to graduate school, and are males. If they are females, then typically they are single, etc. After simplifying the theory with integrity constraints, the most probable worlds have: *married, university, male*. While these pieces of information provide insight, the main advantage of being able to interpret the model is that we can understand exactly how it arrives at its predictions and potentially debug it by adding or removing rules. We now consider a learned MRF for the same dataset:

$$\begin{aligned} P(\omega) = \frac{1}{Z} \exp(&0.3 \cdot male + 3.8 \cdot gradSchool + 4.3 \cdot university \\ &+ 3.3 \cdot highSchool - 3.7 \cdot otherSchool + 5.0 \cdot married \\ &+ 5.1 \cdot single + 1.6 \cdot otherMaritalStatus \\ &- 1.0 \cdot (male \wedge otherSchool) - 0.2 \cdot (male \wedge otherMaritalStatus) \\ &- 1.7 \cdot (otherSchool \wedge married) - 1.6 \cdot (otherSchool \wedge single) \\ &- 10.3 \cdot (gradSchool \wedge university) \\ &- 9.4 \cdot (gradSchool \wedge highSchool) - 9.8 \cdot (university \wedge highSchool) \\ &- 2.9 \cdot (university \wedge otherSchool) - 11.2 \cdot (married \wedge single) \\ &- 8.1 \cdot (married \wedge otherMaritalStatus) \\ &- 8.1 \cdot (single \wedge otherMaritalStatus)) \end{aligned}$$

⁷ Recall that propositional Markov logic networks correspond to Markov random fields (MRFs).

⁸ Since in this section, we are only interested in MAP inference, we show only symbolic weights $\lambda_0 < \lambda_1 < \dots$ in the possibilistic logic theory. Note that the same cannot be done for MRFs.

Note that the last five lines correspond to the integrity constraints. Due to the additive combination of the formula's weights, its predictions are encoded intricately via the interaction between the various formulas. As this particular MRF is quite small, it may be possible to gain insight into its predictions with some effort, but with larger theories this quickly becomes impossible.

Possibilistic logic and default rules To improve interpretability, possibilistic logic theories can be approximated by sets of default rules (see Section 4.1). To illustrate this idea, Table 1 shows a possibilistic logic theory and its approximated set of default rules for a dataset about the presence of plants in different states of the US and provinces of Canada [16]. For illustrative purposes, we only consider California, Montana, New Mexico and Texas. For example, the default rule $nm \sim tx$ intuitively means that plants found in New Mexico are also usually found in Texas. Such rules can be written in a form that is easy to understand, even for people without training in logic.

5.2 MAP inference and marginal inference

We compare our possibilistic logic theories with learned MRFs on both MAP and marginal inference tasks. We have considered the NLCS, MSNBC, and Plants datasets, which have 16, 17 and 69 propositional variables, respectively. These datasets are divided into train, tune, and test sets. We learned the models on the train sets and report results on held-out test sets. We implemented the possibilistic approach in Java, using SAT4J [4] for SAT solving and ReISat [17] for model counting. For MRFs we used approximate MAP inference and Gibbs sampling from the Libra toolkit [22]. For MAP-inference evaluation, we compare the following models:

- PosLog** Obtained by Transformation 2 and logical simplification
- PosLog (50%)** Compacting PosLog to 50% of its size using the method from Section 4.1
- PosLog (10%)** Compacting PosLog to 10% of its size using the method from Section 4.1
- MRF** The L1 learned models from [21]
- Baseline** A model which predicts all variables to be false

To generate queries, we randomly selected k literals (where $1 \leq k \leq$ number of variables $- 1$) for each test example to serve as the evidence and then predicted the most probable assignment for the remaining variables. We measured both accuracy, which is the fraction of examples predicted correctly, and the average Hamming distance between the test-set example and the predicted world.

The results for the MAP inference experiments are shown in Figure 4 and the sizes of the respective models are in Table 2. In general, the possibilistic logic theories outperform MRFs for small evidence sets and do slightly worse for larger ones. Intuitively, the possibilistic logic theories approximate the density in a coarser way than the MRFs. This seems to lead to more robust results for hard problem instances (i.e. small evidence sizes), but less precise predictions for the easier instances. Interestingly, approximately compacting the possibilistic logic theories hardly affects the quality of the results in most cases. For NLCS, however, the 10% theory has been reduced to the point that it only suggests to set everything to false, which is why the result in this case coincides with the baseline.

To evaluate the performance on marginal queries, we randomly sample conjunctions and compute the marginals of the conjunctions on the test set. Figure 5 shows the queries' predicted and empirical test-set probabilities for both possibilistic logic and MRFs on the NLCS dataset. In this case, MRFs always obtained higher marginal

Table 1. **Left:** A possibilistic logic theory modelling a subset of the Plants dataset containing four states: California (ca), Montana (mt), New Mexico (nm) and Texas (tx). **Right:** An approximation of the possibilistic logic theory by short default rules (with antecedents being conjunctions of at most two literals).

Possibilistic logic theory

$$\begin{aligned}
 &(ca \vee nm \vee \neg tx \vee \neg mt, \lambda_{14}), (nm \vee \neg tx \vee \neg mt, \lambda_{13}), (ca \vee \neg tx \vee \neg mt, \lambda_{12}), \\
 &(ca \vee \neg nm \vee \neg mt, \lambda_{11}), (tx \vee \neg ca \vee \neg nm \vee mt, \lambda_{10}), (\neg ca \vee \neg tx \vee nm, \lambda_9), \\
 &(\neg ca \vee \neg nm \vee mt, \lambda_8), (\neg nm \vee tx \vee \neg mt, \lambda_7), (\neg ca \vee nm \vee \neg mt, \lambda_6), \\
 &(tx \vee \neg nm, \lambda_5), (\neg tx \vee \neg mt, \lambda_4), (\neg mt, \lambda_3), (\neg nm, \lambda_2), (\neg tx, \lambda_1), (\neg ca, \lambda_0)
 \end{aligned}$$

Approximation by default rules

$$\begin{aligned}
 &\sim \neg ca, \sim \neg mt, \sim \neg nm, \sim \neg tx, nm \sim tx, \\
 &ca \wedge mt \sim nm, ca \wedge mt \sim tx, ca \wedge nm \sim mt, \\
 &ca \wedge tx \sim mt, ca \wedge tx \sim nm, nm \wedge mt \sim ca, \\
 &tx \wedge mt \sim ca, tx \wedge mt \sim nm
 \end{aligned}$$

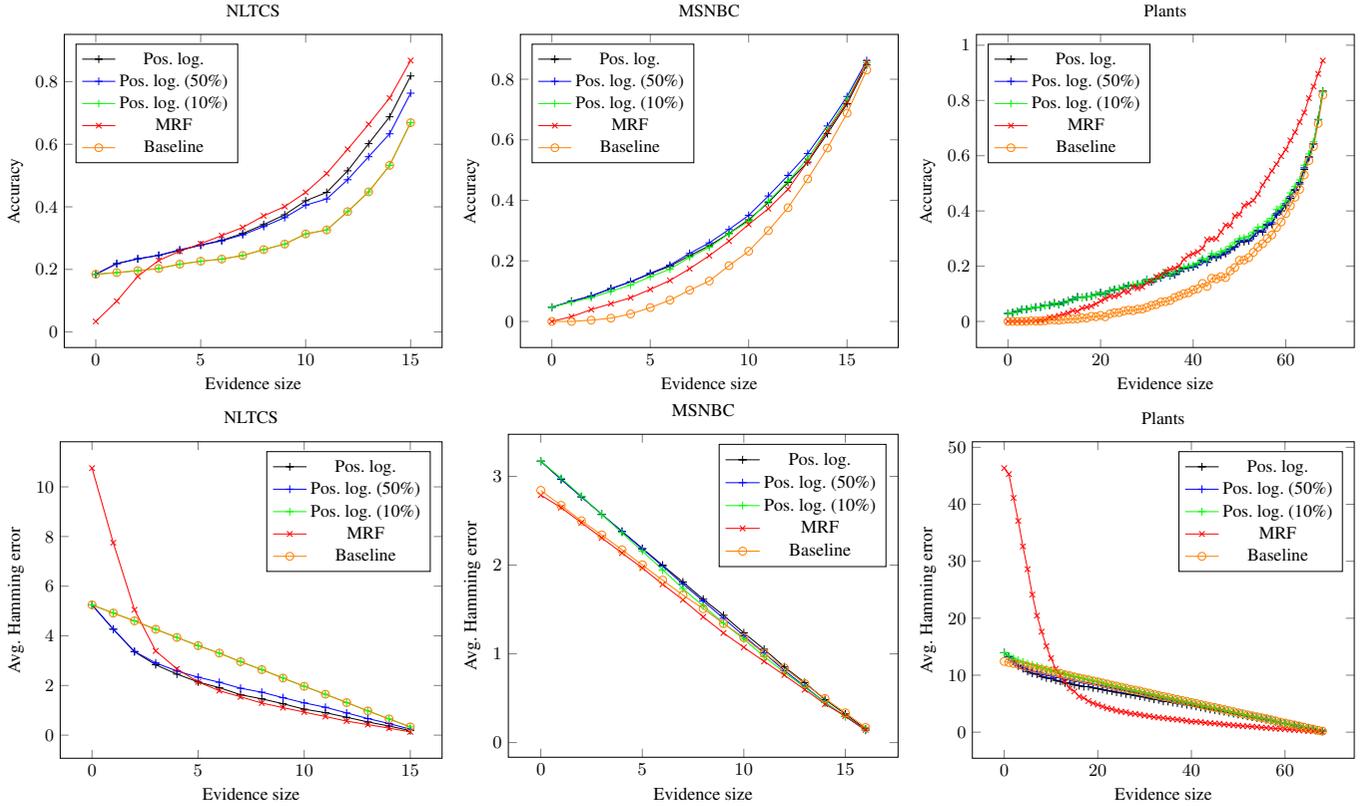


Figure 4. **Top:** Fraction of worlds correctly predicted by MAP inference on NLTCS, MSNBC and Plants datasets, measured on hold-out test sets. **Bottom:** Average Hamming error of possible worlds predicted by MAP inference, measured on hold-out test sets.

Table 2. The number of rules, average number of literals in a rule or branch for a tree, and the size which is the sum of rule lengths or number of nodes of a tree.

Dataset	Model	#Rules (#Branches)	Avg. length	Size
NLTCS	PosLog	121	3.0	363
	PosLog (50%)	71	2.4	172
	PosLog (10%)	16	1	16
	Tree	122	10.9	243
	MRF	135	1.9	254
MSNBC	PosLog	258	4.5	1153
	PosLog (50%)	172	3.3	572
	PosLog (10%)	53	2.1	109
	Tree	259	10.8	517
	MRF	136	1.9	289
Plants	PosLog	632	7.15	4523
	PosLog (50%)	467	4.8	2244
	PosLog (10%)	198	2.3	446
	Tree	655	18.6	1306
	MRF	2322	2.0	4713

log-likelihood. Nevertheless, the possibilistic logic theories' still offer competitive estimates along with the improved interpretability. The other datasets offer qualitatively similar results.

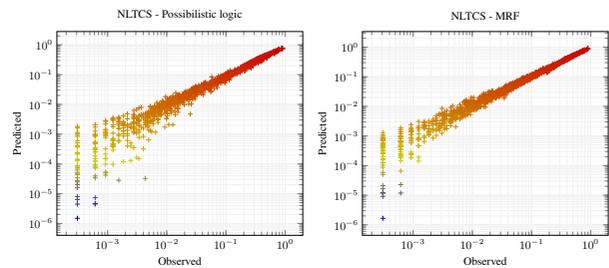


Figure 5. Scatter plots of estimated and empirical marginal probabilities of randomly generated queries on NLTCS dataset. **Left:** Possibilistic logic. **Right:** MRF.

6 CONCLUSIONS

We have introduced a practical method for constructing interpretable possibilistic logic models of probability distributions. The learned models support a variety of inference tasks, such as computing MAP queries and estimating marginal probabilities. Owing to the properties of possibilistic logic, the learned models can be easily edited by explicitly modifying, adding or removing logical rules, or they can be combined together. To maintain the ability to compute marginal probabilities after such modifications, we have proposed a parameter reestimation method based on geometric programming. Our experiments suggest that the method can be very useful for constructing interpretable logical theories from data.

ACKNOWLEDGEMENTS

This work has been supported by a grant from the Leverhulme Trust (RPG-2014-164). Jesse Davis is partially supported by the KU Leuven Research Fund (C22/15/015), and FWO-Vlaanderen (G.0356.12, SBO-150033).

A APPENDIX

A.1 Probability-possibility transformation

A standard method to give a probabilistic interpretation to possibility degrees is by associating a possibility measure Π with a family of probability measures, defined as $\mathcal{P}(\Pi) = \{P \mid P(A) \leq \Pi(A), \forall A \subseteq \Omega\}$. This view leads to the following probability-possibility transformation [11]. Let p be a probability distribution on $\Omega = \{\omega_1, \dots, \omega_n\}$ and assume w.l.o.g. that $p(\omega_i) \geq p(\omega_{i+1})$. Then p induces a possibility distribution π_p defined as $\pi_p(\omega_1) = 1$ and for $i > 1$:

$$\pi_p(\omega_i) = \begin{cases} \sum_{j=i}^n p(\omega_j) & \text{if } p(\omega_{i-1}) > p(\omega_i) \\ \pi_p(\omega_{i-1}) & \text{otherwise} \end{cases}$$

In [19], a syntactic counterpart of this transformation was used to associate each Markov logic network M with a possibilistic logic theory Θ , such that for p the probability distribution associated with M and π the possibility distribution associated with Θ it holds that $p(\omega_1) \leq p(\omega_2)$ iff $\pi(\omega_1) \leq \pi(\omega_2)$. As a result, $(\Theta, \alpha) \vdash_{\text{poss}} \beta$ iff β is true in all the most probable models of α (w.r.t. p). Other probabilistic interpretations of possibility distributions view possibility degrees as the contour function of a mass assignment, in the context of Dempster-Shafer evidence theory [28], or interpret possibility distributions as likelihood functions [9].

For completeness, we present a variant of Transformation 1 corresponding to a direct syntactic counterpart of the standard probability-possibility transformation.

Transformation 4. Let \mathcal{V} be the set of propositional variables. Let T be a density estimation tree. Let $\mathcal{B} = \{B_1, \dots, B_k\}$ be the set of all branches of the tree, represented as conjunctions, and let p_1, \dots, p_k be the estimated probabilities of worlds consistent with the respective branches, i.e. if $\omega \models B_i$ then $p(\omega) = p_i$. Let us assume w.l.o.g. that $p_i \geq p_{i+1}$. Let us define $w_1 = 1$. For $i > 1$, we define:

$$w_i = \begin{cases} \sum_{j=i}^k p_j \cdot 2^{|\mathcal{V}| - |B_j|} & \text{if } p_{i-1} > p_i \\ w_{i-1} & \text{otherwise} \end{cases}$$

We define the possibilistic logic theory corresponding to T as $\Theta_T = \{(\neg B_i, 1 - w_i) \mid B_i \in \mathcal{B}\}$.

A.2 Lexicographic pruning of default rule theories

Default rules of the form $\alpha \sim \beta$, intuitively meaning “if α then typically β ”, offer a convenient way to make what is encoded by a possibilistic logic theory more explicit. If we take a purely qualitative view of possibilistic logic theories (i.e. if we see the weights merely as a way of specifying a ranking of possible worlds), a possibilistic logic theory Θ can be seen as a compact encoding of a set of default rules, i.e. a default theory, where a default $\alpha \sim \beta$ is in that theory if and only if $(\Theta, \{\alpha\}) \vdash_{\text{poss}} \beta$. The resulting default rules tend to be easy to interpret, but an exhaustive enumeration of all defaults would lead to theories in which many of the defaults are redundant. To cope with this problem, we rely on the *lexicographic closure* of default rules, which we describe next. The lexicographic closure [1] is one of several closures that have been studied in the field of non-monotonic reasoning, which allow us to represent an exhaustive set of defaults by a smaller set of defaults from which the complete set can be reconstructed. Smaller sets are usually easier for humans to understand.

To describe the lexicographic closure of default rules, first recall the *Z-ordering* from [24]. A default $\alpha \sim \beta$ is said to be *tolerated* by a set of defaults $\gamma_1 \sim \delta_1, \dots, \gamma_m \sim \delta_m$ if the classical formula $\alpha \wedge \beta \wedge \bigwedge_i (\neg \gamma_i \vee \delta_i)$ is consistent. The *Z-ordering* is a stratification $\Delta_1, \dots, \Delta_k$ of a set Δ of default rules, where each Δ_j contains all defaults $\alpha \sim \beta$ from $\Delta \setminus (\Delta_1 \cup \dots \cup \Delta_{j-1})$ which are tolerated by $\Delta \setminus (\Delta_1 \cup \dots \cup \Delta_{j-1})$. It can be shown that such a stratification always exists when Δ satisfies some natural consistency properties (see [24] for details). Intuitively, Δ_1 contains the most general default rules, Δ_2 contains exceptions to the rules in Δ_1 , Δ_3 contains exceptions to the rules in Δ_2 , etc. The lexicographic closure of a set of default rules is given as follows [1, 2]. For a possible world ω , we write $\text{sat}(\omega, \Delta_j)$ for the number of defaults from Δ_j that are satisfied by ω , i.e. $\text{sat}(\omega, \Delta_j) = |\{\alpha \sim \beta : (\alpha \sim \beta) \in \Delta_j, \omega \models \neg \alpha \vee \beta\}|$. We say that an interpretation ω_1 is *lex-preferred* over an interpretation ω_2 , written $\omega_1 \prec \omega_2$, if there exists a j such that $\text{sat}(\omega_1, \Delta_j) > \text{sat}(\omega_2, \Delta_j)$ while $\text{sat}(\omega_1, \Delta_i) = \text{sat}(\omega_2, \Delta_i)$ for all $i > j$. The default $\alpha \sim \beta$ is in the *lex.closure* of Δ if β is satisfied in all the most lex-preferred models of α , i.e. $\forall \omega \in \llbracket \alpha \rrbracket : (\omega \not\models \beta) \Rightarrow \exists \omega' \in \llbracket \alpha \rrbracket : \omega' \prec \omega$, where $\llbracket \alpha \rrbracket$ is the set of models of α .

Now we can describe pruning of a (large) set of default rules Δ , closed under the axioms of System P and rational monotonicity [18]. Our aim is not to construct the smallest set of defaults, as such a set could actually be more difficult to interpret. In particular, to maintain interpretability we only remove a rule if it is “implied” by a set of rules which are all shorter or equally long (as shorter rules are more interpretable). Furthermore note that methods for constructing the smallest set of defaults are likely to be computationally harder. Let $\Delta_1, \Delta_2, \dots, \Delta_k$ be the *Z-ordering* of Δ . Let us write $|\alpha \sim \beta|$ for the length of the default $\alpha \sim \beta$, e.g. the sum of literal occurrences in the antecedent α and consequent β . First we iteratively prune rules which are redundant in the rational closure sense – we remove a rule $\alpha \sim \beta$ if $(\Theta_R, \{\alpha\}) \vdash_{\text{poss}} \beta$ where $\Theta_R = \bigcup_{i=1}^k \{(\neg \gamma \vee \delta, 1/(k-i+1)) \mid \gamma \sim \delta \in \Delta_i\} \setminus \{\neg \alpha \vee \beta\}$. Then we iteratively prune the rules in Δ as follows. Iterating i from 1 upwards, let $\alpha \sim \beta \in \Delta_i$. Let $L = |\alpha \sim \beta|$. Let $\Phi_j = \{\neg \gamma \vee \delta \mid \gamma \sim \delta \in \Delta_j \setminus \{\alpha \sim \beta\} \text{ and } |\gamma \sim \delta| \leq L \text{ and } \alpha \wedge (\neg \gamma \vee \delta) \not\models \perp\}$ and let $\Theta = \bigcup_{j=1}^{i-1} \{(\varphi, 1/(k-j+1)) \mid \varphi \in \Phi_j\}$ be a possibilistic logic theory. If $(\Theta, \{\alpha\}) \vdash_{\text{poss}} \beta$ we remove $\alpha \sim \beta$ from Δ and repeat this process for other rules in Δ . It can be shown that all default rules from the initial set are contained in the lexicographic closure of the resulting, pruned set (although the closures themselves might differ if the original set was not closed).

REFERENCES

- [1] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade, 'Inconsistency management and prioritized syntax-based entailment', in *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 93, pp. 640–645, (1993).
- [2] Salem Benferhat, Jean F. Bonnefon, and Rui da Silva Neves, 'An overview of possibilistic handling of default reasoning, with experimental studies', *Synthese*, **146**(1-2), 53–70, (2005).
- [3] Salem Benferhat, Didier Dubois, and Henri Prade, *Aggregation and Fusion of Imperfect Information*, chapter From Semantic to Syntactic Approaches to Information Combination in Possibilistic Logic, 141–161, Physica-Verlag HD, Heidelberg, 1998.
- [4] D. Le Berre and A. Parrain, 'The SAT4J library, release 2.2.', *Journal on Satisfiability, Boolean Modeling and Computation*, **7**, 50–64, (2010).
- [5] Stephen Boyd, Seung-Jean Kim, Lieven Vandenbergh, and Arash Hassibi, 'A tutorial on geometric programming', *Optimization and Engineering*, **8**(1), 67–127, (2007).
- [6] Dennis L Bricker, Kenneth O Kortanek, and Lina Xu, 'Maximum likelihood estimates with order restrictions on probabilities and odds ratios: a geometric programming approach', *Advances in Decision Sciences*, **1**(1), 53–65, (1997).
- [7] S. Della Pietra, V. Della Pietra, and J. Lafferty, 'Inducing features of random fields', **19**, 380–392, (1997).
- [8] D. Dubois, J. Lang, and H. Prade, 'Possibilistic logic', in *Handbook of Logic in Artificial Intelligence and Logic Programming*, ed., D. Nute D. Gabbay, C. Hogger J. Robinson, volume 3, 439–513, Oxford University Press, (1994).
- [9] D. Dubois, Serafn Moral, and Henri Prade, 'A semantics for possibility theory based on likelihoods', *Journal of Mathematical Analysis and Applications*, **205**(2), 359 – 380, (1997).
- [10] D. Dubois and H. Prade, 'Possibility theory: qualitative and quantitative aspects', in *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, eds., D.M. Gabbay and P. Smets, volume 1, 169–226, Kluwer Academic, (1998).
- [11] D. Dubois and Henri Prade, 'On several representations of an uncertain body of evidence', in *Fuzzy Information and Decision Processes*, eds., M.M. Gupta and E. Sanchez, 167–181, North-Holland, (1982).
- [12] Didier Dubois, Laurent Foulloy, Gilles Mauris, and Henri Prade, 'Probability-possibility transformations, triangular fuzzy sets, and probabilistic inequalities', *Reliable Computing*, **10**(4), 273–297, (2004).
- [13] F. Dupin de Saint-Cyr, J. Lang, and T. Schiex, 'Penalty logic and its link with Dempster-Shafer theory', in *Uncertainty in Artificial Intelligence*, pp. 204–211, (1994).
- [14] Michael Grant and Stephen Boyd, 'Graph implementations for nonsmooth convex programs', in *Recent Advances in Learning and Control*, eds., V. Blondel, S. Boyd, and H. Kimura, Lecture Notes in Control and Information Sciences, 95–110, Springer-Verlag Limited, (2008).
- [15] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. <http://cvxr.com/cvx>, March 2014.
- [16] Wilhelmiina Hämäläinen and Matti Nykänen, 'Efficient discovery of statistically significant association rules', in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pp. 203–212, (2008).
- [17] Roberto J. Bayardo Jr. and Joseph Daniel Pehoushek, 'Counting models using connected components', in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA.*, pp. 157–162, (2000).
- [18] S. Kraus, D. Lehmann, and M. Magidor, 'Nonmonotonic reasoning, preferential models and cumulative logics', *Artif. Intelligence*, **44**(1-2), 167–207, (1990).
- [19] O. Kuželka, J. Davis, and S. Schockaert, 'Encoding markov logic networks in possibilistic logic', in *Uncertainty in Artificial Intelligence, UAI*, (2015).
- [20] Jérôme Lang, D. Dubois, and Henri Prade, 'A logic of graded possibility and certainty coping with partial inconsistency', in *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 188–196, (1991).
- [21] D. Lowd and J. Davis, 'Improving markov network structure learning using decision trees', *Journal of Machine Learning Research*, **15**, 501–532, (2014).
- [22] Daniel Lowd and Amirmohammad Rooshenas, 'The libra toolkit for probabilistic models', *CoRR*, **abs/1504.00110**, (2015).
- [23] M. Meila and M. Jordan, 'Learning with mixtures of trees', **1**, 1–48, (2000).
- [24] J. Pearl, 'System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning', in *3rd Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 121–135, (1990).
- [25] Parikshit Ram and Alexander G Gray, 'Density estimation trees', in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 627–635. ACM, (2011).
- [26] Matthew Richardson and Pedro Domingos, 'Markov logic networks', *Machine Learning*, **62**(1-2), 107–136, (2006).
- [27] Amirmohammad Rooshenas and Daniel Lowd, 'Learning sum-product networks with direct and indirect variable interactions', in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 710–718, (2014).
- [28] G. Shafer, *A mathematical theory of evidence*, Princeton University Press, 1976.
- [29] I-Cheng Yeh and Che-hui Lien, 'The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients', *Expert Syst. Appl.*, **36**(2), 2473–2480, (2009).
- [30] L.A. Zadeh, 'Fuzzy sets as a basis for a theory of possibility', *Fuzzy Sets and Systems*, **1**, 3–28, (1978).

Unsupervised Ranking of Knowledge Bases for Named Entity Recognition

Yassine Mrabet^{1, 2} and Halil Kilicoglu¹ and Dina Demner-Fushman¹

Abstract. With the continuous growth of freely accessible knowledge bases and the heterogeneity of textual corpora, selecting the most adequate knowledge base for named entity recognition is becoming a challenge in itself. In this paper, we propose an unsupervised method to rank knowledge bases according to their adequacy for the recognition of named entities in a given corpus. Building on a state-of-the-art, unsupervised entity linking approach, we propose several evaluation metrics to measure the lexical and structural adequacy of a knowledge base for a given corpus. We study the correlation between these metrics and three standard performance measures: precision, recall and F1 score. Our multi-domain experiments on 9 different corpora with 6 knowledge bases show that three of the proposed metrics are strong performance predictors having 0.62 to 0.76 Pearson correlation with precision and 0.96 correlation with both recall and F1 score.

1 Background

With the tremendous growth in the amount of textual data, extracting semantic information from unstructured texts has become critical in several applications such as information retrieval, marketing, content management and question answering. Named entity recognition (NER), the task of identifying and categorizing textual mentions into pre-defined semantic categories, plays a key role in such applications. It is also often a prerequisite for other text mining processes such as relation extraction, keyword identification and document clustering.

The range of named entities covered by the NER task has grown continuously since the first MUC conference³. Starting with a few named entity categories, such as PERSON, LOCATION, and ORGANIZATION; nowadays, the entity linking task [22] addresses linking of textual mentions to entities from a reference database or knowledge base (KB) with no semantic restrictions on the type of entities. Besides their role as reference data, KBs are also increasingly used in NER methods. For instance, they have been used in designing features for supervised learning [24], as labeling sources in constructing training corpora [10, 28], and as resources for named entity disambiguation in both unsupervised [17] and supervised [25] approaches.

With the exponential growth of domain-specific KBs and the increasing heterogeneity in open-domain KBs, it becomes important to assess which KB is suitable to extract named entities in a given text or corpus. Qualitative assessments, while useful, can be time-consuming and require domain expertise [8, 4]. On the other hand, an automatic, quantitative evaluation based on KB and corpus char-

acteristics can assist significantly in assessing suitability. We refer to this type of automatic evaluation as *knowledge base ranking for named entity recognition (KB ranking)*.

The term *knowledge base ranking* is often used in the literature to refer to *fact ranking* or *entity ranking*, the task of finding the facts or entities that are the most relevant for a keyword query or a structured query [5, 2]. In this paper, we do not address this task.

Our work can be situated within the wider field of ontology evaluation (see [27] for a review). However, most works in this area cover tasks that are out of the scope of our study. These tasks include, for instance, the evaluation of the general representational (or domain) adequacy of a KB [8, 4], the inner cohesion of an ontology [31], finding relevant criteria for ontology design [18, 30], or checking logical consistency [9]. These criteria are constant for a given knowledge base and do not change according to different contexts of application.

As we are primarily interested in the use of knowledge bases for NER, only a few related studies stand out. Lozano-Tello et al. [15] proposed a generic framework called *OntoMetric* to evaluate the suitability of an ontology for a given application. They defined manually 160 generic features related to ontologies and designed an interface to help users decide which ontology is more relevant to their use case. The users have to manually enter their objective and criteria according to the defined vocabulary. This manual approach is limited and can not be applied to more complex applications such as NER, where we need to take into account the corpus characteristics in addition to ontology features. More generally, manual approaches are not suited to learning relevant features when several empirical observations are needed (e.g., observations on different NER corpora).

Gangemi et al. [8] proposed a formal model to evaluate ontologies, including a component for intended use situation. In particular, they considered the use of natural language processing to evaluate ontologies according to annotated corpora. However, the goal of the comparison was to evaluate the general usefulness (or quality) of an ontology: e.g., the frequency of occurrence of an ontology concept in the annotated corpus is used to measure the importance of the concept. In the same line, they also proposed to use the hierarchy of concepts and entropy to have an estimation of the usefulness of ontology concepts.

Their objective is basically different from ours; we want to evaluate the adequacy of a KB for NER in a given corpus. For the same KB, this evaluation is expected to give different results on different corpora. More precisely, the question that we want to answer is: “if we want to use a KB to find and disambiguate named entities in a given text, how could we know which KB will provide better results?”.

Baseline approaches to KB ranking for NER could be derived from

¹ National Library of Medicine, Bethesda, MD, United States {mrabety, kilicogluh, ddemner}@mail.nih.gov

² CNRS/LORIA, France, yassine.mrabet@loria.fr

³ <http://cs.nyu.edu/cs/faculty/grishman/muc6.html>

KB indexes such as the Linked Open Vocabulary⁴, by selecting the KBs that have more candidate entities for a given textual mention; however, such an approach does not allow ranking of the KBs by their suitability for NER as other aspects come into play. These aspects include, for example, the ambiguity of the text to be annotated from the KB point-of-view, and the contextual similarity between the textual context of the named entities and the KB graph linking the corresponding (or candidate) KB concepts.

To the best of our knowledge, no automatic solution has been proposed previously to rank KBs according to their suitability for NER. This can be partly explained by the lack of KB-agnostic annotation tools for NER; most of the existing tools rely on a specific combination of a learning corpus and a KB.

In this paper, we propose a novel KB ranking method based on an unsupervised NER method. We formally define several evaluation metrics related to the lexical ambiguity of textual corpora and to the structural similarity between the knowledge base and the text to be annotated.

We studied the relevance of the proposed evaluation metrics in ranking 6 different KBs from both the open and biomedical domains for NER in 9 different corpora. Our results show that the proposed metrics are strongly correlated with precision, recall and F1 measures and that they can be used as predictors of the adequacy of a KB for NER in a given textual corpus. This finding paves the way to automatic and fine-grained selection and combination of knowledge bases for named entity recognition.

The remainder of the paper is structured as follows. In the next section, we describe the KB-agnostic named entity recognition method that underlies our approach and the evaluation metrics in more detail. We discuss the motivation behind the different metrics. In Section 3, we present our experiments. Finally, we discuss and analyze the results and perspectives in Section 4 before giving our concluding remarks.

2 Methods

In the current work, we consider a KB to consist of a set of concepts, instances, relations and a set of labels representing natural language expressions of concepts and instances. To ensure the required portability for our approach, we built an unsupervised NER method from an existing, KB-agnostic tool for entity linking called *KODA* [17]. In this section, we present the overall NER process and the evaluation metrics proposed to rank the KBs.

2.1 Named entity recognition

KODA is a KB-agnostic entity linking tool. It exploits TF-IDF indexing of the KB labels, and KB relations to disambiguate the entities in the input text. In the course of this study, we modified and extended *KODA* to build a NER method and used the extended tool as the basis of our experiments. Given a text t and a knowledge base k , our NER process follows the steps outlined below.

- Split t into sentences and perform part-of-speech tagging.
- For each sentence, select textual mentions corresponding to a sequence of allowed part-of-speech tags (e.g., noun, adverb, adjective).
- Use each textual mention as a keyword query to look up KB entities based on TF-IDF search. If no exact match is found between

the mention and the KB entities, select all subsequences of words as potential candidates. The mentions recognized at this step are referred to as *candidate textual mentions*.

- Disambiguate ambiguous mentions (i.e., those that have more than one corresponding KB entity with the maximum TF-IDF score). Disambiguation is performed according to *global coherence*: i.e., select the entity that has more KB relations with the entities obtained from other textual mentions in t . This step is accomplished using Integer Linear Programming [17]. The generic disambiguation process can be viewed as the *selection of the subgraph of the KB that is the most similar to the textual context being annotated*.
- Determine the semantic category of the disambiguated textual mention, by using mappings of KB concepts, as detailed below.
- Filter the entities to keep only the semantic categories considered in the corpus.
- In the case of nested entities or overlapping entities with the same type (e.g., “*San Francisco, CA*” vs. “*San Francisco*”), keep only the entity with the best TF-IDF score.

In order to classify these named entities according to the considered semantic types (e.g., PERSON, LOCATION, ORGANIZATION), we built manual mappings between the concepts of the KBs and the semantic types considered in the target corpora. As large and dense concept hierarchies might be difficult to browse, we first computed the transitive closure offline by considering only the subset of instantiation facts (e.g., *RDF type relation*) and subsumption facts (e.g., *RDFS subclassOf relation*). Next, we sorted the concepts according to their frequency in the closure and extracted manually the relevant concepts, i.e., those that can be mapped to a named entity category according to the corpus. For example, *dbpedia:Place* was mapped to LOCATION in the CoNLL 2003 corpus [23], and *yago:wordnet_illness_114061805* was mapped to DISEASE in the I2B2 corpus [32].

In the online NER step, we collect all the classes associated with the KB entities linked to the disambiguated textual mentions then use the mappings to associate these mentions with a semantic type. If one mention is associated with more than one semantic type, it is considered as ambiguous and discarded from the results of the NER process.

Figure 1 shows an example of named entities recognized by our NER method using DBpedia on a sentence from a New York Times corpus [14]. In this example, there are multiple candidates with the same (best) TF-IDF score for the term “Malone” in the KB. Global coherence led to the selection of only one candidate (*dbpedia:Kevin_Malone*) because it is linked with the entities *dbpedia:Carlos_Perez_(pitcher)* and *dbpedia:San_Francisco* in DBpedia triples.

2.2 Evaluation of KB Adequacy for NER

We propose several evaluation metrics by defining and combining three elementary principles:

1. *Ambiguity*: How ambiguous is the text with respect to a KB?
2. *Coverage*: How much of the text has been annotated and disambiguated with the KB?
3. *Structure*: To what extent did KB relations participate in the disambiguation?

To define relevant metrics taking into account these 3 aspects, we make the distinction between mentions recognized lexically with the

⁴ <http://lov.okfn.org/dataset/lov/>

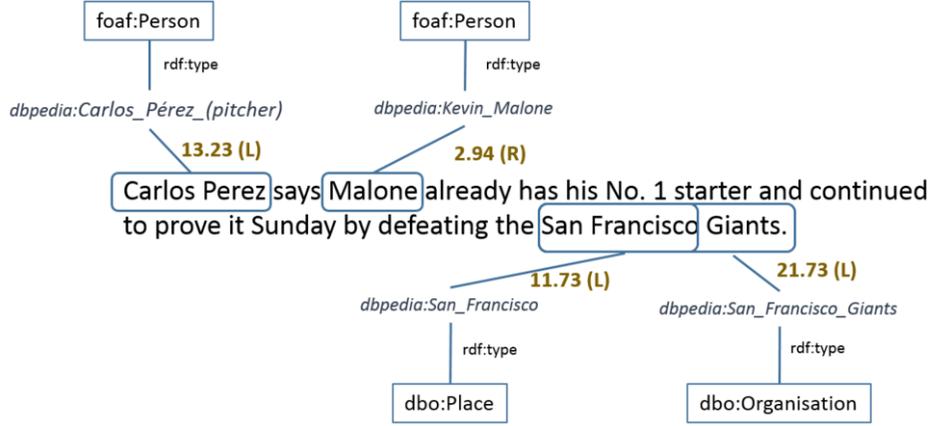


Figure 1. Example of named entity recognition using DBpedia. The numbers on the edges are TD-IDF scores. *L* indicates that disambiguation was performed lexically. *R* indicates that disambiguation was performed using KB relations.

best TF-IDF score and the ambiguous mentions disambiguated using the KB relations. We denote the set of all candidate mentions in a corpus C according to a KB k as $M_k(C)$ and the set of disambiguated mentions as $D_k(C)$. Figure 2 presents the mentions sets that are generated by our KB-agnostic recognition process.

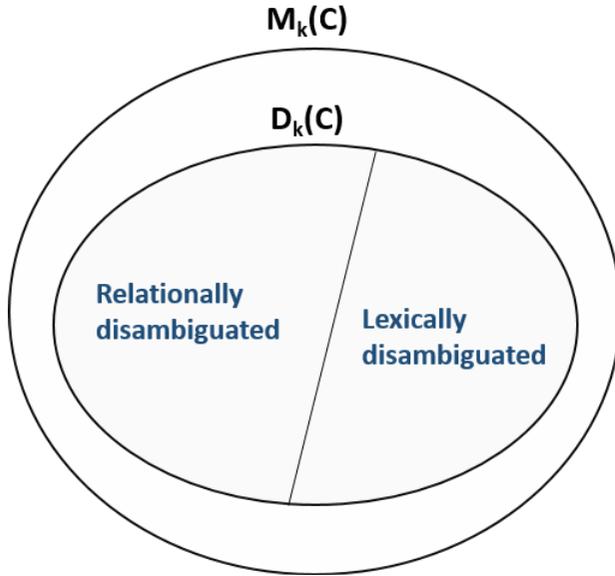


Figure 2. Named Entity Sets as recognized with Knowledge Base k in corpus C

We propose and study 10 evaluation metrics. These metrics are Coverage (V), Disambiguation Ratio (D), Lexical Disambiguation Ratio (L), Relation Disambiguation Ratio (R), Average Corpus Ambiguity (A), Average TF-IDF Score (S), Lexical Adequacy ($LEXQ$), Graph Adequacy (GQ), Weighted Quality (WQ) and Overall PERFORMANCE INDICATOR ($OPERA$). They are described below.

Coverage ($V_k(C)$): This metric indicates the percentage of corpus tokens that have been annotated and disambiguated with the knowl-

edge base k for corpus C .

$$V_k(C) = \frac{\sum_{m \in D_k(C)} |tokens(m)|}{|tokens(C)|} \quad (1)$$

Disambiguation Ratio ($d_k(C)$): This metric indicates the ratio of textual mentions that have been disambiguated among the set of detected (annotated) mentions.

$$d_k(C) = \frac{|D_k(C)|}{|M_k(C)|} \quad (2)$$

Lexical Disambiguation Ratio ($L_k(C)$): This metric indicates the ratio of mentions that are disambiguated using only their TF-IDF score. Low $L_k(C)$ values indicate a bigger disambiguation problem. For a given mention $m \in M_k(C)$ it is computed as follows:

$$L_k(C) = \frac{|\{m \in M_k(C) \text{ s.t. } N_{max}(m, k) = 1\}|}{|M_k(C)|} \quad (3)$$

Where $N_{max}(m, k)$ is the number of entities in k that share the maximum TF-IDF score for the mention m .

Relation Disambiguation Ratio ($R_k(C)$): This feature indicates the ratio of textual mentions from $D_k(C)$ that have been disambiguated using KB relations (cf. section 2.1). Higher values of $R_k(C)$ indicate a stronger participation of the KB graph in disambiguation.

$$R_k(C) = \frac{|\{m \in D_k(C) \text{ s.t. } N_{max}(m, k) > 1\}|}{|M_k(C)|} = 1 - L_k(C) \quad (4)$$

Average Score (S_k): This is the average best TF-IDF score for mentions in $M_k(C)$. A high TF-IDF average would indicate that the corpus targets specific subsets of the KB that use highly informative terms. S_k is computed as follows:

$$S_k(C) = \frac{\sum_{m \in M_k(C)} score(m, k)}{|M_k(C)|} \quad (5)$$

Average Corpus Ambiguity ($A_{k,T}(C)$): This metric represents the average ambiguity level in a corpus C according to a KB k (cf. equation 7). It is the average of the **Lexical Ambiguity** of each mention

m in $M_k(C)$. Highly ambiguous mentions are likely to be unclassifiable or wrongly classified by the KB. The formula for Lexical Ambiguity ($a_{k,T}(m)$) is presented in equation 6 below. T is the observation threshold, i.e., $a_{k,T}(m)$ is computed against the first T search results and $r \geq N_{max}(m, k)$ is the actual number of results from the KB index.

$$a_{k,T}(m) = \frac{\text{Min}(N_{max}(m, k), T)}{\text{Min}(r, T)} \quad (6)$$

$$A_{k,T}(C) = \frac{\sum_{m \in M_k(C)} a_{k,T}(m)}{|M_k(C)|} \quad (7)$$

From these elementary metrics, we derive several composite evaluation metrics to predict both the quantity and the quality of the disambiguation provided by one KB for a given corpus.

Lexical Adequacy ($LEXQ_k(C)$): This metric represents the absolute ratio of named entities that have been disambiguated with TF-IDF search.

$$LEXQ_k(C) = L_k(C) \times V_k(C) \quad (8)$$

Graph Adequacy ($GQ_k(C)$): This metric indicates how useful the KB graph is in disambiguating named entities in a given corpus. Coverage is used as a coefficient to take into account the discrepancies in size and coverage between different KBs.

$$GQ_k(C) = R_k(C) \times V_k(C) \quad (9)$$

Weighted Quality ($WQ_k(C)$): This metric is a weighted combination of:

- A quality indicator for lexical disambiguation ($\frac{\text{Norm}(S_k(C))}{1 + A_{k,T}(C)}$), which uses the normalized value of average TF-IDF score ($\text{Norm}(S_k(C))$), and the average corpus ambiguity ($A_{k,T}(C)$). The motivation here is (i) that a high average of TF-IDF values w.r.t. other KBs indicate that the textual mentions in the corpus are using (highly) informative terms from the KB and (ii) the more ambiguous the mentions are, the riskier is the selection of the one mention with best TF-IDF score.
- A quality indicator for relational disambiguation, consisting in the average corpus ambiguity $A_{k,T}(C)$. The motivation here is that having more candidate KB entities to chose from increases the odds of finding relations between the good candidates. However, if ambiguity is too high, it can lead to relations between the wrong KB entities. In practice, such high-ambiguity threshold would depend (i) on the considered knowledge base, (ii) on the targeted corpus, and (iii) on the observation threshold T used to compute $a_{k,T}(m)$. Therefore, for relational disambiguation, finding a balanced estimation between the positive and negative impact of ambiguity is not straightforward. In this paper, we chose to consider only the positive aspect of ambiguity for relational disambiguation and to analyze the impact of this choice in our experiments.

We use the contribution of lexical disambiguation $L_k(C)$ to weight the quality indicator for lexical disambiguation, and the contribution of relational disambiguation $R_k(C)$ to weight the quality indicator for relation-based disambiguation (we have from equation 4 that $L_k(C) = 1 - R_k(C)$). The final formula for the overall quality indicator $WQ_k(C)$ is:

$$WQ_k(C) = (L_k(C) \times \frac{\text{Norm}(S_k(C))}{1 + A_{k,T}(C)}) + (R_k(C) \times A_{k,T}(C)) \quad (10)$$

Overall PERFORMANCE Predictor ($OPERA_k(C)$): This metric combines quality metrics (weighted quality) with quantitative measures (coverage) to account for the size of the knowledge bases and the amount of lexical matches between the corpus and the KB. It also uses $d_k(C)$ as a factor indicating how successful the KB was in disambiguating the automatically detected mentions.

$$OPERA_k(C) = WQ_k(C) \times d_k(C) \times V_k(C) \quad (11)$$

For comparison, we defined two baseline metrics. The first metric uses the average TF-IDF score ($S_k(C)$). The second baseline metric is $S_k(C) \times d_k(C) \times V_k(C)$, which takes into account the coverage of the knowledge base to combine both basic quality and quantity factors.

3 Experiments

We applied our unsupervised named entity recognition method to extract named entities from 4 open-domain corpora and 5 biomedical corpora using 3 open-domain knowledge bases and 3 biomedical knowledge bases. The corpora used in the experiments are described below.

- **TREC** corpus [21] consists of sentences extracted from TREC documents. Similar to the CoNLL03 corpus, this corpus includes the following entity types: PERSON, ORGANIZATION, LOCATION, and OTHER.
- **NYT** corpus [14] consists of 8,000 named entities (PERSON, ORGANIZATION, LOCATION) from a random subset of New York Times articles (1998-2000) in the TREC corpus [26]. The documents were pre-annotated with a named entity tagger and then manually corrected by two annotators.
- **WikiNER** corpus [1] was created by manual annotation of the body text of 145 Wikipedia articles describing various named entities, with a roughly equal proportion of article topics from each of the four CoNLL03 entity types. Initial annotation was performed using a fine-grained inventory of 96 entity types (e.g., CITY, COMPANY) which were then mapped to CoNLL03 classes. Three annotators were involved in the annotation task and inter-annotator agreement was measured on a portion of the corpus.
- **CoNLL03** named entity corpus [23] consists of English and German documents. The English portion of the corpus, used in our experiments, is taken from the Reuters Corpus⁵ and consists of news stories from August 1996 to August 1997. The corpus was manually annotated, mostly following the MUC guidelines. In addition to MUC named entity types (PERSON, ORGANIZATION, LOCATION), an additional category (MISC) was also annotated. The annotated entities are non-overlapping and non-nested.
- **AZDC** (Arizona Disease Corpus) [12] consists of 2,783 sentences from 793 PubMed abstracts annotated with disease mentions. One annotator performed the annotation. A textual mention was annotated if it could be mapped to a unique concept with a relevant semantic type (e.g., *Disease of Syndrome, Neoplastic Process*) in the UMLS Metathesaurus [3]. Acronyms and negated/hedged mentions were annotated, while symptoms, general disease classes (e.g., *infection*) and overlapping mentions were ignored.
- **i2b2** corpus [32] consists of discharge summaries contributed by Partners Healthcare, Beth Israel Deaconess Medical Center, and the University of Pittsburgh Medical Center as well as progress reports from University of Pittsburgh Medical Center. The named

⁵ <http://trec.nist.gov/data/reuters/reuters.html>

entity annotation was performed manually and focuses on three categories: PROBLEM, TEST, TREATMENT. All documents were de-identified.

- **NCBI disease corpus** [6] consists of 793 PubMed abstracts also used in AZDC; however, in this corpus, all sentences in these abstracts were annotated. Pre-annotations from an automatic classifier were used as the basis of annotation. The annotations guidelines were similar to those of AZDC. Nested and non-continuous mentions were not annotated.
- **CDR corpus** [29] consists of 1500 PubMed abstracts discussing chemical-induced diseases and side effects, annotated for DISEASE and CHEMICAL categories. The corpus was manually annotated by the CTD (Comparative Toxicogenomics Database)⁶ staff.
- **Berkeley04 corpus** [20] consists of the first 100 titles and the first 40 abstracts from 59 MEDLINE 2001 data files. No keywords were used to retrieve the documents. Named entities of PROBLEM and TREATMENT categories were annotated by a single annotator.

In the open-domain experiments, we considered only PERSON, ORGANIZATION, and LOCATION as semantic types and discarded MISC and OTHER, as they are too ambiguous from a knowledge-base perspective and strongly biased according to different corpora.

We tested our approach in both the open-domain and biomedical domain using 6 knowledge bases: DBpedia, Yago, OpenCyc, UMLS, Snomed-CT and MeSH, described below.

- **DBpedia**[13] is a community-curated RDF knowledge base constructed semi-automatically from Wikipedia. Each Wikipedia article is interpreted as an entity in DBpedia and articles' infoboxes are used to extract automatically raw RDF triples, using the article entity as subject, the first column of the infobox as predicate and the second column as object. Manual mappings are then performed by the DBpedia community to reconcile the predicate names with the RDF properties in the reference DBpedia schema. DBpedia entities are also linked to other datasets in the Linked Open Data cloud⁷ such as YAGO or Freebase. In the scope of our experiments we used the English DBpedia 2014 version. After indexing, the DBpedia database consisted of 6,921,894 entities, 25,864,784 relations and 12,782,266 terms.
- **YAGO3** [16] is a large open-domain knowledge base built from Wikipedia, WordNet and GeoNames. It describes more than 10 million entities described by more than 120 million facts. However, most facts are type statements and *RDFS subClassOf* links. After indexing the Yago3 database consisted of 19,081,230 terms, 5,216,294 relations and 5,327,864 entities.
- **OpenCyc**⁸ is an open-domain knowledge base. It is a freely available version of the Cyc database. The 4.0 release of OpenCyc used in our experiments includes 800K terms as lexical descriptions of 240K concepts. Overall, the knowledge base contains about 1 million triples.
- **UMLS**[3] Metathesaurus 2015-AA consists of more than 100 biomedical vocabularies and contains more than 800K biomedical concepts with millions of relations between them. These relations are mostly lexical relations (e.g., synonymy, meronymy) and not domain relations⁹. Each concept in the UMLS Metathesaurus is associated with a set of semantic types from the UMLS semantic network. In the scope of our approach, we need to have domain

relations; we consider the semantic network classes (e.g. *Disease or Syndrome, Drug*) as entities instead of concepts, and the semantic network relations (e.g., *causes, treats*) as potential facts between the concepts. By extending the potential relations using the classes hierarchy (e.g., considering the potential link *type 2 diabetes, treats, antibiotics* from the general link *Drug, treats, Disease or Syndrome*, we obtain a set of 2,408 potential links that we use as knowledge base relations. After indexing, the UMLS database consists of 133 entities, 2,408 relations and 3,693,095 terms.

- **MeSH**¹⁰ (Medical Subject Headings) is a hierarchically-organized terminology designed mainly for indexing biomedical information. We use the 2016 RDF version of MeSH¹¹ as a knowledge base for biomedical entities. After indexing, the MeSH database contains 792,775 terms for 348,278 concepts, and 953,640 relations.
- **Snomed-CT**¹² (Systematized Nomenclature of Medicine – Clinical Terms) is a standardized clinical vocabulary used by health professionals for the exchange of clinical health information. It encompasses 806,831 terms, 421,308 concept and 1,836,908 relations.

The statistics from each knowledge base are presented in Table 1¹³.

We used relaxed position-based matching to evaluate the performance of our unsupervised NER method. Different corpora often adopt different criteria for named entity boundaries; for example, some may include adjectives and determiners, while others ignore them. These variations make exact named entity boundaries unsuitable for correlation studies. Table 2 presents the precision, recall and F1 score based on relaxed position-matching (values for exact matching F1 scores are 5% to 21% lower for individual corpora).

DBpedia outperformed the other open-domain knowledge bases on all open-domain corpora. This can be explained by the fact that DBpedia benefits from Wikipedia disambiguation pages and redirections, and consequently has a richer set of domain relations than YAGO and richer lexicalization than OpenCyc.

On the biomedical corpora, UMLS obtained the best recall but lower precision than DBpedia. The fact that UMLS has a better recall was expected due to its broader coverage for medical terms. The fact that DBpedia obtained better precision than UMLS can be explained by the fact that DBpedia has lower ambiguity for medical terms, which enhances the quality of both TF-IDF based search and relational disambiguation. Aside from these two general behaviours, there are no noticeable regularities where some KBs do consistently better than others on different corpora.

We study the correlation between the proposed unsupervised evaluation metrics and the standard performance measures for NER, namely, Precision, Recall and F1 score. We use the Pearson's correlation factor, ρ , to study the correlation between the metrics and the performance measures. More precisely, $\rho_C(\mu, \alpha)$ is expressed as:

$$\frac{N \sum_i \mu_i(C) \times \alpha_i(C) - (\sum_i \mu_i(C) \sum_i \alpha_i(C))}{\sqrt{N \sum_i \mu_i(C)^2 - (\sum_i \mu_i(C))^2} \sqrt{N \sum_i \alpha_i(C)^2 - (\sum_i \alpha_i(C))^2}} \quad (12)$$

¹⁰ <https://www.nlm.nih.gov/mesh/>

¹¹ <https://id.nlm.nih.gov/mesh/>

¹² https://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html

¹³ The number of relations here is considered to be the number of distinct subject-object pairs

⁶ <http://ctdbase.org>

⁷ <http://linkeddata.org>

⁸ <http://sw.opencyc.org/>

⁹ <http://www.ncbi.nlm.nih.gov/books/NBK9684/#ch02.sec2.4>

KB	Domain	Entities	Relations	Labels	Density
DBpedia2014 [13]	Open	6,921,894	25,864,784	12,782,266	$5.39 \cdot 10^{-7}$
YAGO3 [16]	Open	5,327,864	5,216,294	19,081,230	$1.83 \cdot 10^{-7}$
OpenCyc ¹⁴	Open	238,443	754,792	829,203	$1.32 \cdot 10^{-5}$
UMLS Lite [3]	Biomedical	133	2,408	3,693,095	0.13
Snomed CT ¹⁵	Biomedical	421,308	1,836,908	806,831	$1.03 \cdot 10^{-5}$
MeSH ¹⁶	Biomedical	348,278	953,640	792,775	$7.86 \cdot 10^{-6}$

Table 1. Knowledge Bases

Corpus	DBpedia			YAGO			OpenCyc			UMLS			MeSH			Snomed-CT		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TREC	82.9	49.0	61.6	80.3	29.9	43.5	66.7	30.3	41.6	-	-	-	-	-	-	-	-	-
NYT	69.7	42.8	53.1	68.5	27.9	39.7	48.9	20.5	28.9	-	-	-	-	-	-	-	-	-
WikiNER	82.1	49.0	61.4	81.3	31.7	45.6	63.2	26.5	37.3	-	-	-	-	-	-	-	-	-
CoNLL 03	69.1	49.3	57.6	65.7	31.4	42.5	48.5	31.4	38.1	-	-	-	-	-	-	-	-	-
AZDC	74.3	65.5	69.6	69.7	50.0	58.2	78.4	34.3	47.8	67.0	70.2	68.6	76.8	61.0	68.0	70.2	54.2	61.2
I2B2	65.3	31.5	42.5	56.8	16.0	24.9	51.1	23.7	32.3	60.5	35.8	45.0	74.7	20.6	32.3	53.6	15.3	23.8
NCBI	73.5	70.7	72.0	67.9	55.4	61.0	77.9	40.8	53.6	62.8	72.4	67.2	75.9	62.2	68.4	71.5	59.5	65.0
CDR	74.1	64.2	68.8	70.7	19.1	30.1	64.3	38.4	48.1	66.3	64.9	65.6	71.5	25.8	37.9	60.7	36.7	45.8
Berkeley04	71.3	65.1	68.1	64.5	38.1	47.9	59.3	39.9	47.7	62.6	69.8	66.0	78.6	50.2	61.3	67.0	39.9	50.0

Table 2. Precision (P), Recall (R) and F1 score for unsupervised NER. Best results are highlighted per row (corpus)

Where N is the number of knowledge bases, $\mu_i(C)$ represents the value of a performance measure (i.e., precision, recall, F1) when using knowledge base k_i to extract named entities from corpus C . $\alpha_i(C)$ represents the value of an annotation metric, α (e.g., graph adequacy, lexical adequacy) when knowledge base k_i is used to extract named entities from corpus C .

The values of the Pearson factor range from -1 (perfect negative correlation) to +1 (perfect positive correlation), with 0 indicating the absence of correlation. The Pearson factor is the most relevant for our study as it relies on the actual values of the variables, instead of their rank. A strong Pearson correlation would therefore suggest the portability of our method to additional corpora and knowledge bases. In contrast, the Spearman's rank correlation factor will not use the actual values of precision, recall and F1, but rather an integer rank value (e.g., a difference of 50% in F1 between 2 KBs could become equivalent to a difference of 1%), which does not allow assessing the scalability of the approach. Table 3 presents the Pearson correlation values for each metric.

4 Findings and Discussion

Our results show strong correlations between the proposed metrics and Recall, Precision and F1. Recall was naturally correlated with tokens coverage with 0.96 average Pearson factor on all corpora, 0.92 minimum correlation and 0.001 variance.

Precision was positively correlated with our Weighted Quality (WQ) metric with 0.59 average Pearson factor on all corpora, and

0.14 variance. One outlier behavior was observed for 2 corpora out of 9 (NCBI Disease corpus and Arizona Disease corpus), where WQ was not correlated with precision. These two corpora use the same document set. The NCBI corpus annotation extends AZDC annotations by annotating all sentences. When we studied these two corpora more closely with an error analysis, we observed that a disease name is often annotated only once in an abstract even if it occurs multiple time, which leads to a random behaviour for precision. From this perspective, our method was able to detect the bias of the manual annotation. If these two corpora are not included, the correlation of WQ with precision reaches an average of 0.76 with a variance of 0.001.

F1 scores were strongly correlated with the *OPERA* metric ($WQ \times D \times V$) on all corpora: average correlation of 0.96, minimum 0.88 and variance of 0.001, which follows the observations on the elementary metrics; i.e., WQ is a strong predictor of precision and V is a strong predictor of recall.

Our evaluation metric outperformed the TF-IDF baselines by +0.09 Pearson value for F1 and recall and +0.21 Pearson value for precision. The TF-IDF baseline had also a high variance of 0.22 for an average correlation value of 0.41, which shows that the quality of lexical matches does not provide a reliable indicator of precision. Our Lexical Disambiguation Ratio (L) had the best correlation for precision (0.62) with a relatively low variance of 0.107. L indicates the difficulty of the disambiguation problem as it represents the number of non-ambiguous mentions. This also shows that ambiguity, which is derived from TF-IDF scores, is more useful than the raw TF-IDF

Correlation	Precision			Recall			F1		
	Average	Range	Variance	Average	Range	Variance	Average	Range	Variance
Baselines									
TF-IDF	+ 0.41	[-0.43, +0.86]	0.217	+ 0.180	[-0.12, +0.62]	0.060	+ 0.26	[-0.01, +0.60]	0.04
$TF - IDF \times V$	+ 0.32	[-0.71, +0.99]	0.409	+0.87	[+0.76, +0.94]	0.002	+ 0.87	[+0.76, +0.96]	0.004
Composite Metrics									
<i>OPERA</i> ($WQ \times D \times V$)	+ 0.27	[-0.52, +0.81]	0.196	<u>0.96</u>	[+0.90, +0.99]	<u>0.001</u>	+ 0.96	[+0.88, +0.99]	0.001
Weighted Quality (WQ)	<u>+ 0.59</u>	[+0.00, +0.99]	<u>0.146</u>	+ 0.22	[-0.23, +0.59]	0.063	+ 0.32	[-0.10, +0.75]	0.071
Disambiguation Coverage ($D \times V$)	- 0.08	[-0.73, +0.53]	0.170	+ 0.88	[+0.70, +0.95]	0.005	+ 0.80	[+0.53, +0.93]	0.014
Lexical Adequacy ($L \times V$)	+ 0.30	[-0.49, +0.85]	0.221	+0.94	[+0.81, +0.99]	0.004	<u>+ 0.94</u>	<u>[+0.79, +0.99]</u>	<u>0.004</u>
Graph Adequacy ($R \times V$)	-0.51	[-0.99, +0.56]	0.224	+ 0.23	[-0.79, +0.95]	0.31	+ 0.10	[-0.85, +0.90]	0.31
Elementary Metrics									
Tokens Coverage V	+ 0.06	[-0.80, +0.73]	0.29	+ 0.96	[+0.92, +0.99]	$7e^{-4}$	+ 0.90	[+0.78, +0.99]	0.005
Average Ambiguity	- 0.61	[-0.99, +0.58]	0.229	-0.02	[-0.66, +0.96]	0.227	- 0.13	[-0.85, +0.91]	0.23
Disambiguation Ratio (D)	-0.20	[-0.54, +0.26]	0.04	+ 0.65	[+0.31, +0.84]	0.02	+ 0.57	[+0.10, +0.84]	0.03
Lexical Ratio (L)	+0.62	[-0.05, +0.99]	0.107	+0.21	[-0.66, +0.95]	0.26	+ 0.31	[-0.55, +0.98]	0.23

Table 3. Range, Variance and Average Pearson correlation factors between annotation metrics and Precision, Recall and F1 score. Best results are highlighted, second best are underlined.

values.

Including the Disambiguation Ratio (d) in the formula of *OPERA* ($WQ \times d \times V$) led to a better correlation for F1 score. We also observe that Disambiguation Coverage ($d \times V$) is less correlated for recall than V (0.90 vs 0.80 Pearson values, respectively). From additional experiments, we also found that $WQ \times V$ had an average correlation of 0.88 only (compared to 0.96 with $WQ \times d \times V$). Therefore, we can conclude that the elementary metric d had a positive impact for the prediction of precision values.

Our study is not exhaustive with regards to the number of metrics that might be considered. However, our results show that the proposed, general annotation metrics can predict, to a large extent, the adequacy of a KB for annotating named entities in a given corpus.

We limited our named entity recognition approach to commonly used methods. TF-IDF scores and global coherence maximization with KB relations are used in many related studies, including supervised classification approaches [19, 7, 11]. Therefore, we think that our observations can benefit other named entity recognition methods, provided that they use these two general principles.

We have no evidence at the current stage that our evaluation metrics would be relevant for recognition methods that do not rely on global coherence and TF-IDF scores. This includes token classification methods such as conditional random fields, which rely primarily on annotated corpora. A potential future direction is to use our unsupervised annotations to provide KB-derived semantic features at token level to study the performance of supervised classifiers.

Another potential future direction is to extend our method to evaluation of training corpora for supervised classification. In this setting, a training corpus can be seen as a knowledge base where the manual annotations are knowledge base entities and the co-occurrences of two annotations in the same sentence or context indicate the knowledge base relations.

5 Conclusions

We presented a new ranking approach to assess the suitability of knowledge bases for named entity recognition in a given corpus. More precisely, we proposed several unsupervised annotation metrics and studied their correlation with performance measures such as precision, recall and F1 score. Our results show that these metrics can be strong predictors of NER performance and that they significantly improve ranking relevance when compared to TF-IDF baselines. With the important increase in scope of named entities, our approach can play a key role in large-scale NER as it allows selecting relevant knowledge bases for a given textual context. It can also be applied to the selection of sub-graphs from the same (large) knowledge base. Our short-term goal is to deploy a web service that takes natural language texts as input and ranks all indexed knowledge bases according to the performance predictors proposed in this paper. This also includes the integration of other KBs deemed to be of sufficient interest. We also plan to study the performance of supervised classifiers according to these metrics when they use unsupervised knowledge base annotations as training features.

Acknowledgements

This work was supported by the intramural research program at the U.S. National Library of Medicine, National Institutes of Health.

REFERENCES

- [1] Dominic Balasuriya, Nicky Ringland, Joel Nothman, Tara Murphy, and James R Curran, 'Named entity recognition in wikipedia', in *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pp. 10–18. Association for Computational Linguistics, (2009).
- [2] Krisztian Balog, Edgar Meij, and Maarten De Rijke, 'Entity search: building bridges between two worlds', in *Proceedings of the 3rd International Semantic Search Workshop*, p. 9. ACM, (2010).
- [3] Olivier Bodenreider, 'The unified medical language system (umls): integrating biomedical terminology', *Nucleic acids research*, **32**(suppl 1), D267–D270, (2004).
- [4] Janez Brank, Marko Grobelnik, and Dunja Mladenic, 'A survey of ontology evaluation techniques', in *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)*, pp. 166–170, (2005).
- [5] Marc Bron, Krisztian Balog, and Maarten De Rijke, 'Example based entity search in the web of data', in *Advances in Information Retrieval*, 392–403, Springer, (2013).
- [6] Rezarta Islamaj Doğan and Zhiyong Lu, 'An improved corpus of disease mentions in pubmed citations', in *Proceedings of the 2012 workshop on biomedical natural language processing*, pp. 91–99. Association for Computational Linguistics, (2012).
- [7] Paolo Ferragina and Ugo Scaiella, 'Tagme: on-the-fly annotation of short text fragments (by wikipedia entities)', in *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 1625–1628. ACM, (2010).
- [8] Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, and Jos Lehmann, 'A theoretical framework for ontology evaluation and validation', in *SWAP*, volume 166. Citeseer, (2005).
- [9] Asunción Gómez-Pérez, 'Evaluation of ontologies', *International Journal of intelligent systems*, **16**(3), 391–409, (2001).
- [10] Younggyun Hahm, Jungyeul Park, Kyungtae Lim, Youngsik Kim, Dosam Hwang, and Key-Sun Choi, 'Named entity corpus construction using wikipedia and dbpedia ontology', in *LREC*, pp. 2565–2569, (2014).
- [11] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti, 'Collective annotation of wikipedia entities in web text', in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 457–466. ACM, (2009).
- [12] Robert Leaman, Christopher Miller, and G Gonzalez, 'Enabling recognition of diseases in biomedical text with machine learning: corpus and benchmark', in *Proceedings of the 2009 Symposium on Languages in Biology and Medicine*, volume 82, (2009).
- [13] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al., 'Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia', *Semantic Web Journal*, **5**, 1–29, (2014).
- [14] Xin Li, Paul Morie, and Dan Roth, 'Identification and tracing of ambiguous names: Discriminative and generative approaches', in *Proceedings of the National Conference on Artificial Intelligence*, pp. 419–424. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, (2004).
- [15] Adolfo Lozano-Tello and Asunción Gómez-Pérez, 'Ontometric: A method to choose the appropriate ontology', *Journal of database management*, **2**(15), 1–18, (2004).
- [16] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek, 'Yago3: A knowledge base from multilingual wikipedias', in *7th Biennial Conference on Innovative Data Systems Research*. CIDR Conference, (2014).
- [17] Yassine Mrabet, Claire Gardent, Muriel Foulonneau, Elena Simperl, and Eric Ras, 'Towards knowledge-driven annotation', in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pp. 2425–2431, (2015).
- [18] Fabian Neuhaus, Amanda Vizedom, Ken Baclawski, Mike Bennett, Mike Dean, Michael Denny, Michael Grüninger, Ali Hashemi, Terry Longstreth, Leo Obrst, et al., 'Towards ontology evaluation across the life cycle', *Applied Ontology*, **8**(3), 179–194, (2013).
- [19] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson, 'Local and global algorithms for disambiguation to wikipedia', in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 1375–1384. Association for Computational Linguistics, (2011).
- [20] Barbara Rosario and Marti A Hearst, 'Classifying semantic relations in bioscience texts', in *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, p. 430. Association for Computational Linguistics, (2004).
- [21] Dan Roth and Wen-tau Yih, 'Global inference for entity and relation identification via a linear programming formulation', *Introduction to statistical relational learning*, 553–580, (2007).
- [22] Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum, 'Large-scale cross-document coreference using distributed inference and hierarchical models', in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 793–803. Association for Computational Linguistics, (2011).
- [23] Erik F. Tjong Kim Sang and Fien De Meulder, 'Introduction to the conll-2003 shared task: Language-independent named entity recognition', in *Proceedings of CoNLL-2003*, eds., Walter Daelemans and Miles Osborne, pp. 142–147. Edmonton, Canada, (2003).
- [24] Maksim Tkachenko and Andrey Simanovsky, 'Named entity recognition: Exploring features.', in *KONVENS*, pp. 118–127, (2012).
- [25] Felix Tristram, Sebastian Walter, Philipp Cimiano, and Christina Unger, 'Weasel: a machine learning based approach to entity linking combining different features', in *Proceedings of 3th International Workshop on NLP and DBpedia, co-located with the 14th International Semantic Web Conference (ISWC 2015), October 11-15, USA*, (2015).
- [26] Ellen M. Voorhees, 'Overview of the TREC 2002 question answering track', in *Proceedings of The Eleventh Text REtrieval Conference, TREC 2002, Gaithersburg, Maryland, USA, November 19-22, 2002*, eds., Ellen M. Voorhees and Lori P. Buckland, volume Special Publication 500-251. National Institute of Standards and Technology (NIST), (2002).
- [27] Denny Vrandečić, *Ontology evaluation*, Springer, 2009.
- [28] Cristofer Weber and Renata Vieira, 'Building a corpus for named entity recognition using portuguese wikipedia and dbpedia', in *I Workshop on Tools and Resources for Automatically Processing Portuguese and Spanish*, pp. 9–15, (2014).
- [29] Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Jiao Li, Thomas C. Wieggers, and Zhiyong Lu, 'Assessing the state of the art in biomedical relation extraction: overview of the BioCreative V chemical-disease relation (CDR) task', *Database*, **2016**, (2016).
- [30] Christopher A Welty, Ruchi Mahindru, and Jennifer Chu-Carroll, 'Evaluating ontological analysis', in *Semantic Integration Workshop (SI-2003)*, p. 92. Citeseer, (2003).
- [31] Haining Yao, Anthony Mark Orme, and Letha Eitzkorn, 'Cohesion metrics for ontology design and application', *Journal of Computer science*, **1**(1), 107–113, (2005).
- [32] Uzuner, South BR, Shen S, and DuVall SL, '2010 i2b2/va challenge on concepts, assertions, and relations in clinical text', in *Journal of the American Medical Informatics Association: JAMIA*, volume 18, pp. 552–556, (2011).

Skeleton-Based Orienteering for Level Set Estimation

Lorenzo Bottarelli¹ and Manuele Bicego¹ and Jason Blum² and Alessandro Farinelli¹

Abstract. In recent years, the use of unmanned vehicles for monitoring spatial environmental phenomena has gained increasing attention. Within this context, an interesting problem is *level set estimation*, where the goal is to identify regions of space where the analyzed phenomena (for example the PH value in a body of water) is above or below a given threshold level. Typically, in the literature this problem is approached with techniques which search for the most interesting sampling locations to collect the desired information (i.e., locations where we can gain the most information by sampling). However, the common assumption underlying this class of approaches is that acquiring a sample is expensive (e.g., in terms of consumed energy and time). In this paper, we take a different perspective on the same problem by considering the case where a mobile vehicle can continuously acquire measurements with a negligible cost, through high rate sampling sensors. In this scenario, it is crucial to reduce the path length that the mobile platform executes to collect the data. To address this issue, we propose a novel algorithm, called *Skeleton-Based Orienteering for Level Set Estimation* (SBOLSE). Our approach starts from the LSE formulation introduced in [10] and formulates the level set estimation problem as an orienteering problem. This allows one to determine informative locations while considering the length of the path. To combat the complexity associated with the orienteering approach, we propose a heuristic approach based on the concept of topological skeletonization. We evaluate our algorithm by comparing it with the state of the art approaches (i.e., LSE and LSE-batch) both on a real world dataset extracted from mobile platforms and on a synthetic dataset extracted from CO2 maps. Results show that our approach achieves a near optimal classification accuracy while significantly reducing the travel distance (up to 70% w.r.t LSE and 30% w.r.t LSE-batch).

1 INTRODUCTION

The goal of environmental analysis is to collect information, generating an accurate model for a specific environmental process. For example when monitoring the quality of a body of water, operators might be interested in modeling how crucial parameters such as PH level, Dissolved Oxygen and temperature vary across time and space. These analyses usually require the collection of large data sets in harsh conditions, hence the use of mobile sensors such as unmanned ground vehicles (UGVs), unmanned aerial vehicles (UAVs) or autonomous surface vessels (ASVs). For an exhaustive overview on advancements and applications see [7].

A successful monitoring operation must acquire a sufficient amount of data to build an accurate model of the environmental phe-

nomena of interest. However, the data collection process must consider limited resources such as energy and time. Therefore, it is crucial to carefully select measurement locations to acquire as much information as possible while minimizing energy and time required for data collection. An important aspect to consider is that the choice of the future locations to visit is dependent on previously collected data. Traditional, off-line sampling methods [6] do not represent a proper choice in this context – we refer to these processes as passive learning methods. Krause and Guestrin [14] survey advances to efficiently evaluate observation selection and illustrate the effectiveness of the approaches on environmental phenomena monitoring.

In contrast, active learning techniques [1, 16] aim to incrementally build a model of the environmental process during the data collection phase. Such techniques are very well suited for guiding mobile sensors and can be used to focus the data collection process on specific regions of the environment, so as to minimize the energy required for navigation [18].

In the simplest setting, the analysis process aims at collecting uniformly distributed data over a selected area. However, in many scientific and environmental monitoring applications, we are not interested in the precise value of the phenomena in every single location, rather we are interested in locating the regions of the space where the measurements exceed a given threshold level. This problem is typically referred to as the “level set estimation problem” [11]. For example, monitoring the water in a lake may require identification of the regions where the PH level of the water exceeds a critical value or to detect contours of biological and chemical plumes. Previous work on the level set estimation problem such as [5] focuses on a network composed by a combination of static and mobile sensors, while [20] focused on controlling the movement and communication of a sensor network without giving much attention to the choice of the sampling locations.



Figure 1. Platypus Lutra-T

In a more recent work on level set estimation [10] the data collection task is formalized as a classification problem with sequen-

¹ Computer Science Department, University of Verona, Italy
name.surname@univr.it

² Robotics Institute, Carnegie Mellon University and Platypus LLC, Pittsburgh USA
jasonblu@andrew.cmu.edu

tial measurements; the proposed LSE algorithm uses Gaussian Processes (GP) [17] to estimate sampling points that reduce uncertainty around a given threshold level of the modeled function. The authors show a near-optimal classification for every region of the space with a reduced number of sampled locations compared to previous approaches. However, in the standard algorithm they do not explicitly take into account the path between the sampling locations, but they simply choose as the next point to be visited *the most informative* point, according to an ambiguity measure they derive from the Gaussian Process. They discuss the possibility to reduce the path length of the mobile sensor by proposing a *batch* version of their algorithm, where they determine a set of new sampling locations, again according to the ambiguity measure derived from the Gaussian Process. Even if an efficient path between these points can be computed, again the choice of such points does not consider at all the distance the mobile sensor must cover to visit all such locations. Finally, more recently, [11] proposed a new receding horizon approach built on the LSE algorithm. Their method is designed for ASVs equipped with a probe that allows an aquatic sensor to be lowered into the water, and the algorithm uses a path planner to select sampling locations that lie on a feasible path for the probe within a predefined vertical transect plane.

In this paper, we also address the problem of level set estimation by using active learning techniques with sequential measurements. However, in our case, we have a further objective, namely we aim also at determining efficient paths for mobile sensors (instead of determining single sampling locations) so to optimize the data collection process. Specifically our techniques are motivated by the recent development of low-cost, small mobile platforms that can perform continuous-sampling in various body of waters (lakes, rivers and coastal areas). For example, consider the autonomous surface vessel shown in Figure 1. This platform is small (about 1 meter long and 50 cm wide) and it is equipped with various probes that can measure parameters such as PH, Dissolved Oxygen, temperature, and electrical conductivity with sampling rate between 1 and 10 Hz. In this setting the cost to perform a single measurement is negligible, and the most crucial issue for the data collection process is the battery lifetime for the vessel. In fact, to meet the space constraint of this platform, batteries have a limited capacity that results in constraints on total path length. Hence, in this work we aim at optimizing the total path length required by the agent to achieve near optimal classification of the analyzed regions, rather than the number of samples extracted during the executions (which is an important criteria for previous works).

In this perspective, the approach we propose formulates the Level Set Estimation problem as an Orienteering Problem (OP) [22]. In the general formulation of the OP we start with a set of locations, each one associated to a given score, and the goal is to choose the locations to visit so to maximize the sum of the scores while keeping the time (or the distance travelled) below a given budget. In the level set estimation problem we can see the sampling candidates as the locations to be visited, each one equipped with an *informativeness* score. For example we can use the already introduced ambiguity [10] to measure the value of the points. In this case, the LSE solution introduced in [10] simply chooses the most ambiguous point as the next point to be visited. The batch variant simply selects few points, again without considering the path. In contrast, by solving the OP in this setting, we are now trying to maximize the total informativeness of the points visited while keeping the travelled distance below a given budget, i.e. while explicitly considering the *cost* of the exploration (i.e., the length of the path). The OP is known to be NP-Hard. While we can use heuristic approaches to solve the problem, to perform

on-line exploration we need to reduce the size of the orienteering instance (i.e. the number of possible locations to be visited). To this end, we propose a heuristic based on topological skeletonization, a process introduced in the image processing community [9] aimed at reducing regions in a binary image to a thin (skeletal) representation — the *skeleton*, sometimes also called *medial axis*. In particular, we approximate the regions of the possible points to be visited (i.e. the unclassified points) with their skeleton, thus drastically reducing the size of the orienteering instance. As we will show in our empirical evaluation, this heuristic does not significantly affect the accuracy of the classification.

As a final comment, it is important to note that a related approach has been proposed in [19], where an orienteering-inspired technique has been applied to a related but different problem concerning information gathering. However, there are several important differences with respect to our work: i) the technique introduced in [19] does not aim at solving the level set estimation problem; ii) they propose an algorithm to solve the *submodular orienteering* problem (a particular kind of OP introduced by Chekuri and Pal [3]); iii) finally, our main objective is to determine efficient paths for mobile sensors (instead of determining single sampling locations) so to optimize the data collection phase and reduce the energy required in this process.

The main contributions of this paper to the state of the art are:

- We propose a novel algorithm that uses an orienteering formulation to solve the level set estimation problem.
- We propose a topological skeletonization as a heuristic to reduce the number of points on which we apply the orienteering algorithm.
- We empirically evaluated our algorithm comparing it to the current state of the art approach (i.e., LSE and LSE-batch [10]). Specifically, we consider a real-world dataset composed of measurements of the PH level of the water acquired with our mobile watercraft, and a synthetic dataset based on publicly available CO2 maps. Results show an advantage in terms of total travel distance, hence proving the feasibility of a skeleton-based orienteering approach to solve the level set estimation problem.

2 PROBLEM STATEMENT AND BACKGROUND

In the same spirit of [10], we formalize our approach for the level set estimation problem as an active learning problem, where we want to select next measurement locations so to optimize the information gathering process.

The environmental phenomena of interest is represented by an unknown scalar field. The area of the environment is discretised in a matrix where each element represents a location with an associated scalar value. For example, in practical application each element could be associated to a squared meter of the environment’s surface and each element represents a sampling location x_i that must be classified according to a threshold level.

Specifically, given a threshold level h and a set of locations $D \subseteq \mathbb{R}^d$, we want to infer knowledge about the unknown scalar field $f : \mathbb{R}^d \mapsto \mathbb{R}$ and then to classify all points $x \in D$ into either $H = \{x \mid f(x) > h\}$ (called superlevel set) or $L = \{x \mid f(x) \leq h\}$ (called sublevel set). The scalar field is modeled with a Gaussian Process (GP) [17]. The problem then is to select the best set of locations x_i where to perform (noisy) measurements $y_i = f(x_i) + e_i$ while optimizing the total path length required for the mobile agents to analyze these points. Our proposed approach faces this problem using an Orienteering-based approach. Since the Orienteering problem

is computationally heavy, reducing the number of candidate points to be considered is crucial: in our approach this is done by exploiting a skeletonization-based heuristic. In the remainder of this section we will summarize the needed background: the Gaussian Processes, the formulation of the Level Set problem with Gaussian Processes – together with the solutions proposed in [10]–, the Orienteering problem and the Topological skeletonization.

2.1 Gaussian Processes

Gaussian Processes are a very important and widely used tool in machine learning [17]. In probability theory, a Gaussian Process (GP) is a statistical distribution and offers a way to model an unknown function without using parameters. In our case the function to be modeled is the scalar field f . A GP is completely defined by its mean function $\mu(x)$ that formulates prior knowledge about the values of the function f^3 , and its covariance function (also called kernel function) $k(x, x')$ which encodes the smoothness properties of the function samples. A GP can then be denoted as $\mathcal{GP}(\mu(x), k(x, x'))$. At a given time t , we consider a set of noisy measurements $Y_t = \{y_1, y_2, \dots, y_t\}$ taken at locations $X_t = \{x_1, x_2, \dots, x_t\}$. We assume that $y_i = f(x_i) + e_i$ where $e_i \sim \mathcal{N}(0, \sigma_n^2)$ (i.e., measurements noise with zero mean) and we consider a GP prior $\mathcal{GP}(0, k(x, x'))$. Under these assumptions, the posterior over f is still a GP and its mean and variance can be computed as follows [17]:

$$\mu_t(x) = \mathbf{k}_t(x)^T (\mathbf{K}_t + \sigma_n^2 \mathbf{I})^{-1} Y_t \quad (1)$$

$$\sigma_t^2(x) = k(x, x) - \mathbf{k}_t(x)^T (\mathbf{K}_t + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_t(x) \quad (2)$$

where $\mathbf{k}_t(x) = [k(x_1, x), \dots, k(x_t, x)]^T$ and $\mathbf{K}_t = [k(x, x')]_{x, x' \in X_t}$

Using the above equations, we can update the GP with the new knowledge acquired through the observations (i.e., the set of measurements).

2.2 Level Set Estimation using Gaussian Processes

The formulation of the level set estimation problem using Gaussian Processes has been introduced in [10]. We have a set of sample locations D (that represents our area of interest) and we want to classify each location $x_i \in D$ into the two sets H or L previously defined by a threshold level h . This formulation uses the inferred mean (1) and variance (2) from the GP learnt on the scalar field to construct an interval:

$$Q_t(x) = \left[\mu_{t-1}(x) \pm \beta_t^{1/2} \sigma_{t-1}(x) \right] \quad (3)$$

for any point $x \in D$, where the parameter β acts as a scaling factor for the interval. The algorithm uses the intersection of all previous intervals to define a confidence interval

$$C_t(x) = \bigcap_{i=1}^t Q_i(x) \quad (4)$$

The classification of a point x into H or L depends on the position of its confidence interval with respect to the threshold level h . Intuitively if the entire interval lies above h , then with high probability $f(x) > h$ and x should belong to H . Similarly if the entire interval lies below h then x should belong to the L set. These conditions are relaxed by introducing an accuracy parameter ϵ which acts as a

trade-off parameter between classification accuracy and number of samples required. Specifically:

$$H_t = \{x \mid \min(C_t(x)) + \epsilon > h\} \quad (5)$$

$$L_t = \{x \mid \max(C_t(x)) - \epsilon \leq h\} \quad (6)$$

This confidence interval allows the algorithm to either classify a point into the superlevel or the sublevel set. However, this classification scheme does not permit classifying all locations $x_i \in D$ at time t when $|Y_t| \ll |D|$, leaving a set of unclassified locations:

$$U_t = D \setminus (L_t \cup H_t) \quad (7)$$

Hence for the points that belong to U_t , we have to defer the decision until enough information is available. Given this formulation the goal is to select new sampling locations $x_i \in U_t$ to acquire new data and classify the points in U_t according to the equations (5) and (6).

2.2.1 The solutions of [10]: the LSE algorithm and the LSE batch algorithm

[10] proposed two solutions to this problem, both based on the concept of *ambiguity* of the candidate points. In particular, at a given iteration, the algorithm exploits the confidence intervals $C_t(x)$ derived from the Gaussian Process to calculate the ambiguity of all unclassified points:

$$a_t(x) = \min\{\max(C_t(x)) - h, h - \min(C_t(x))\} \quad (8)$$

Given this concept, two solutions are presented in [10] to select the set of next points to be sampled:

1. *LSE*: in this case, the set of the next points to be sampled is composed by only one point, in particular the one with the highest ambiguity. Clearly, this solution does not take into consideration the distance of the chosen point from the current position, i.e. it does not consider the length of the path.
2. *LSE Batch*: this second algorithm is aimed at alleviating this problem, and opts for the selection of multiple locations to be sampled next. In particular, such locations are sequentially selected by considering both their ambiguity defined in eq. (8) and their joint mutual information, as derived from the Gaussian Process – for more information see [10] and [11]. An efficient path between such locations is then computed.

Although the main goal of the LSE Batch approach is to select multiple locations and to compute an efficient path between them, this algorithm is far in spirit from what we propose in this paper: actually, as in the LSE algorithm, the length of the path is not a variable explicitly considered in the choice of the points to be sampled next. The main reason for this is that in both cases their assumption is that the process of acquiring new data is costly. Therefore their main goal is to minimize the number of sampling locations.

2.3 Orienteering

The Orienteering Problem (OP) originates from the sport game of orienteering. In this game, the start and end points are specified along with other locations (i.e., checkpoints) which have associated score. The players aim at visiting as many checkpoints as possible in order to maximize the total score and have to reach the end point within a given time frame. The same problem can model several different contexts. For example, consider the problem in which a traveling salesperson has a set of cities which he could visit. Assuming that the

³ This can be assumed to be zero without loss of generality

salesperson knows the number of sales he/she can expect in each city, the goal is then to plan a route so as to maximize the total number of sales while keeping the total length of such route within a given budget (i.e., the maximum distance he/she can travel in one day).

More formally, the OP can be formulated in the following way: given a set of N locations each with a score $S_i \geq 0$, a starting location 1, an ending location N and the travel time t_{ij} for all couples of locations i and j (with $i \neq j$), the goal is to determine a path, limited by a given budget T_{max} , that visits a subset of these locations, in order to maximize the total collected score.

The OP can intuitively be defined with the aid of a weighted undirected graph $G = (V, E)$ where $V = \{v_1, \dots, v_N\}$ is the set of nodes (locations) and E is the set of edges. In this formulation the nonnegative score S_i of location i is associated with a vertex $v_i \in V$ and the travel time t_{ij} between location i and j is associated with each edge $e_{ij} \in E$. The OP consists of determining a Hamiltonian path over a subset of V , including the start node (v_1) and end node (v_N), and having a length not exceeding the bound T_{max} , in order to maximize the total collected score.

Therefore, the orienteering is a combination of node selection and shortest path computation between these nodes, hence it can be casted as a combination of the Knapsack Problem (KP) and the Traveling Salesman Problem (TSP) problems [4], where the KP goal is to maximize the total score collected while the TSP aims at minimizing the travel distance. This formulation is typically referred to as a generalized travelling salesman problem (GTSP) [8]. The orienteering problem is known to be an NP-hard problem, as it contains the well known traveling salesman problem as a special case.

This NP-hard problem arises in routing and scheduling applications and it is also known as the selective traveling salesperson problem ([15], [21]) or the maximum collection problem ([13]). A number of practical applications has been modeled as OP and many heuristic approaches have been developed to combat the inherent complexity of the OP. In most cases, the orienteering is defined as a path to be found between distinct locations, rather than a circuit where $v_1 \equiv v_N$. In some applications, however, v_1 can coincide with v_N but the difference between both formulations is not significant. For a general review we suggest the survey proposed by Vansteenkoven et al. [22].

2.4 Topological Skeletonization

In digital image processing and shape analysis, *skeletonization* is a process for reducing regions in a binary image to a thin (skeletal) representation while throwing away most of the original pixels (see example in Figure 2). The skeleton preserves and usually emphasizes the geometrical properties of the shape, such as its connectivity, topology, length and direction.

Skeletonization was first introduced by Blum [2], and it can be described by using an intuitive model of fire propagation. If one "sets fire" at all points on the boundary of an image the skeleton forms at the points in the region where two or more "fires" meet. This intuitive description has several different mathematical definitions and in the relevant literature it is sometimes referred to as *medial axis* or *thinning* [9].

Skeletonization has been used in several applications ranging from computer vision to image analysis and digital image processing. There are many algorithms that are tailored for different application contexts. Such approaches mainly vary in run time and properties of the produced skeleton (e.g., whether it is a connected component or not), however they all significantly compress the input. In this paper

we are interested in skeletonization mainly to reduce the number of points that we must consider when planning the route for the robotic platforms. Hence, we select a basic approach based on morphological operators (see Section 4.3)



Figure 2. Example of a topological skeletonization applied to an image. On top the binarized input image and on the bottom the skeletonized version.

3 SBOLSE ALGORITHM

Using both the LSE solutions proposed in [10], the mobile sensor is guided toward the most informative points, without taking into account the path of the mobile sensor. For example, the LSE algorithm assumes that the mobile sensor moves from the current position to the next selected location following a straight line. Another issue is that the measure is collected only at the final location, without considering all the points traversed by the sensor along its path. On the contrary, here we consider applications where measuring devices can provide data while the robotic platform is moving. For example, the mobile platforms we use here are equipped with probes that measure various parameters (e.g., the PH level) with a given frequency while the platform is moving. In this scenario, our goal is then minimizing the path length while collecting as much information as possible to correctly classify all points $x_i \in D$.

In what follows we present our Skeleton-Based Orienteering for Level Set Estimation (SBOLSE) algorithm, which starts from the LSE framework but is specifically designed for continuous sampling sensors in which the cost required to extract a sample is negligible but it is necessary to optimize the total path of the mobile platform to minimize battery consumption.

The proposed algorithm considers the knowledge about unclassified locations $x_i \in U_i$ to build an OP instance and to select a sequence of visit locations (i.e., a path). The algorithm aims at optimizing the information that can be acquired along the path while

meeting the budget on the travel distance. Next, we propose a heuristic approach based on the topological skeletonization to combat the computational complexity associated with the OP, empirically showing that the classification accuracy does not suffer a significant degradation while greatly reducing the computation time.

Algorithm 1 SBOLSE algorithm

Input: set D , threshold h , accuracy parameter ϵ , prior known data $X \subset D$, starting location x_{start}
Output: sets H and L

```

1:  $t \leftarrow 0$ 
2:  $x_0 \leftarrow x_{start}$ 
3:  $H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$ 
4: while  $H_t \cup L_t \neq D$  do
5:    $t \leftarrow t + 1$ 
6:   Compute GP posterior  $\mu(x)$  and  $\sigma^2(x)$  for all  $x \in U_t$ 
7:   Classify and update  $H_t, L_t, U_t$  according to LSE [10]
8:    $x_c \leftarrow$  current position
9:    $G \leftarrow buildGraph(x_c, U_t)$ 
10:   $path \leftarrow orienteeringStep(G, budget)$ 
11:  Execute  $path$ 
12:  $H \leftarrow H_t, L \leftarrow L_t$ 

```

The pseudo-code of Algorithm 1 describes the steps of our SBOLSE approach. The algorithm maintains three sets of points: the current superlevel H_t and sublevel L_t sets, as well as the set of unclassified points U_t . At each iteration t we update the Gaussian Process posterior by integrating the new information gathered at the preceding iteration (line 6). Then we compute the confidence intervals $C_t(x)$ for each point $x \in U_t(x)$, classify them in one of the three sets and then compute the sequence of locations to be visited. To compute such sequence of locations we consider the ambiguity defined by equation (8) of the unclassified points and build an OP instance. Specifically, in line 9 we create a graph from the unclassified points U_t (Algorithm 2) and then compute a path (line 10) using the orienteeringStep procedure (Algorithm 3). The algorithm terminates when $H_t \cup L_t = D$, i.e. when all points are classified and thus $U_t = \emptyset$. Note that during the execution of the path (Algorithm 1, line 11) if the platform moves over locations not yet analyzed but already classified according to LSE [10], these are evaluated and, in case, re-classified considering newly acquired data.

3.1 Building the graph

In the buildGraph procedure we take all the unclassified locations U_t and we build an un directed weighted graph, where all locations are connected. This graph will then be used in the orienteering procedure.

As shown in Algorithm 2, the first node of the graph represents the current location of the mobile sensor (line 1). This location defines the starting position for the orienteering solver. Subsequently we build the nodes set V and the edges set E . The function $w(\cdot)$ denotes respectively the weight of a node or the weight of an edge. The weight of a node $w(v_i)$ (line 7) is the ambiguity measure (equation 8) of the location that the node represents. The weight of the first node is an exception as this represents the current position of the mobile sensor and hence the location has been visited and classified. The weight of the edges $w(e_{ij})$ (line 13) is the travel distance between the locations represented by the nodes v_i and v_j .

Algorithm 2 buildGraph procedure

Input: current position x_c , unclassified elements U_t
Output: weighted undirected graph G

```

1:  $V \leftarrow v_1 \equiv x_c$ 
2:  $w(v_1) \leftarrow 0$ 
3:  $n \leftarrow 1$ 
4: for all  $x_i \in U_t$  do
5:    $n \leftarrow n + 1$ 
6:    $V \leftarrow V \cup v_n \equiv x_i$ 
7:    $w(v_n) \leftarrow a(x_i)$ 
8:  $E \leftarrow \emptyset$ 
9: for all  $v_i \in V$  do
10:  for all  $v_j \in V$  do
11:    if  $v_i \neq v_j$  then
12:       $E \leftarrow E \cup e_{ij}$ 
13:       $w(e_{ij}) \leftarrow dist(v_i, v_j)$ 
14:  $G \leftarrow (V, E)$ 

```

3.2 Orienteering Step

Algorithm 3 orienteeringStep procedure

Input: graph $G = (V, E)$, budget B
Output: $bestPath$

```

1:  $bestPath \leftarrow \emptyset$ 
2:  $bestPathValue \leftarrow 0$ 
3: for  $i$  in range(2,  $|V|$ ) do
4:   if  $dist(v_1, v_i) \leq budget$  then
5:      $path \leftarrow orienteeringHeuristic(G, v_1, v_i, B)$ 
6:     if  $value(path) > bestPathValue$  then
7:        $bestPath \leftarrow path$ 
8:        $bestPathValue \leftarrow value(path)$ 

```

In the orienteeringStep procedure we use the previously built undirected weighted graph G and consider this as the input to the orienteering problem. In particular we have a fixed starting point (i.e. the current location of the mobile agent), but we do not have an ending point (which is required in the classical formulation of the orienteering problem). It makes clearly sense that the starting point should be equal to the destination point, however in the classic orienteering problem the rewards of every node are fixed. In our case, rewards changes during the execution of the procedure since the information value of every location decreases while the robot acquires new data. Hence, making a single run of orienteering would not take into account the adaptivity required by such scenario. Therefore, we iterate the process for smaller segments. The choice of the length (budget) of these segments allows a tradeoff between adaptivity and computation requirements. To choose the destination we perform an orienteering heuristic multiple times (Algorithm 3, line 5), assuming as destination every unclassified location in the graph that is reachable with the given budget. Every time we solve an orienteering instance we obtain a new path. The procedure keeps track of the best discovered one and returns this as final route to be executed from the SBOLSE algorithm. Specifically with $value(path)$ (line 6 and 8) we indicate the summation of the nodes' weights in that route, that is $value(path) = \sum_{v_i \in path} w(v_i)$. Since the orienteering problem aims at maximizing the score within a given travel budget, using this procedure we obtain a path that maximizes the information selected among the unclassified locations for the level set estimation problem. In this work we did not focus on the computational efficiency

but rather on a novel formalization of the problem. The choice of having a completely connected graph and repeating the orienteering step n -times for every possible reachable destination represents the simplest choice for formalizing this problem. Improving these aspects to reduce the computation required is matter of future work. Current times do not prevent real-time operations.

3.3 Skeletonization

In most practical applications of level set estimation the input is a set of dense points that must be classified. Specifically, when we start the data acquisition process, we must consider the entire surface of the selected portion of the environment. These data are typically discretized and organized in a matrix where each entry represents a small portion of the surface (i.e., a square of 50 centimeters in our experiments).

Now, given some smoothness of the environmental phenomena, locations with higher classification uncertainty usually cluster into areas where the unknown scalar field has high probability to cross the threshold level. Considering all such points could be considered redundant. This motivates the use of a topological skeletonization algorithm to compress the input.

Specifically, we consider the matrix containing the information about the ambiguity measure (eq. 8) of the unclassified points U_t as a binary image, where unclassified points are set to 1 and classified points are 0. We then apply a topological skeletonization to such image, and we maintain as interesting points to be classified only the points of the skeleton. This greatly reduces the number of locations that we must consider in the *buildGraph* procedure presented in section 3.1 (see the example in Figure 3).

3.4 Theoretical analysis

For what concerns the theoretical analysis of our approach, notice that Gotovos et. al. with Theorem 1 in [10] prove the convergence of the LSE algorithm. Even though the selection procedure of our SBOLSE algorithm differs from LSE, we used the same classification rules (Algorithm 1, lines 6-7). As in LSE, our technique iterates until every point is classified with respect to a threshold level h and with an accuracy parameter ϵ , hence we can ensure the convergence of the SBOLSE algorithm.

4 EMPIRICAL EVALUATION

In this section we now present an empirical evaluation of our proposed technique. First (in sections 4.1 and 4.2) we present the comparison of our technique with state of the art approaches on two different datasets. Then, in section 4.3, we analyze the results of the topological skeletonization applied to the ambiguity measure (as explained in section 3.3), showing that this heuristic significantly reduces the size for the orienteering instances while preserving the overall classification accuracy.

Specifically we compare: i) Our SBOLSE technique with a variant of the LSE algorithm [10], which we designed to meet the continuous sampling setting; ii) The batch version of the two approaches tested in i). More specifically, the algorithms we compare are:

- **SBOLSE**: Our algorithm as explained in section 3.
- **CS**: This algorithm is a variant of the LSE as described in [10]. The classification and sample selection is the same except that all locations on the straight line between the last position and the

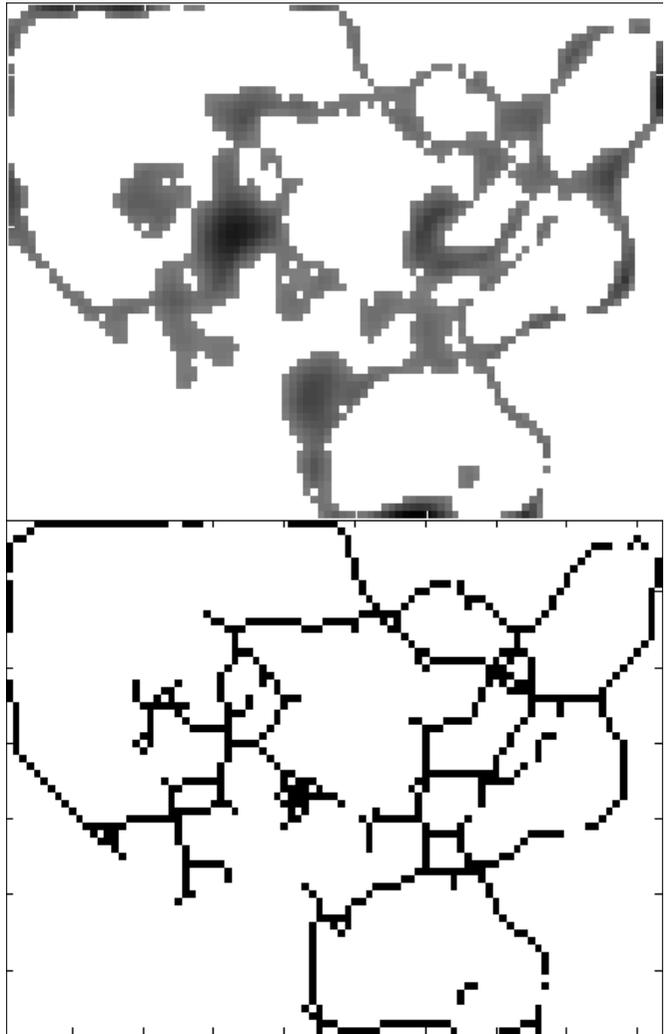


Figure 3. Example of the topological skeletonization applied to the data matrix containing the ambiguity measure for the unclassified points U_t . On top the data matrix before the skeletonization, a darker color corresponds to an higher value of ambiguity. On the bottom the skeletonized version.

next selected location are analyzed so to simulate a continuous sampling sensor.

- **CS _{$b \times X$}** : This algorithm is a variant of the LSE_{batch} as described in [10]. Similarly to CS we analyze all locations in the straight line between one location and the next one to simulate a continuous sampling sensor, XX identifies the dimension (i.e. the number of elements) of the batch. We performed tests with batches of different sizes and determined as optimal value a batch size of 30. We did not observe a significant reduction of the total path length with batches of size larger than 30.

Regarding our SBOLSE algorithm, we implemented a simple orienteering heuristic inspired by the *center of gravity* technique as proposed by Golden et al. [8]. We performed the skeletonization with a basic algorithm, based on morphological operators, as implemented in the MATLAB function `bwmorph`. We used the F_1 -score as in [10] to assess the classification accuracy for all the results of our tests. The F_1 -score is often used in information retrieval for measuring binary

classifications. In our case we consider the superlevel set as the positives and the sublevel set as the negatives elements.

4.1 Real data experiments

The first dataset consists of real-world data relative to the PH level extracted from waters of the Persian Gulf near Doha (Qatar) using the boat in Figure 1. The data collected has been aggregated in a 68×93 matrix where each element represents a sampling location x_i that must be classified according to a threshold level. In particular, each element represents 0.5 square meters of the analyzed surface. The value associated to that element is the average of all the samples extracted in that portion of the surface. An example of this dataset is shown in Figure 4.

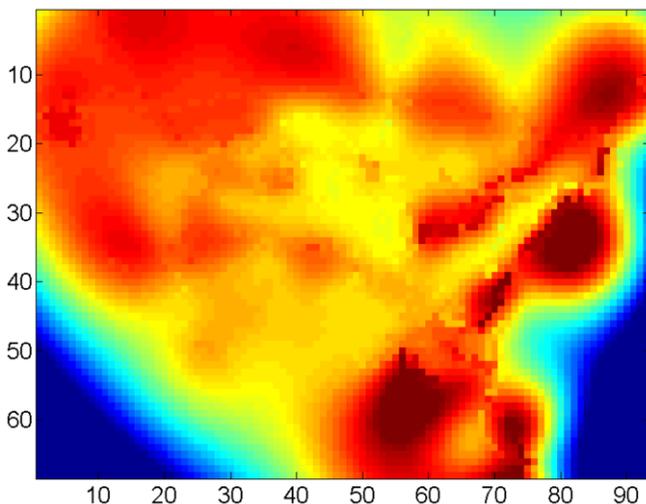


Figure 4. Scalar field of the real-world dataset, i.e. the PH level of waters extracted in the Persian Gulf near Doha (Qatar).

On this first set of experiments we considered three different thresholds to classify the PH scalar field, specifically 7.40, 7.42 and 7.44. We performed some test to determine the best parameter setting for β and ϵ in such a way that the classification accuracy is high for all the algorithms we compare.

Following standard approaches in the literature (e.g., [10]), we start from ten random initial prior composed by 10% of the elements of the matrix, for a total of 30 instances per algorithm. The priors were used to fit the hyperparameters of an isometric Matérn-3 covariance function. The results of this set of experiments are shown in Table 1. For a fair comparison the budget for the orienteering subroutine has been set to the same length that would have been traveled by the standard LSE algorithm, that is the distance between the current location and the selected sampling point. In this way both methods consider the knowledge about the environment with a new GP update after the same amount of traveled distance.

We can observe that the F_1 -score, (which indicates the accuracy of the classification) is higher than 97% for all the algorithms. Regarding the total path length, our SBOLSE algorithm performs very well, reducing the path required by the mobile sensor by about 70% compared to the standard LSE algorithm proposed in [10]. Also the comparison with the batch version of the LSE algorithm still show an advantage by about 32% in the total path length.

Table 1. Results of F_1 -score and total path length using the real world PH dataset extracted from waters of the Persian Gulf near Doha (Qatar), \bar{x} is the average of all experiments and $SE_{\bar{x}}$ is the standard error of the mean.

	F_1 -score		Path Length	
	\bar{x}	$SE_{\bar{x}}$	\bar{x}	$SE_{\bar{x}}$
SBOLSE	97.23	0.066	473.6	6.203
CS	98.22	0.039	1560.8	18.582
CS _{b30}	97.54	0.061	687.9	14.296

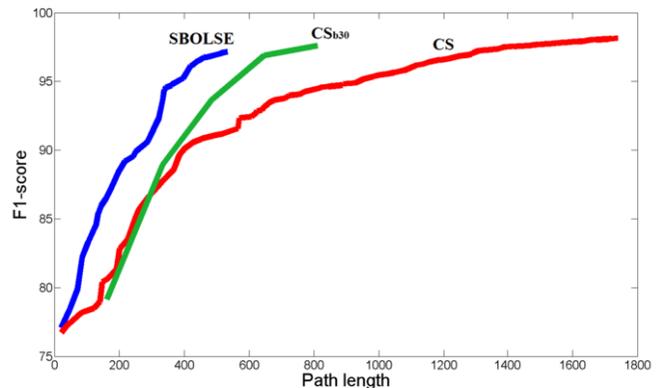


Figure 5. Runtime F_1 -score comparison on the typical example instance of the real dataset, varying the path length between SBOLSE, CS and CS_{b30} algorithms.

4.2 Synthetic CO2 dataset experiments

The second dataset consists of 10 matrices 60×179 . In this case we assume that each element represents 1 square meters of the surface. These matrices have been extracted from the color channels of portions of CO2 analysis images⁴ to obtain a dataset with a scalar field consistent with a typical environmental topology. One example of a matrix from the dataset is presented in Figure 6. The main purpose of this dataset is to test our technique on bigger matrices (more than 10,000 elements to classify) and to assess the quality of the algorithm on data different than a scalar field extracted from a body of water.

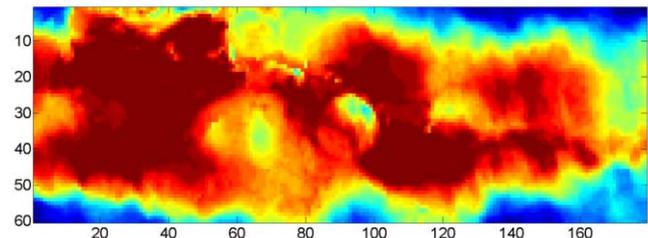


Figure 6. Example of one of the synthetic scalar fields extracted from a CO2 analysis map.

In our experiments we considered a threshold equals to 85% of the maximum value in the scalar field. Also in this case we determined

⁴ <http://oco.jpl.nasa.gov/galleries/gallerydataproducs/>

the best parameter setting and then performed tests with all the three algorithms previously described with five random initial prior composed by 10% of the elements of the matrix, for a total of 150 experiments. Again the priors were used to fit the hyperparameters of an isometric Matérn-3 [17] covariance function. The results of these experiments are shown in table 2. We can observe that these are similar to what was obtained in the real-world dataset. Specifically we obtain a reduction of about 75% of the path length compared to the standard LSE algorithm and about 25% compared to the batch version.

Table 2. Results of F₁-score and total path length using the synthetic CO2 dataset, \bar{x} is the average of all experiments and $SE_{\bar{x}}$ is the standard error of the mean.

	F ₁ -score		Path Length	
	\bar{x}	$SE_{\bar{x}}$	\bar{x}	$SE_{\bar{x}}$
SBOLSE	97.99	0.100	1355.6	26.156
CS	98.66	0.071	5588.1	136.864
CS _{b30}	98.25	0.089	1782.7	34.052

4.3 Topological skeletonization results

This test aims at computing some statistics on the unclassified points U_t during the execution of the algorithm before and after the operation of topological skeletonization. In particular, we empirically show that this heuristic significantly reduces the amount of points that need to be analyzed during the orienteering step. In Figure 7 we can observe the average reduction in the number of unclassified points after the skeletonization on a typical example instance of the real dataset. This directly translates in space reduction of the graph G used to perform the orienteering operation.

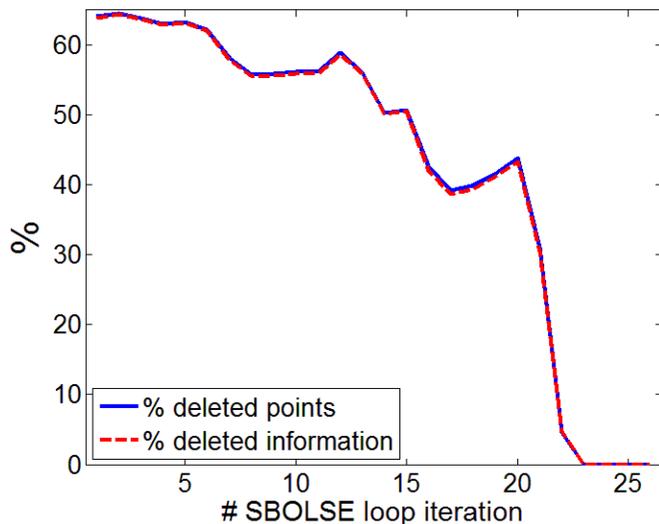


Figure 7. Reduction in the number of unclassified points and information after the topological skeletonization, during the execution of the SBOLSE algorithm

As reported in previous section, to perform the skeletonization we used a basic algorithm based on morphological operators as implemented in MATLAB function `bwmorph`. Although the operation is very simple and fast and with such a technique we do not take into account the amount of information in the deleted points, in Figure 7

we can observe that after this operation, statistically the percentage of points and information deleted are similar. This is due to the fact that, given the typical environmental phenomena, in the area representing the unclassified data, the highest amount of available information (points with the highest ambiguity value) is concentrated in the middle of the area itself. This suggests that the skeletonization is a good heuristic to apply in order to discard some of these points.

The applied algorithm based on morphological operators in some of the cases reduces the area giving a skeletonization centered where the information is concentrated (see Figure 3), whereas in other locations this is not the case. However this observation leaves open the possibility of applying different skeletonization techniques that better preserves this property, further increasing the usefulness of our technique. For example many definitions of skeleton make use of the concept of distance function [12], which is a function that for each point inside a shape gives its distance to the closest point on the boundary. Further investigations could use a similar concept in order to generate a skeletonization based on the amount of data present in the area.

We now show a comparison between two versions of our SBOLSE algorithm, with and without skeletonization of the orienteering instances, specifically we performed the experiments on the real world dataset. Results in table 3 show that the application of the skeletonization heuristic does not significantly influence the classification quality. At the same time, however, this method allows us to greatly reduce the complexity of the orienteering heuristic as previously shown in Figures 7

Table 3. Comparison between our SBOLSE algorithm using the orienteering subroutine with and without the topological skeletonization on the unclassified data U_t

	F ₁ -score		Path Length	
	\bar{x}	$SE_{\bar{x}}$	\bar{x}	$SE_{\bar{x}}$
with	97.37	0.152	449.0	7.414
without	97.70	0.075	525.0	13.891

5 CONCLUSION

In this paper we proposed a new algorithm (i.e., SBOLSE) for the level set estimation problem, considering mobile watercraft equipped with continuous-sampling sensors. Our technique formulates the level set estimation problem as an orienteering problem where the ambiguity about the classification of a location represents the score in the orienteering formulation. Our SBOLSE algorithm implements an orienteering heuristic solution as a subroutine to select an informative path that meets a given travel budget. Moreover, we present an approach based on the topological skeletonization to reduce the size of the orienteering instance we solve, allowing for on-line classification. Results show that our approach significantly outperforms the current state of the art algorithms for the level set estimation problem (i.e., LSE and LSE-batch) in terms of total travel distance, while maintaining a near-optimal classification quality.

ACKNOWLEDGEMENTS

This work was supported by the European Unions Horizon 2020 research and innovation programme under grant agreement No 689341. This work reflects only the authors' view and the EASME is not responsible for any use that may be made of the information it contains.

REFERENCES

- [1] M. A. Batalin, M. Rahimi, Y. Yu, D. Liu, A. Kansal, G. S. Sukhatme, W. J. Kaiser, M. Hansen, G. J. Pottie, M. Srivastava, and D. Estrin, 'Call and response: Experiments in sampling the environment', in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pp. 25–38, New York, NY, USA, (2004). ACM.
- [2] Harry Blum, 'A Transformation for Extracting New Descriptors of Shape', *Models for the Perception of Speech and Visual Form*, 362–380, (1967).
- [3] Chandra Chekuri and M. Pal, 'A recursive greedy algorithm for walks in directed graphs', in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pp. 245–253, (Oct 2005).
- [4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, MIT Press, third edn., 2009.
- [5] K. Dantu and G. Sukhatme, 'Detecting and tracking level sets of scalar fields using a robotic sensor network', in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 3665–3672, (April 2007).
- [6] A. Dhariwal, B. Zhang, B. Stauffer, C. Oberg, G. S. Sukhatme, D. A. Caron, and A. A. Requicha, 'Networked aquatic microbial observing system', in *International Conference on Robotics and Automation*, pp. 4285–4287, Orlando, FL, (May 2006). IEEE.
- [7] M. Dunbabin and L. Marques, 'Robots for environmental monitoring: Significant advancements and applications', *Robotics Automation Magazine, IEEE*, **19**(1), 24–39, (March 2012).
- [8] Bruce L. Golden, Larry Levy, and Rakesh Vohra, 'The orienteering problem', *Naval Research Logistics (NRL)*, **34**(3), 307–318, (1987).
- [9] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing (3rd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.
- [10] Alkis Gotovos, Nathalie Casati, Gregory Hitz, and Andreas Krause, 'Active learning for level set estimation', in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13*, pp. 1344–1350. AAAI Press, (2013).
- [11] G. Hitz, A. Gotovos, F. Pomerleau, M.-E. Garneau, C. Pradaliere, A. Krause, and R.Y. Siegwart, 'Fully autonomous focused exploration for robotic environmental monitoring', in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2658–2664, (May 2014).
- [12] Anil K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.
- [13] S. Kataoka and S. Morito, 'An algorithm for single constraint maximum collection problem', *Journal of the Operations Research Society of Japan*, **31**(4), 515–531, (1988).
- [14] Andreas Krause and Carlos Guestrin, 'Near-optimal observation selection using submodular functions', in *National Conference on Artificial Intelligence (AAAI), Nectar track*, (July 2007).
- [15] Gilbert Laporte and Silvano Martello, 'The selective travelling salesman problem', *Discrete Applied Mathematics*, **26**(2), 193–207, (1990).
- [16] M. Rahimi, R. Pon, W. J. Kaiser, G. S. Sukhatme, D. Estrin, and M. Srivastava, 'Adaptive sampling for environmental robotics', in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pp. 3537–3544 Vol.4, (April 2004).
- [17] C. E. Rasmussen and Williams C. K. I., *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, USA, 2006.
- [18] Aarti Singh, Robert Nowak, and Parmesh Ramanathan, 'Active learning for adaptive mobile sensing networks', in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks, IPSN '06*, pp. 60–68, New York, NY, USA, (2006). ACM.
- [19] Amarjeet Singh, Andreas Krause, and William J. Kaiser, 'Nonmyopic adaptive informative path planning for multiple robots', in *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI'09*, pp. 1843–1850, San Francisco, CA, USA, (2009). Morgan Kaufmann Publishers Inc.
- [20] S. Srinivasan, K. Ramamritham, and P. Kulkarni, 'Ace in the hole: Adaptive contour estimation using collaborating mobile sensors', in *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pp. 147–158, (April 2008).
- [21] T. Thomadsen and T. Stidsen, 'The quadratic selective travelling salesman problem', Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 305, DK-2800 Kgs. Lyngby, (2003).
- [22] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden, 'The orienteering problem: a survey', *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, **209**(1), 1–10, (2011).

Interruptible Task Execution with Resumption in Golog

Gesche Gierse and Tim Niemueller and Jens Claßen and Gerhard Lakemeyer¹

Abstract. Mobile robots should perform a growing number of tasks and react to time-critical events. Thus, the ability to interrupt a task and resume it later is crucial. While interleaved execution occurs often in robotics, existing approaches do not consider the fact that interrupting a task and resuming an interrupted task often requires intermediate steps. In this paper we present an approach to interruptible task execution with resumption. We propose INTRGOLOG which extends INDIGOLOG by task interruption and resumption through introducing new constructs to determine and fulfill the requirements of tasks. Our experiments on a service robot and in simulation show that the ability to switch to another task enables a robot to react in a swift and reliable fashion to new events.

1 INTRODUCTION

Mobile robots are envisioned to fulfill a growing number of tasks in the future—in domestic as well as in industrial scenarios. A required capability then is *task switching*. The problem turns out to be more involved than is obvious. For example, while a robot is cleaning up the breakfast table putting tableware into the dishwasher in the kitchen, the ringing of the doorbell adds the higher prioritized task to answer the door. Completing the initial task first would take too long (the mailman might already be gone by the time the robot answers the door). Hence the robot must *interrupt its current task* as soon as possible. But the robot may not be able to perform the new task right away, e.g., it might be carrying an object that it needs to put down first in order to have its hand free to open the door. Therefore, it needs to reason about the necessary steps to resolve conflicts between requirements when switching tasks. After completing the high priority task, the robot is expected to *resume the interrupted task*. This might involve remembering parameters of the steps performed when switching: if the robot held an object when the doorbell rang and placed it down on the quickly accessible kitchen counter, it must later retrieve the object from there to complete cleaning up the table.

In this paper, we formalize task switching as an extension of INDIGOLOG accounting for a number of problems that arise. We introduce a new kind of interruption. In contrast to CONGOLOG interrupts which may yield control to other programs at arbitrary points, we define an extended transition semantics that explicitly handles the addition, removal, and switching of tasks, which may involve the introduction of intermediate steps for resolving requirement conflicts. We also provide constructs to avoid task switching in certain parts of a program, to force the re-execution of sub-programs marked indivisible, and a well-defined interruption of durative actions.

For this purpose, we define a new transition relation $TTrans$ for task operations (adding, removing, switching) on top of



Figure 1. Evaluation scenario. The robot moves cups from the table to the dishwasher, when the doorbell rings. It should then answer the door.

INDIGOLOG's original *Trans* which defines how to execute the respective current task. To determine the intermediate steps necessary on a task switch, we introduce the new concept of *promises*. These are terms denoting asserted or required conditions (of the running or the interrupting task) used to determine conflicts during task switching. Promises are formulated with regard to actions, i.e., whether an action asserts or revokes a promise. We rely on *durative* actions, such that long-running actions can be interrupted. They are defined as a start and an (exogenous) end action. Finally, *re-execution sequences* allow to specify parts of a program that must be repeated in full on resumption, e.g., to repeat a sensing action before grasping if the latter was interrupted.

Our main contribution is the formalization of task interruption and resumption as an extension of INDIGOLOG's transition semantics and the implementation of INTRGOLOG that can deal with cases where procedures for intermediate steps can be pre-programmed.

We have evaluated the approach both in simulation and on a real robot. The results show that the proposed system enables the robot to react to new events in a swift fashion and that it indeed reduces high priority task latency.

In Section 2, we briefly introduce the Situation Calculus and GOLOG. Then we explain our approach in detail (Section 3). The evaluation is described in Section 4. Some related work is discussed in Section 5 before concluding in Section 6.

2 THE SITUATION CALCULUS AND GOLOG

In this section, we will describe the situation calculus and GOLOG and some of its variants, upon which our approach builds.

2.1 Situation Calculus

The situation calculus [16] is a dialect of first-order logic that describes a changing world. It has the sorts *situations* and *actions*. A *situation* is defined by nested terms of the form $do(a, s)$ starting from

¹ Knowledge-Based Systems Group, RWTH Aachen University, Germany, email: {gierse,niemueller,classen,lakemeyer}@kbsg.rwth-aachen.de

the initial situation S_0 , where $do(a, s)$ describes the situation that occurs after executing *action* a in situation s . *Actions* are encoded by functions and may have preconditions, e.g., the action `pick_up(x)` is only possible if the robot has an empty hand. The status of the world is described by *fluents*, e.g., `holding(x, s)` denotes that the agent is holding object x in situation s . Effects of actions are formalized by *successor state axioms*. These concepts can be formalized in a basic action theory \mathcal{D} as described in [19].

2.2 GOLOG

GOLOG [14] is a procedural programming language based on the situation calculus. The situation calculus is used to find a legal sequence of actions to satisfy a GOLOG program. The programming language features different programming constructs, such as sequence, while loops, and conditions. Additionally, it exhibits non-deterministic constructs, for example $\pi x.\delta$, where x is chosen non-deterministically such that δ can be legally executed. As an example, consider a GOLOG program to clear a table:

```

while  $\exists object.on\_table(object)$  do
   $\pi x.on\_table(x)?; pick\_up(x); put\_on\_floor(x)$ 
endWhile

```

As long as at least one object is on the table, the program non-deterministically select one, picks it up and puts it down on the floor.

In GOLOG, programs can only be executed offline. That means, given a GOLOG program, a sequence of actions is computed that fulfills the program. Then the whole sequence of actions is executed. In real-world applications this is unrealistic: a robot has to be able to sense its surroundings and make decisions based on this. Also, actions may fail and thus a pre-computed sequence cannot be executed. Therefore, extensions have been formalized to deal with these problems, two of which we are going to describe in more detail. In the following GOLOG will denote the family of programming languages.

2.2.1 CONGOLOG and INDIGOLOG

CONGOLOG [2] is an extension that adds concurrent execution (multiple programs are executed step-wise interleaved), prioritized interrupts (with a higher priority interrupt handling program urgent actions can be executed), and exogenous actions (events in the environment). However, execution still happens offline.

INDIGOLOG [3] extends CONGOLOG by adding sensing and online execution.² It interprets programs in an incremental way, such that modifications to fluents can alter the trace of situations to program completion. Such traces may then depend on data sensed during execution. These sensing results are stored in a history.

Our work is an extension of INDIGOLOG. It relies on incremental execution in order to be able to interrupt and resume tasks and to interleave the execution of multiple tasks.

2.2.2 GOLOG and INDIGOLOG Programs

A GOLOG program δ can execute an action (a) or verify a condition ($\phi?$). These basic operations can be combined with the following control structures: Sequence ($\delta_1; \delta_2$), non-deterministic

branch ($\delta_1 | \delta_2$), non-deterministic choice of argument ($\pi x.\delta$), non-deterministic iteration (δ^*), if-conditions, while-loops, and procedure calls. To search for a plan ahead of execution, the search operator $\Sigma(\delta)$ is used. Concurrently executing two programs is expressed by $\delta_1 || \delta_2$. Additionally, there are prioritized concurrency ($\delta_1 \gg \delta_2$), and concurrent iteration ($\delta^{||}$). To interrupt a program whenever condition ϕ holds, one writes $\langle \phi \rightarrow \delta \rangle$.

2.2.3 Transition Semantics

Both CONGOLOG and INDIGOLOG use a transition semantics which defines how a program δ in some situation s can evolve to a program δ' and a situation s' . As an example consider the programming construct a which executes an action:

$$Trans(a, s, \delta', s') \equiv Poss(a[s], s) \wedge \delta' = nil \wedge s' = do(a, s)$$

Here an action a transforms to the program *nil* which denotes the empty program. The transition may only be used if it is possible to execute action a in the current situation s . The resulting situation s' is the situation that occurs when action a is executed in situation s .

A program δ may terminate if it is final, i.e., $Final(\delta, s)$ holds. For example, the program *nil* is final, while the program a is not final – an action still needs to be executed. The predicate is defined recursively, e.g., a sequence of actions $\delta_1; \delta_2$ may terminate if and only if both δ_1 and δ_2 are final.

A *history* is used to represent a sequence of actions with their sensing results. A history σ is of the form $\sigma = (a_1, \mu_1) \cdot \dots \cdot (a_n, \mu_n)$ if actions a_1, \dots, a_n were executed and each action a_i returned the sensing result μ_i . It is convenient to use $end[\sigma]$ as an abbreviation for the end situation of history, defined by: $end[\epsilon] = S_0$; and inductively, $end[\sigma \cdot (a, x)] = do(a, end[\sigma])$ [3]. Additionally, $Sensed[\sigma]$ denotes the formula that contains all sensing results of history σ [3].

2.2.4 Online Execution

While GOLOG tries to find a sequence of actions beforehand and then tries to execute all actions, INDIGOLOG can execute one action and afterwards decide how to execute the remaining program. This is especially useful since we can delay decisions to the execution time and use information gained by sensing actions. The transition semantics allow to search for the next step of the program. Assume we started a program and at some later point the program δ is remaining to be executed. Additionally, we have executed some actions and obtained corresponding sensing results. In the online execution the interpreter will now either stop, if the remaining program δ is final, or find one transition from δ and situation s to δ' and some situation s' . If s and s' are not equal, s' will be of the form $do(a, s)$. In this case the action a is executed in the real world. The next step is determined by the basic action theory, the transition and final rules, and the sensing results. We also monitor exogenous actions, i.e., actions the agent observes but does not execute itself. These are added to the history of executed actions.

2.2.5 Concurrency and Interrupts

In INDIGOLOG, programs which are run concurrently are executed in an interleaved fashion. An interrupt $\langle \phi \rightarrow \delta \rangle$ executes its program δ if the condition ϕ holds and no process with higher priority is executed. It will suspend processes of lower priority. After execution of the interrupt, suspended processes are continued. A major

² Since INDIGOLOG includes CONGOLOG we focus on the former.

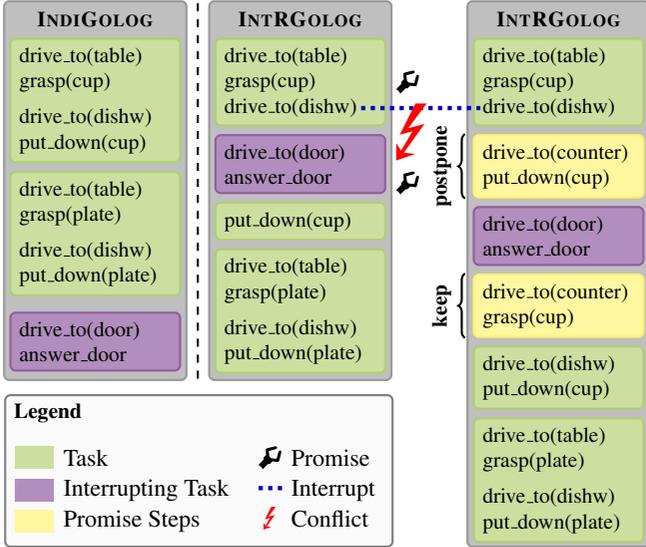


Figure 2. *Task interruption.* The left shows how the interrupting task is simply appended in INDIGOLOG, while INTRGOLOG inserts it as soon as the interrupt event occurs. The conflict caused by the `hand_used` promise is resolved by introducing suitable steps to postpone and keep the promise.

problem with interrupts is that they yield control to interrupts at arbitrary points. This is a problem for interruptible task execution with resumption, since additional actions might be necessary before the switch can be performed, and again when resuming the original task.

As an example why GOLOG interrupts do not suffice for task interruption consider the following program:

```
{doorbell_rings → answer_door} } } clean-up
```

If the doorbell rings, the robot will interrupt and answer the door, otherwise it will clean up. This is a simple approach to interleaved task execution and it does not work as expected in the envisioned situations: Consider that the doorbell rings while the robot is about to put an object in the dishwasher. The switch will be immediate, thus the robot drives to the door, but it cannot open it, because its hand is still full. Therefore, the higher prioritized program will yield control and the clean up may continue. However, the robot is not in front of the dishwasher anymore, thus the precondition of the put-in-dishwasher action does not hold anymore. So, without performing intermediate steps, the combined interruptible program may no longer be executable. To avoid this, one would have to anticipate all problems that may potentially arise, along with possible solutions, and include those in the interrupt. This is not only tedious, but also error-prone, particularly when the interplay between multiple interleaved tasks has to be considered. Instead, we introduce a new notion of interrupts where conflicts are avoided by executing intermediate steps on interruption.

2.2.6 Durative actions

To represent the concurrency of real-world actions *durative actions* [19] can be used. It essentially means splitting up an action in an instantaneous start and an (exogenous) instantaneous end action that occurs when the action is finished. As long as the action is being executed, a fluent for this action is true. In between, a program may continue or interrupts may be handled. Another possible representation has been used in CC-GOLOG [7], where an action

`waitFor(condition)` is used to have the agent wait until some formula *condition* (which depends on continuously changing fluents such as the robot position) becomes true, and then continue executing further actions (e.g., stopping the movement). Here however, we want to allow the interruption of durative actions through exogenous events. Therefore, we have:

$$\text{exec_dur}(d) \stackrel{\text{def}}{=} \text{dur_act_start}(d); \text{wait_for}(d)$$

$$\text{wait_for}(d) \stackrel{\text{def}}{=} \mathbf{while} \text{ dur_flu}(d) \mathbf{do} \text{ wait} \mathbf{endWhile}$$

This is the typical case where nothing except waiting (and thereby checking for interrupts) happens while one durative action is running. Our formalization also allows to state explicitly in which order to start durative actions and wait for their termination, and thus interleave such processes.

3 EXTENDED TRANSITION SEMANTICS

The robot should be able to execute a set of tasks instead of just one program. Therefore, we define tasks which consist of a program, a priority, and a unique ID. We extend the transition semantics of INDIGOLOG to work on a set of tasks. Instead of directly using the transition predicate *Trans*, we introduce *TTrans* whose semantics is added on top of the original *Trans* predicate. With it a task can be added or removed. It facilitates task switching and uses the original *Trans* to execute the current task.

As described in Section 2.2.5 CONGOLOG interrupts are not sufficient to handle task switches, since intermediate steps may be necessary to interrupt and later resume a task. To ensure that these necessary intermediate steps are performed when switching tasks we introduce two additional concepts.

Re-execution sequences ensure that actions that need to be executed together are fully repeated on resumption. For example, the robot uses a sensing action to detect an object on a table and is about to grasp it. When interrupted in this situation it needs to sense if the object is still there on resumption. Essentially, an interrupted task is modified by restoring an already executed part of the program.

Promises are used to specify necessary conditions for (parts of) programs and to determine conflicts during task switches. The basic action theory defines which action asserts or retracts a promise during execution. When switching to a new task, the system determines if the promises are *conflicting*, that is whether any currently asserted promise is required by the new task. In this case, domain-specific procedures are used to *postpone* and later *keep* the promise. The postpone program is executed before starting a different task and the keep procedure is executed before resuming the interrupted task. Task-specific information may be stored during postponing to be used when a promise is to be kept again.

As an example, consider Figure 2. The left column shows a classic INDIGOLOG implementation that requires the two objects to be cleaned up before answering the door, as the clean-up program was already running. The two right columns show INTRGOLOG's behavior. Once the interrupt event occurs (blue dotted line), the respective action is interrupted and the new higher priority task inserted. But the robot has grasped an object with its only hand and needs this very hand free in order to successfully execute the new, higher prioritized task of answering the door. A promise `hand_used` is defined by the user. Finishing grasping an object or starting to open a door would assert the promise, while finishing putting down an object or finishing opening the door retracts it. When switching after grasping

to opening the door, the robot would postpone the promise *hand_used* by driving to a nearby counter and putting down what it is carrying. However, to later resume the task, it then needs to have the object in its hand again. The *keep* procedure will handle this by remembering where it put the object and then retrieve it (yellow boxes). Afterwards the previous task can be resumed.

3.1 Task Definition

A task is defined as a tuple $\tau = \langle \delta, \text{prio}, \text{id} \rangle$, where δ is a program as before (that will be modified by *Trans*), *prio* is the task's priority and *id* is a unique ID. As a shorthand notation we will use $\text{prog}(\tau)$, $\text{prio}(\tau)$ and $\text{id}(\tau)$ to access the program, the priority and the ID of the task τ , respectively.

We write finite sets τ_1, \dots, τ_n of tasks as $\Omega = \{\tau_1, \dots, \tau_n\}$. We will write $\Omega[\tau_i = \tau'_i]$ to denote the task set that is like Ω except with τ_i replaced by τ'_i . Furthermore, $\Omega[\text{prog}(\tau_i) = \delta']$ will denote Ω where the program of the task τ_i is changed to δ' . Also, $\Omega[\{\tau_1 = \tau'_1, \dots, \tau_k = \tau'_k\}]$ will denote that all tasks τ_i are changed to τ'_i for $1 \leq i \leq k$ and the other tasks remain identical to Ω .

3.2 Transition Semantics Extension Idea

We define an extension of *Trans* called *TTrans*. It will handle tasks, while referring to the standard INDIGOLOG transition semantics to execute a single task. Intuitively, *Trans* defines transitions between configurations consisting of a program δ and a situation s : $(\delta, s) \vdash (\delta', s')$. In contrast *TTrans* yields transitions of the form $(i, \Omega, s) \vdash (i', \Omega', s')$, where i, i' are the indexes denoting the current task, Ω, Ω' are task sets and s, s' are situations. A modification of Ω to Ω' can be either *adding* a new task, *removing* a task, *executing* the current task or *switching* to the task with highest priority.

Generally, our new top-level transition predicate *TTrans* uses fluents that are set by exogenous actions to decide which of the above possibilities are used in each step of an online execution.

In the following, we will present a number of sufficient conditions for doing a transition step $TTrans(i, \Omega, s, i', \Omega', s')$. The actual axiom defining *TTrans* is then understood as the completion of those conditions, i.e., $TTrans(i, \Omega, s, i', \Omega', s')$ is equivalent to the disjunction over all right-hand sides of the rules given below. All situation calculus terms (e.g., fluents, actions, procedures) are set in monospace, while internal formulas are set in *italics*.

3.3 Adding a Task

A task is added by an exogenous action $\text{add_task}(\tau)$. We use the fluent taskupdate which will denote whether there is a new task. Additionally, if there is a new task, the fluent new_task will contain the new task. If this fluent is set, we will update the set of tasks Ω .

$$\begin{aligned} TTrans(i, \Omega, s, i', \Omega', s') \subset & \\ \text{taskupdate}[s] \wedge \exists \tau_{\text{new}}. \text{new_task}[s] = \tau_{\text{new}} & \\ \wedge \Omega' = \Omega \cup \{\tau_{\text{new}}\} \wedge s = s' \wedge i = i' & \end{aligned} \quad (1)$$

3.4 Removing a Task

A task can also be removed from Ω . If a task is final, it can be removed. Similar to *Trans* we also add an extended version of *Final* called *TFinal* to define if tasks are final. A set of tasks is final if it

is empty and a single task is final if its program is final regarding the original CONGOLOG's *Final*-rules:

$$\begin{aligned} TFinal(\Omega, s) &\equiv \Omega = \emptyset \\ TFinal(\tau, s) &\equiv Final(\text{prog}(\tau)) \end{aligned}$$

Besides removing final tasks, a task can also be deleted using an exogenous action $\text{remove_task}(\text{id})$. We use the fluent taskremove to denote that a task deletion is requested. The fluent delete_task will contain the ID of the task that should be removed. The index i denoting the current task is set to zero to express that currently no task is selected to execute (the selection of the new task will be handled by the task switch case described below).

$$\begin{aligned} TTrans(i, \Omega, s, i', \Omega', s') \subset & \\ s = s' \wedge \exists \tau \in \Omega \exists \text{id}. (Final(\tau, s) \vee \text{taskremove}[s] & \\ \wedge \text{delete_task}[s] = \text{id}) \wedge \text{id}(\tau) = \text{id} & \\ \wedge (\text{id} = i \supset i' = 0) \wedge (\text{id} \neq i \supset i' = i) & \\ \wedge \Omega' = \Omega \setminus \{\tau\} & \end{aligned} \quad (2)$$

3.5 Executing a Task

We can execute the current task if the following conditions hold: A proper current task is defined by the *id* i (zero indicates that this is not the case, either because the current task was finished or removed by the user). There is neither a new task to be added nor a task to be removed, and if task switching is allowed, the current task has highest priority. A step of the program of the current task is the executed according to the original *Trans*:

$$\begin{aligned} TTrans(i, \Omega, s, i', \Omega', s') \subset & \\ \exists \tau_i \in \Omega. \text{id}(\tau_i) = i \wedge i \neq 0 \wedge i' = i \wedge \neg \text{taskupdate}[s] & \\ \wedge \neg \text{taskremove}[s] \wedge (\neg \text{switching_allowed}[s] & \\ \vee \forall \tau_k \in \Omega. \text{prio}(\tau_k) \leq \text{prio}(\tau_i)) & \\ \wedge \exists \gamma. Trans(\text{prog}(\tau_i), s, \gamma, s') \wedge \Omega' = \Omega[\text{prog}(\tau_i) = \gamma] & \end{aligned} \quad (3)$$

3.6 Switching Tasks

Switching a task contains some technicalities, so we will discuss the corresponding rule in an incremental manner. We will first present a simple version that does not use any intermediate steps and then describe the formalities needed for re-execution sequences and promises and how they are integrated in *TTrans*. For now consider switching to the task with highest priority:

$$\begin{aligned} TTrans(i, \Omega, s, i', \Omega', s) \subset & \\ \exists \tau_j \in \Omega. \forall \tau_k \in \Omega. \text{prio}(\tau_k) \leq \text{prio}(\tau_j) & \\ \wedge i' = \text{id}(\tau_j) \wedge \exists \tau_i \in \Omega. \text{id}(\tau_i) = i & \\ \wedge \text{prio}(\tau_j) > \text{prio}(\tau_i) \wedge \text{id}(\tau_i) \neq \text{id}(\tau_j) & \\ \wedge \Omega' = \Omega & \end{aligned} \quad (4)$$

Next, we include the fluent switching_allowed that enables and disables task switching. The fluent is toggled by the actions enable_switching and disable_switching . In the definition of removing tasks we introduced the case where currently no task is selected, indicated by $i = 0$. This is a special case, because no task can be executed by the *TTrans*-rule for execution. So even

if switching is disabled, we need to select a new current task if $i = 0$. The resulting rule looks as follows, where the new part is underlined:

$$\begin{aligned}
TTrans(i, \Omega, s, i', \Omega', s) \subset & \quad (5) \\
\underline{(i = 0 \vee \text{switching_allowed}[s])} & \\
\wedge \exists \tau_j \in \Omega. \forall \tau_k \in \Omega. \text{prio}(\tau_k) \leq \text{prio}(\tau_j) & \\
\wedge i' = id(\tau_j) \wedge (\exists \tau_i \in \Omega. id(\tau_i) = i \wedge \text{prio}(\tau_j) > \text{prio}(\tau_i)) & \\
\wedge id(\tau_i) \neq id(\tau_j) \vee \underline{i = 0} \wedge \Omega' = \Omega &
\end{aligned}$$

3.6.1 Re-execution Sequences

A sequence of actions can be marked as a re-execution sequence. It is then executed as usual, however, in the case of a task switch within that sequence, the whole re-execution sequence is executed from scratch at the resumption of the task.

The programmer uses the construct **reexec**(δ) in their procedures. *Trans* then transforms this to **re**(δ, δ). Transitions from **re**(δ', δ) will only modify δ' , while δ remains the original δ . We extend (CONGOLOG's original) *Trans* and *Final* by the following:

$$\begin{aligned}
Trans(\mathbf{reexec}(\delta), s, \mathbf{re}(\delta, \delta), s) &\equiv True \\
Trans(\mathbf{re}(\delta, \delta_{re}), s, \delta', s') &\equiv \\
\exists \gamma. Trans(\delta, s, \gamma, s') \wedge \delta' = \mathbf{re}(\gamma, \delta_{re}) & \\
\vee Final(\mathbf{re}(\delta, \delta_{re})) \wedge \delta' = nil & \\
Final(\mathbf{reexec}(\delta), s) &\equiv False \\
Final(\mathbf{re}(\delta, \delta_{re}), s) &\equiv Final(\delta, s)
\end{aligned}$$

As an example consider the following program:

```

reexec(exec_dur(detect_cup);
  if cups_on_table then
    exec_dur(grasp) else noop endIf)

```

where the robot grasps a cup, if it detected one. If the robot is interrupted when starting to grasp the object, the robot would continue with the whole program above on resumption. Let the above program be denoted by **reexec**(δ_{SG}). Internally, this is transformed to **re**(δ_{SG}, δ_{SG}). When executing it in a situation where cups are sensed, we would eventually reach **re**(exec_dur(grasp), δ_{SG}). If interrupted at that time, we want to execute **re**(δ_{SG}, δ_{SG}) again on resumption. Thus, we now extend Formula 5 to replace all occurrences of **re**(δ, δ_{re}) in the current task by **re**(δ_{re}, δ_{re}) before switching. This is done by the formula *reset_re*(τ_i, τ'_i).

$$\begin{aligned}
TTrans(i, \Omega, s, i', \Omega', s) \subset & \quad (6) \\
\underline{(i = 0 \vee \text{switching_allowed}[s])} & \\
\wedge \exists \tau_j \in \Omega. \forall \tau_k \in \Omega. \text{prio}(\tau_k) \leq \text{prio}(\tau_j) & \\
\wedge i' = id(\tau_j) \wedge (\exists \tau_i \in \Omega. id(\tau_i) = i \wedge \text{prio}(\tau_j) > \text{prio}(\tau_i)) & \\
\wedge id(\tau_i) \neq id(\tau_j) \vee i = 0 & \\
\wedge \underline{\exists \tau'_i. \text{reset_re}(\tau_i, \tau'_i)} & \\
\wedge \underline{\Omega' = \Omega[\text{prog}(\tau_i) = \tau'_i]} &
\end{aligned}$$

To handle cases where intermediate steps are also necessary before starting the new task we introduce the concept of *promises*.

3.6.2 Promises

Promises are terms that denote asserted or required conditions of programs. They are defined as part of the basic action theory. We introduce new situation independent predicates to describe promises and which actions assert and retract a promise.

Procedures to *postpone* and *keep* a promise need to be defined. These are the intermediate steps executed before starting the interrupting task and before resuming the interrupted task. We use **promise**(*prom*, *order*) to denote that the term *prom* is a promise and is associated with a numeric value (natural number) *order* to indicate in which order to postpone and keep promises in case of an interruption. To denote that an action *a* starts a promise *prom*, we write **asserts_promise**(*a*, *prom*). For example, a successful exogenous *end_grasp* starts the promise *has_obj*. A promise stops being asserted when an action retracts it, written **retracts_promise**(*a*, *prom*). For example, a successful exogenous *end_grasp* starts the promise *has_obj* and the action *end_put_down* ends the promise *has_obj*.

Internally, the fluent **asserted_promise**(*prom*, *s*) indicates that the promise *prom* is asserted in situation *s*. The successor state axiom of this fluent is described using the predicates from above:

$$\begin{aligned}
\mathbf{asserted_promise}(p, do(a, s)) &\equiv \\
\mathbf{asserts_promise}(a, p) \vee & \\
\mathbf{asserted_promise}(p, s) \wedge \neg \mathbf{retracts_promise}(a, p) &
\end{aligned}$$

Now we need to define procedures to perform intermediate steps when switching a task. They are called *postpone* and *keep* procedures. In the case of the promise *hand_used* the robot would need to put away what it is holding with the *postpone* procedure. To later resume the interrupted task, the robot needs to hold the object again before continuing the task. Thus, in the *keep* procedure it has to find the object again. Our extension provides two actions and a fluent to help with remembering necessary parameters for keeping a promise. We introduce a fluent **has_param**(*i*, *prom*, *key*, *s*)=*value* that indicates that the task with ID *i* has the parameter *value* for the postponed promise *prom* and the key *key* in situation *s*. When postponing a promise, we can use the action **set_param**(*id*, *prom*, *key*, *value*) to set this fluent. In the case of *hand_used* we would for example save the table where we put the object. In the *keep* procedure we can then read this value from the fluent. After successfully using the information, the action **used_param**(*id*, *prom*, *key*) resets the fluent **has_param**(*id*, *prom*, *key*) to *false*.

As an example consider the procedures to keep and postpone the promise *hand_used*. As domain independent fluents, we first define which procedure keeps and postpones the promise:

$$\begin{aligned}
\mathbf{postpone_promise}(\mathbf{hand_used}, \mathbf{put_somewhere}(Id)). & \\
\mathbf{keep_promise}(\mathbf{hand_used}, \mathbf{get_back_obj}(Id)). &
\end{aligned}$$

To *postpone* we define a procedure which will ensure that it is actually holding something, in that case it will sense the nearest table, go there, put the object down and remember the place.

```

proc put_somewhere(Id)
  sense_holding_obj;
if holding_obj then
  sense_nearest_table;
  ?(nearest_table = Table);
  exec_dur(drive_to(Table));
  exec_dur(put_down);
  set_param(hand_used, Id, table, Table)
endIf endProc

```

Before resuming the task, we will check if a location for an object to grasp was saved (the promise `hand_used` could have been in effect because the robot was opening a door when interrupted, in that case we do not want to pick up any object. Re-execution sequences will take care of restarting durative actions that were interrupted during execution). In that case the robot will go to the table, detect the object and pick it up. Afterwards it will indicate that it used the information stored by the postpone procedure, which will reset the fluent for that promise, ID and key to *false*.

```

proc get_back_obj(Id)
  if ¬has_param(hand_used, Id, table) = false then
    ?(has_param(hand_used, Id, table) = Table);
    exec_dur(drive_to(Table));
    exec_dur(detect_obj);
    if obj_on_table then exec_dur(grasp) endIf;
    used_param(hand_used, Id, table)
  endIf endProc

```

So, in case of a task interruption, we need to check if there is a promise that is currently asserted but may be needed in the interrupting task (like a hand holding an object and the interrupting task needs the hand to open the door). To define this, we will look ahead what promises could be asserted by the interrupting task by extending the definition of `asserts_promise` to programs. For example:

```

asserts_promise( $\delta_1; \delta_2, p$ )  $\stackrel{\text{def}}{=}$ 
  asserts_promise( $\delta_1, p$ )  $\vee$  asserts_promise( $\delta_2, p$ )
asserts_promise( $\pi v. \delta, p$ )  $\stackrel{\text{def}}{=}$   $\exists v. \text{asserts\_promise}(\delta, p)$ 

```

The rules for the remaining constructs are defined in a similar manner. With this lookahead we can describe if a promise is *contradicting*, i.e., it is asserted and the interrupting task τ could assert it, too.

```

is_contradicting_promise(prom, order,  $\tau, s$ )  $\stackrel{\text{def}}{=}$ 
  promise(prom, order)  $\wedge$  asserted_promise(prom, s)
   $\wedge$  asserts_promise( $\tau, \text{prom}$ )

```

For all contradicting promises we want to prepend the program of the interrupting task with the corresponding *postpone* procedure and the interrupted task with the respective *keep* procedure. We define two helper functions to prepend the current task with these procedures given the current situation s , the set of tasks Ω and the ID j of the interrupting task:

```

postpone_contradicting_promises( $s, \Omega, j$ ) =  $\Omega'$   $\stackrel{\text{def}}{=}$ 
   $\exists \tau \in \Omega. id(\tau) = j$ 
   $\wedge \Omega' = \Omega[\text{prog}(\tau) = \text{disallow\_switching}; \delta_p;$ 
    allow_switching;  $\text{prog}(\tau)]$ 

```

where $\delta_p = \delta_1; \dots; \delta_n$ and $\delta_1, \dots, \delta_n$ are exactly those δ_i such that

```

 $\exists \text{prom}_i, \text{order}_i$ 
  is_contradicting_promise( $\text{prom}_i, \text{order}_i, \tau, s$ )  $\wedge$ 
  postpone_promise( $\text{prom}_i, \delta_i$ )

```

and $\text{order}_i > \text{order}_j$ for all $i > j$.

In a similar manner, we define a function to prepend each task that asserted a conflicting promise with the associated *keep* procedure:

```

keep_postponed_promises( $\Omega, s$ ) =  $\Omega'$   $\stackrel{\text{def}}{=}$ 
   $\Omega' = \Omega[\{\text{prog}(\tau) = \delta_k^T; \text{prog}(\tau) \mid \tau \in \Omega\}]$ 

```

where $\delta_k^T = \delta_n; \dots; \delta_1$ and $\delta_1, \dots, \delta_n$ are exactly those δ_i such that

```

 $\exists \text{prom}_i, \text{order}_i$ 
  is_contradicting_promise( $\text{prom}_i, \text{order}_i, \tau, s$ )
   $\wedge$  keep_promise( $\text{prom}_i, \delta_i$ )
   $\wedge$  promise_made_to( $\text{prom}_i, id(\tau), s$ )

```

and $\text{order}_i > \text{order}_j$ for all $i > j$. Notice that the order of the keep and postpone procedures is reversed to allow dealing with promises that depend upon other promises.

We can now put both helper functions together to modify the set of tasks with all *postpone* and *keep* procedures:

```

handle_proms( $j, \Omega, s$ ) =  $\Omega'$   $\stackrel{\text{def}}{=}$ 
   $\wedge \Omega'' = \text{keep\_postponed\_promises}(\Omega, s)$ 
   $\wedge \Omega' = \text{postpone\_contradicting\_promises}(s, \Omega'', j)$ 

```

To handle promises when switching a task, we then extend Formula 6 to use this function after handling re-execution sequences. This leads us to the final version of the *TTrans*-rule for switching a task:

$$\begin{aligned}
 TTrans(i, \Omega, s, i', \Omega', s) \subset & \quad (7) \\
 (i = 0 \vee \text{switching_allowed}[s]) & \\
 \wedge \exists \tau_j \in \Omega. \forall \tau_k \in \Omega. \text{prio}(\tau_k) \leq \text{prio}(\tau_j) & \\
 \wedge i' = id(\tau_j) \wedge (\exists \tau_i \in \Omega. id(\tau_i) = i \wedge \text{prio}(\tau_j) > \text{prio}(\tau_i)) & \\
 \wedge id(\tau_i) \neq id(\tau_j) \vee i = 0 & \\
 \wedge \exists \tau'_i. \text{reset_re}(\tau_i, \tau'_i) & \\
 \wedge \Omega' = \text{handle_proms}(id(\tau_j), \Omega[\tau_i = \tau'_i], s) &
 \end{aligned}$$

We then take the disjunction over all right hand sides of the formulas for adding, removing, executing and switching tasks to gain a formula for *TTrans*.

3.7 Online Execution

We adapt the definition of online execution presented in [3] to use *TTrans* instead of *Trans* and *TFinal* instead of *Final*: Assume we started with a set of tasks Ω_0 and a current task i_0 (with $\tau_{i_0} \in \Omega_0$) in situation S_0 . Then at some later point, we have executed actions a_1, \dots, a_k and have obtained sensing results μ_1, \dots, μ_k and therefore are now in history $\sigma = (a_1, \mu_1) \cdot \dots \cdot (a_k, \mu_k)$ with the set of tasks Ω left to be executed, where one task $\tau_i \in \Omega$ is the current task. At each execution step, we need to decide what to do next. This can be either stopping execution if all tasks are finished, executing a step of the current task, or changing the set of tasks.

- stop, if $D \cup C \cup \mathcal{D}_{\mathcal{I}\mathcal{L}} \cup \mathcal{C}_{\mathcal{I}\mathcal{L}} \cup \text{Sensed}[\sigma] \models TFinal(\Omega, \text{end}[\sigma]);$
- return the set of remaining tasks Ω' and id i' of the current task, if $D \cup C \cup \mathcal{D}_{\mathcal{I}\mathcal{L}} \cup \mathcal{C}_{\mathcal{I}\mathcal{L}} \cup \text{Sensed}[\sigma] \models TTrans(i, \Omega, \text{end}[\sigma], i', \Omega', \text{end}[\sigma]),$ and no action is required in this step;

- return action a , i and Ω' , if
 $\mathcal{D} \cup \mathcal{C} \cup \mathcal{D}_{\mathcal{I}\mathcal{L}} \cup \mathcal{C}_{\mathcal{I}\mathcal{L}} \cup \text{Sensed}[\sigma] \models$
 $TTrans(i, \Omega, \text{end}[\sigma], i, \Omega', \text{do}(a, \text{end}[\sigma])).$

Here \mathcal{D} is an action theory for INDIGOLOG and \mathcal{C} is a set of axioms defining the predicates $Trans$ and $Final$ as in INDIGOLOG. $\mathcal{D}_{\mathcal{I}\mathcal{L}}$ contains the additional basic action theory axioms needed for the new built-in actions, fluents and procedures. $\mathcal{C}_{\mathcal{I}\mathcal{L}}$ contains the axioms for $TTrans$ as described above and the new $TFinal$ -rules described in Section 3.4. It also contains the extension of $Trans$ and $Final$ for re-execution sequences from Section 3.6.1. If the online execution returned an action, that action will be executed on the robot and sensing results are added to the history.

So analogously to [3] an online execution of a set of programs Ω and a current task number i starting from a history σ is defined as a sequence of online configurations $(i_0 = i, \Omega_0 = \Omega, \sigma_0 = \sigma), \dots, (i_n, \Omega_n, \sigma_n)$ such that for $j = 0, \dots, n-1$:

$$\mathcal{D} \cup \mathcal{C} \cup \mathcal{D}_{\mathcal{I}\mathcal{L}} \cup \mathcal{C}_{\mathcal{I}\mathcal{L}} \cup \text{Sensed}[\sigma_j] \models$$

$$TTrans(i_j, \Omega_j, \text{end}[\sigma_j], i_{j+1}, \Omega_{j+1}, \text{end}[\sigma_{j+1}])$$

$$\sigma_{j+1} = \begin{cases} \sigma_j & \text{if } \text{end}[\sigma_{j+1}] = \text{end}[\sigma_j] \\ \sigma_j \cdot (a, \mu) & \text{if } \text{end}[\sigma_{j+1}] = \text{do}(a, \text{end}[\sigma_j]) \\ & \text{and } a \text{ returns } \mu. \end{cases}$$

With this formalization we can now prove that our approach is able to simulate INDIGOLOG.

Theorem 1. *Online execution of the program δ in INDIGOLOG is equivalent to executing the single task $\langle \delta, \text{prio}, 1 \rangle$ in INTRGOLOG, given the same sensing results and that no tasks are added or removed.*

Proof. We show, given the online execution of a program δ in INDIGOLOG starting from a history σ as a sequence of configurations $(\delta = \delta_0, \sigma = \sigma_0), \dots, (\delta_n, \sigma_n)$, it holds for all $0 \leq i < n$ that

$$\mathcal{D} \cup \mathcal{C} \cup \text{Sensed}[\sigma_i] \models Trans(\delta_i, \text{end}[\sigma_i], \delta_{i+1}, \text{end}[\sigma_{i+1}])$$

if and only if

$$\mathcal{D} \cup \mathcal{C} \cup \mathcal{D}_{\mathcal{I}\mathcal{L}} \cup \mathcal{C}_{\mathcal{I}\mathcal{L}} \cup \text{Sensed}[\sigma_i] \models$$

$$TTrans(1, \{\langle \delta_i, \text{prio}, 1 \rangle\}, \text{end}[\sigma_i],$$

$$1, \{\langle \delta_{i+1}, \text{prio}, 1 \rangle\}, \text{end}[\sigma_{i+1}]),$$

where \mathcal{D} , \mathcal{C} , $\mathcal{D}_{\mathcal{I}\mathcal{L}}$ and $\mathcal{C}_{\mathcal{I}\mathcal{L}}$ are defined as above.

We can show this by induction over the configuration steps $1 \dots n$. The idea here is that because no new tasks are added, only the $TTrans$ -rule for execution can be used: Since by prerequisite of the theorem no tasks are added or removed, the conditions on the right hand side of the Formulas 1 and 2 do not hold. Also, Formula 7 cannot be used since there is only one task that may not be removed and thus no task τ_j exists that fulfills this formula (e.g., $id(\tau_j) \neq id(\tau_i)$).

Thus, only the rule to execute the current task can be used. Since it uses the original $Trans$ -rules this will yield the same results as applying the semantics of INDIGOLOG. \square

4 EVALUATION

As a basis for our implementation we used the standard Prolog implementation of INDIGOLOG.³ To show the possibilities offered by

³ Project page at <https://bitbucket.org/ssardina/indigolog>

our approach we choose a domestic service robot scenario shown in Figure 1. The setting consists of two tasks:

Clean up The robot has to clean up a table in the living room by moving all cups on it to the dishwasher in the kitchen. Since the robot has only one arm it can only transport one cup at a time.

Answer Door The robot drives to the blue door and opens it.

We focus on the situation, where the robot is currently on the way from the table to the dishwasher with a cup in its hand when the doorbell rings. With INTRGOLOG the robot can drive to a nearby counter, put down the cup and then answer the door. As discussed in Section 2.2.5 it would be tedious, error prone and not scalable to use INDIGOLOG to successfully interrupt the clean-up task. Thus, the robot will drive to the dishwasher and finish loading the cup inside before answering the door. We compare the times needed to reach the door after the doorbell rang.

We tested the scenario on a real robot [4] as well as in simulation. The simulation is based on Gazebo [13] using a Robotino 3 robot. Benchmarks were made with the simulation on an Intel Core i7-3770 at 3.40 GHz with 4 cores and hyper-threading enabled.

To determine intermediate steps we use a promise `handused` and procedures to `postpone` and `keep` that promise as described in Section 3.6.2. For the clean up task the robot drives to a specified table. As long as there are cups on the table it grasps one and puts it in the dishwasher. Then it returns to the table and checks if there are any cups left (or new ones were put there):

```
proc(clean_up(Table), [
  exec_dur(drive_to(Table)),
  exec_dur(detect_cup),
  while(cups_on_table, [
    reexec([exec_dur(detect_cup),
            if(cups_on_table, [grasp], [])]),
    if(holding_cup, [
      reexec([exec_dur(drive_to(kitchen)),
              exec_dur(put_in_dishwasher)]),
      reexec([exec_dur(drive_to(Table)),
              exec_dur(detect_cup)]),
      [exec_dur(detect_cup)]])]).
```

When answering the door, the robot drives to the door and opens it.

```
proc(answer_door, [ exec_dur(drive_to(door)),
  exec_dur(open_door)]).
```

The doorbell is modeled as an exogenous action, that adds a task with the program `answer_door` and high priority.

To measure the improvement in reaction time, we compare to a simplified version of the above clean up program in INDIGOLOG. There, the robot only drives to the table once, picks up an object and puts it in the dishwasher. Afterwards, it answers the door:

```
proc(clean_up_once_indigolog(Table), [
  exec_dur(drive_to(Table)),
  exec_dur(detect_cup),
  if(cups_on_table, [grasp], []),
  if(holding_cup, [
    exec_dur(drive_to(kitchen)),
    exec_dur(put_in_dishwasher),
    answer_door], [])]).
```

The signal of the doorbell was given to both INDIGOLOG and INTRGOLOG randomly (but with a similar distribution) after the robot picked up the cup but had not yet placed it in the dishwasher.

While INTRGOLOG is able to switch to the new task inserted by the exogenous doorbell action, INDIGOLOG performs its simplified procedure. The results are shown in Figure 3. Blue dots indicate the time INDIGOLOG needed from receiving the doorbell until arriving at the door, red squares the times for INTRGOLOG. The X-axis denotes time since the program start. When the doorbell rings early, the INTRGOLOG time is significantly shorter. In contrast, when the doorbell sounds close to arriving at the dishwasher, no significant time advantage can be achieved by putting the cup somewhere else than the dishwasher. On average, reaction times were about 15 % shorter. The experiment was run 700 times for both systems.

To make the comparison challenging we simplified the task significantly for the INDIGOLOG agent. Otherwise, it would have needed to drive back to the table and put away all cups before answering the door. This simple scenario already shows the potential of our approach. With INTRGOLOG the robot is able to react to new events promptly. In addition, it enables the robot to resume tasks after interruption. This makes the robot more reliable since given instructions are finished while unexpected events are handled appropriately.

5 RELATED WORK

Interleaved execution of processes has been used in operating systems to simulate parallel execution on a single CPU [21]. In contrast to operating systems, we do not want to simulate parallelism, because context switches in the real world are very time consuming. Our approach interleaves a task with a more important one.

Scheduling algorithms can be formalized in GOLOG [18]. However, the approach uses offline computation and cannot take into account new tasks or unexpected duration of tasks. Another scheduling-based approach is Temporal Flexible Golog [5], where low-level components of a robot are scheduled such that the execution fulfills time constraints between the components. However, INTRGOLOG focuses on tasks with intermediate steps between switching tasks. In our work the term *promise* denotes that we promise the programmer that a specified condition will hold when the program is executed, even if the task was interrupted in between. The term has already been used in a GOLOG context [15]. There the word is used in a multi-agent setting to describe that one agent promises another one that it will perform a request. Schiffer et al. [20] propose a self-maintenance system in READYLOG to ensure that during plan execution some predefined constraints are satisfied. They monitor the execution, detect events and try to fix occurring problems. In contrast, we focus on interleaving tasks such that resumption is possible.

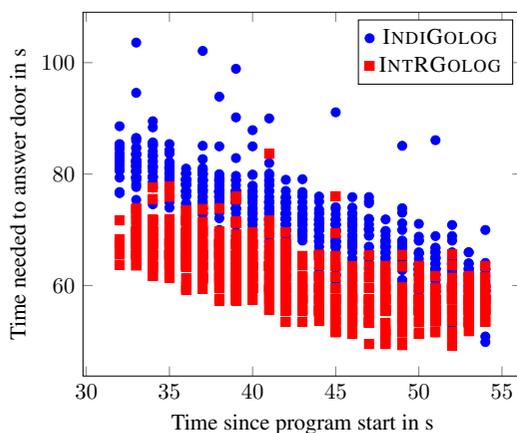


Figure 3. Time needed to arrive at the door since the doorbell rang.

A GOLOG-based approach is presented by Kelly and Pearce in [12], where the focus is on extending GOLOG with true concurrency in order to allow to coordinate multiple agents acting in parallel. However, they do not present a mechanism to deal with conflicts arising during the execution of concurrent programs akin to our promises.

Belief-Desire-Intention (BDI) systems like PRS [10] or AgentSpeak(L) [17] allow to execute multiple plans in an interleaved or parallel manner. In this context the challenges that arise with parallel execution of multiple plans have been discussed. In [1] summaries of concurrent hierarchical plans are used to identify associated preconditions and effects that can be used to reduce backtracking. Their formalism however only considers propositional, STRIPS-style actions and is less expressive than GOLOG, in particular regarding loops and recursive procedures. Harland et al. [8] propose to perform clean-up steps when suspending and resuming plans. The corresponding clean-up methods have to be defined for every plan. In comparison, our *promises* are action-specific; the intermediate steps necessary to suspend and resume a certain task are then implicit.

Haythem et al. [11] use a sequence of acts called *cascade* to detect failures and to remember what is left to be done after an interruption. They resume a cascade exactly where stopped and handle failures as new interrupts. In comparison, we determine intermediate steps in advance to avoid failures caused by interruptions.

6 CONCLUSION

In this work we described the problem of task interruption and resumption. We presented a first formalization of an extension of INDIGOLOG called INTRGOLOG to address some of the simpler cases of interruption and resumption. The introduction of the semantics predicates *TTrans* and *TFinal* allow our programming language to handle a set of tasks instead of one program. We introduced new constructs to determine intermediate steps when switching a task: The concept of *promises* defines asserted or required conditions of a running or interrupting task and *Re-execution sequences* allow to repeat certain parts of a program in full.

The evaluation showed that INTRGOLOG can react quickly to new events in real-world scenarios. As an example, interrupting a clean up task with an object in hand and putting it somewhere on the way leads on average to a 15 % latency reduction compared to bringing the object to its destination first. After reacting to an interrupting task, the robot can resume previous tasks without user intervention. Thereby, INTRGOLOG enables the robot to adapt to changing demands and new events in a swift and reliable fashion.

Opportunities for future work lie for instance in comparing the estimated time necessary for switching to the time for finishing the current task. For example, Gianni et al. use an estimation of *switching costs* to decide whether to react to stimuli with task switching [6]. This would avoid unnecessary or inefficient task switching, thus making our approach more efficient. Another promising possibility would be to integrate continual planning in GOLOG [9] with our approach. This would allow to recover promises in a more generic fashion without specific procedures for each promise. It would also provide more robustness if keeping a promise fails, e.g., because a cup cannot be retrieved since it was moved by someone else.

ACKNOWLEDGEMENTS

This work was supported by the German National Science Foundation (DFG) research unit FOR 1513 on Hybrid Reasoning for Intelligent Systems (<http://www.hybrid-reasoning.org>).

REFERENCES

- [1] Bradley J Clement and Edmund H Durfee, 'Theory for coordinating concurrent hierarchical planning agents using summary information', in *AAAI Conference on Artificial Intelligence (AAAI)*, (1999).
- [2] Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque, 'ConGolog, a concurrent programming language based on the situation calculus', *Artificial Intelligence*, **121**(1), (2000).
- [3] Giuseppe De Giacomo, Yves Lespérance, Hector J. Levesque, and Sebastian Sardina, 'IndiGolog: A high-level programming language for embedded reasoning agents', *Multi-Agent Programming: Languages, Tools and Applications*, (2009).
- [4] Alexander Ferrein, Tim Niemueller, Stefan Schiffer, and Gerhard Lakemeyer, 'Lessons learnt from developing the embodied AI platform Caesar for domestic service robotics', in *AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI II*, (2013).
- [5] Alberto Finzi and Fiora Pirri, 'Switching tasks and flexible reasoning in the situation calculus', *Department of Computer and System Sciences Antonio Ruberti Technical Reports*, **2**(7), (2010).
- [6] Mario Gianni, Panagiotis Papadakis, and Fiora Pirri, 'Shifting and inhibition in cognitive control', in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) - Workshop on Cognitive Neuroscience Robotics*, (2012).
- [7] Henrik Grosskreutz and Gerhard Lakemeyer, 'cc-Golog: Towards more realistic logic-based robot controllers', in *AAAI Conference on Artificial Intelligence (AAAI)*, (2000).
- [8] James Harland, David N Morley, John Thangarajah, and Neil Yorke-Smith, 'Aborting, suspending, and resuming goals and plans in BDI agents', *Autonomous Agents and Multi-Agent Systems*, (2015).
- [9] Till Hofmann, Tim Niemueller, Jens Claßen, and Gerhard Lakemeyer, 'Continual planning in Golog', in *AAAI Conference on Artificial Intelligence (AAAI)*, (2016).
- [10] François Félix Ingrand, Raja Chatila, Rachid Alami, and Frédéric Robert, 'PRS: A high level supervision and control language for autonomous mobile robots', in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, (1996).
- [11] Haythem O. Ismail and Stuart C. Shapiro, 'Conscious error recovery and interrupt handling', in *Proc. of the Int. Conf. on Artificial Intelligence (ICAI)*, (2000).
- [12] Ryan F. Kelly and Adrian R. Pearce, 'Towards high-level programming for distributed problem solving', in *IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology (IAT)*, (2006).
- [13] Nathan Koenig and Andrew Howard, 'Design and use paradigms for gazebo, an open-source multi-robot simulator', in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, (2004).
- [14] Hector J. Levesque, Raymond Reiter, Yves Lesperance, Fangzhen Lin, and Richard B. Scherl, 'GOLOG: A logic programming language for dynamic domains', *The Journal of Logic Programming*, **31**(1), (1997).
- [15] Yisong Liu, Lili Dong, and Yamin Sun, 'Cooperation model of multi-agent system based on the situation calculus', in *IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology (IAT)*, (2006).
- [16] John McCarthy and Patrick Hayes, 'Some philosophical problems from the standpoint of artificial intelligence', *Readings in Artificial Intelligence*, (1969).
- [17] Anand S. Rao, 'AgentSpeak(L): BDI agents speak out in a logical computable language', in *Agents Breaking Away*, Springer, (1996).
- [18] Ray Reiter and Zheng Yuhua, 'Scheduling in the situation calculus: A case study', *Annals of Mathematics and Artificial Intelligence*, **21**(2-4), (1997).
- [19] Raymond Reiter, *Knowledge in action: logical foundations for specifying and implementing dynamical systems*, MIT press, 2001.
- [20] Stefan Schiffer, Andreas Wortmann, and Gerhard Lakemeyer, 'Self-Maintenance for Autonomous Robots controlled by ReadyLog', in *IARP Workshop on Technical Challenges for Dependable Robots in Human Environments*, (2010).
- [21] Andrew S. Tanenbaum, *Modern operating systems*, volume 3, Pearson Prentice Hall, 2009.

Constructing Hierarchical Task Models Using Invariance Analysis

Damir Lotinac and Anders Jonsson¹

Abstract. Hierarchical Task Networks (HTNs) are a common model for encoding knowledge about planning domains in the form of task decompositions. We present a novel algorithm that uses invariant analysis to construct an HTN from the PDDL description of a planning domain and a single representative instance. The algorithm defines two types of composite tasks that interact to achieve the goal of a planning instance. One type of task achieves fluents by traversing invariants in which only one fluent can be true at a time. The other type of task applies a single action, which first involves ensuring that the precondition of the action holds. The resulting HTN can be applied to any instance of the planning domain, and is provably sound. We show that the performance of our algorithm is comparable to algorithms that learn HTNs from examples and use added knowledge.

Introduction

Hierarchical Task Networks, or HTNs, are a popular tool for encoding hierarchical structure into planning domains. In the past, HTNs have been successfully used in a variety of planning applications: military planning [26], Web service composition [32], unmanned air vehicle control [24], strategic game playing [30, 23], personalized patient care [29] and business process management [9], to name a few.

Although HTNs are at least as expressive as STRIPS planning [6], the main reason for their popularity is that they restrict actions choices. By excluding portions of the state space, search proceeds more quickly towards the goal, or in HTN terminology, towards generating a valid expansion of the initial task list. In the extreme case, each task has a single possible expansion, and planning is reduced to a simple traversal of the task hierarchy. Properly designing an HTN can be a time-consuming task for a human expert, but once this work is done, there is little need to optimize search.

Another powerful characteristic of HTNs is that tasks are parameterized, making it possible to encode knowledge about an entire planning domain, not just individual planning instances. Although identifying effective decomposition strategies for all instances of a domain can be arduous, once an HTN has been constructed for a planning domain, it can be used to solve an entire family of instances more efficiently.

Thus, the main reason that HTNs are frequently used in real-world applications is that they offer a potent mechanism for reducing the search effort required to solve a family of large-scale planning instances. This is also the reason that HTNs were so successful in the hand tailored track of early incarnations of the International Planning Competition (IPC): the participants were given access to the

planning domains beforehand and designed HTNs that effectively narrowed the search to a tiny portion of the state space. It is not a coincidence that the HTN planner that achieved the largest coverage at IPC-2002 and was for a long time regarded as the state-of-the-art in HTN planning, SHOP2 [27], performs blind search in the task space to compute a valid expansion. Most of the work required to reduce the search effort is performed while designing the HTN, and once this work is done, there is little need to optimize search to solve each instance efficiently.

In summary, HTNs offer a mechanism for human experts to encode prior knowledge about a planning domain. Typically, this requires many hours of fine-tuning, debugging and testing. This is fine for specific planning applications in which the initial effort is compensated by the subsequent reduction in search time during successive applications of the planner. However, a large body of research in the planning community is dedicated to finding domain-independent approaches to planning. Traditionally, HTNs have not found a place in this body of research because of their domain-dependent emphasis.

A key motivation for this work is to explore whether it is possible to generate HTNs automatically in a domain-independent way. We also want to investigate whether such HTNs offer benefits similar to those designed by human experts. In the literature there exist two techniques that construct HTNs automatically [11, 35]. These techniques rely on annotated traces of plans that solve a set of instances from a domain.

In this paper we present a novel algorithm for generating HTNs automatically. Our algorithm takes as input the PDDL description of a planning domain and a single representative instance. Unlike previous approaches, the algorithm does not require solution plans for a subset of instances of the domain. Instead, our approach is to generate HTNs that encode invariant graphs of planning domains. An invariant graph is similar to a lifted domain transition graph, but can be subdivided on types. To traverse an invariant graph we define two types of tasks: one that reaches a certain node of an invariant graph, achieving the associated fluent, and one that traverses a single edge of an invariant graph, applying the associated action. These two types of tasks are interleaved, in that the expansion of one type of task involves tasks of the other type.

In experiments we test our approach on planning benchmarks from the International Planning Competition (IPC) and on the instances used in experiments with HTN-MAKER [11]. To solve the instances we use the SHOP2 planner [27], which performs blind search in the task space to compute a valid expansion. We show that the HTNs constructed by our algorithm solve all training and test set instances used to evaluate HTN-MAKER.

The rest of the paper is organized as follows. We first provide a

¹ Universitat Pompeu Fabra, Roc Boronat 138, 08018 Barcelona, Spain. Email: {damir.lotinac, anders.jonsson}@upf.edu.

background on planning and HTNs. We then introduce the concept of invariant graphs, and show how to generate invariant graphs for a given planning domain. We then describe our base algorithm for constructing HTNs. Next, we describe several optimizations on top of the base algorithm, and present experimental results. We conclude with a discussion of related work and possible directions for future work.

Planning

We consider the fragment of PDDL 2.1 [7] that models typed STRIPS planning domains with positive preconditions and goals. To represent planning domains we adapt a definition based on function symbols [1]. Given a set X , let X^n denote the set of vectors of length n whose elements are symbols in X . Given such a vector $x \in X^n$, let $x_k \in X$, $1 \leq k \leq n$, denote the k -th element of x .

We distinguish between typed and untyped function symbols. An untyped function symbol f has associated arity $\alpha(f)$. In addition, a typed function symbol f has an associated type list $\beta(f) \in \mathcal{T}^{\alpha(f)}$, where $\mathcal{T} = \{\tau_1, \dots, \tau_n\}$ is a set of types. Let F be a set of function symbols, and let $\Sigma = \Sigma_1 \cup \dots \cup \Sigma_n$ be a set of objects, where Σ_i , $1 \leq i \leq n$, is a set of objects of type τ_i . We define $F_\Sigma = \{f[x] : f \in F, x \in \Sigma^{\alpha(f)}\}$ as the set of new objects obtained by applying each function symbol in F to each vector of objects in Σ of the appropriate arity. If f is typed, $f[x]$ has to satisfy the additional constraint $x_k \in \Sigma_{\beta_k(f)}$ for each k , $1 \leq k \leq \alpha(f)$, i.e. the type of each element in x is determined by the type list $\beta(f)$ of f .

Let f and g be two function symbols in F . An *argument map from f to g* is a function $\varphi : \Sigma^{\alpha(f)} \rightarrow \Sigma^{\alpha(g)}$ mapping arguments of f to arguments of g . An argument map φ allows us to map each object $f[x] \in F_\Sigma$ to an object $g[\varphi(x)] \in F_\Sigma$. In PDDL, argument maps have a restricted form: each element in $\varphi(x)$ is either an element from x or a constant object in Σ independent of x . WLOG we assume that argument maps are well-defined for typed function symbols.

A planning domain is a tuple $\mathbf{d} = \langle \mathcal{T}, <, P, A \rangle$, where $\mathcal{T} = \{\tau_1, \dots, \tau_n\}$ is a set of types, $<$ is an inheritance relation on types, P is a set of typed function symbols called *predicates*, and A is a set of typed function symbols called *actions*. Each action $a \in A$ has a set of preconditions $\text{pre}(a)$, a set of add effects $\text{add}(a)$ and a set of delete effects $\text{del}(a)$. Each element in these three sets is a pair (p, φ) consisting of a predicate $p \in P$ and an argument map φ from a to p .

Given \mathbf{d} , a planning instance is a tuple $\mathbf{p} = \langle \Omega, \text{init}, \text{goal} \rangle$, where $\Omega = \Omega_1 \cup \dots \cup \Omega_n$ is a set of objects of each type. The instance \mathbf{p} implicitly defines a set of fluents P_Ω and a set of grounded actions A_Ω . A grounded action $a[x] \in A_\Omega$ has a set of preconditions $\text{pre}(a[x])$, a set of add effects $\text{add}(a[x])$ and a set of delete effects $\text{del}(a[x])$. Each element in these sets is a fluent $p[\varphi(x)] \in P_\Omega$, where (p, φ) is the associated precondition or effect of the action a . The initial state $\text{init} \in P_\Omega$ and goal state $\text{goal} \in P_\Omega$ are both subsets of fluents. We often abuse notation by dropping the argument x of elements in P_Ω and A_Ω .

A state $s \subseteq P_\Omega$ is a subset of fluents that are true, while fluents in $P_\Omega \setminus s$ are false. A grounded action $a \in A_\Omega$ is applicable in s if and only if $\text{pre}(a) \subseteq s$, and the result of applying a in s is a new state $s \times a = (s \setminus \text{del}(a)) \cup \text{add}(a)$. A plan for \mathbf{p} is a sequence of grounded actions $\pi = \langle a_1, \dots, a_m \rangle$ such that a_j , $1 \leq j \leq m$, is applicable in $\text{init} \times a_1 \times \dots \times a_{j-1}$, and π solves \mathbf{p} if it reaches the goal state, i.e. if $\text{goal} \subseteq \text{init} \times a_1 \times \dots \times a_m$.

We use the LOGISTICS domain to illustrate the PDDL definition of a planning domain and instance. The types and predicates of LOGISTICS are given by

```
(:types truck airplane – vehicle
  package vehicle – physobj
  airport location – place
  city place physobj – object)

(:predicates (incity ?loc – place ?city – city)
  (at ?obj – physobj ?loc – place)
  (in ?pkg – package ?veh – vehicle)).
```

An example action for LOGISTICS is given by

```
(:action loadtruck
:parameters (?p – pkg ?t – truck ?l – place)
:precondition (and (at ?t ?l) (at ?p ?l))
:effect (and (not (at ?p ?l)) (in ?p ?t))).
```

We use the following LOGISTICS instance as a running example:

```
(:objects a1 – airplane ap1 ap2 – airport
  c1 c2 – city l1 l2 – location
  t1 t2 – truck p1 – package)

(:init (at p1 l1) (at a1 ap2)
  (at t1 l1) (incity l1 c1) (incity ap1 c1)
  (at t2 l2) (incity l2 c2) (incity ap2 c2))

(:goal (and (at p1 ap1))).
```

Hierarchical Task Networks

Our HTN definition is inspired by Geier and Bercher [8]. However, just as for STRIPS planning, we separate the definition into a domain part and an instance part. We also impose additional restrictions: a task network can contain at most one copy of each task, and task decomposition is limited to *progression*, always decomposing tasks with no predecessor.

An HTN domain is a tuple $\mathbf{h} = \langle P, A, C, M \rangle$ consisting of four sets of untyped function symbols. Specifically, P is the set of *predicates*, A is the set of *actions* (i.e. primitive tasks), C is the set of *compound tasks* and M is the set of *decomposition methods*. Predicates and actions are defined as for STRIPS domains but, unlike STRIPS domains, HTN domains are untyped and we allow negative preconditions.

Each method $m \in M$ has an associated tuple $\langle c, tn_m, \text{pre}(m) \rangle$ where $c \in C$ is a compound task with the same arity as m , tn_m is a *task network* and $\text{pre}(m)$ is a set of preconditions, defined as for actions. The task network $tn_m = \langle T, \prec \rangle$ consists of a set T of pairs (t, φ) , where $t \in A \cup C$ is a task and φ is an argument map from m to t , and a partial order \prec on the tasks in T .

Given an HTN domain \mathbf{h} , an HTN instance is a tuple $\mathbf{s} = \langle \Omega, \text{init}, tn_I \rangle$, where Ω is a set of objects and init is an initial state. The instance \mathbf{s} induces sets P_Ω and A_Ω of fluents and grounded actions, and sets C_Ω and M_Ω of grounded compound tasks and grounded methods, respectively. A grounded method $m[x] \in M_\Omega$ has associated tuple $\langle c[x], tn_m[x], \text{pre}(m[x]) \rangle$, where $c[x]$ is a grounded compound task and the precondition $\text{pre}(m[x])$ is derived as for grounded actions. The grounded task network $tn_m[x] = \langle T_x, \prec \rangle$ is defined by $T_x = \{t[\varphi(x)] : (t, \varphi) \in T\}$. The initial grounded task network $tn_I = \langle \{t_I\}, \emptyset \rangle$ contains a single grounded compound task $t_I \in C_\Omega$.

An HTN state (s, tn) consists of a state $s \subseteq P_\Omega$ on fluents and a grounded task network tn . We use $(s, tn) \rightarrow_D (s', tn')$ to denote that an HTN state decomposes into another HTN state, where $tn = \langle T_x, \prec \rangle$ and $tn' = \langle T_y, \prec' \rangle$. A valid *progression decomposition* consists in choosing a grounded task $t \in T_x$ such that $t' \not\prec t$ for each $t' \in T_x$, and applying one of the following rules:

1. If t is primitive, the decomposition is applicable if $\text{pre}(t) \subseteq s$, and the resulting HTN state is given by $s' = s \times t$, $T_y = T_x \setminus \{t\}$ and $\prec' = \{(t_1, t_2) \in \prec \mid (t_1, t_2) \in T_y\}$.
2. If t is compound, a grounded method $m = \langle t, tn, \text{pre}(m) \rangle$ with $tn = (T_m, \prec_m)$ is applicable if $\text{pre}(m) \subseteq s$, and the resulting HTN state is given by $s' = s$, $T_y = T_x \setminus \{t\} \cup T_m$ and

$$\begin{aligned} \prec' &= \{(t_1, t_2) \in \prec \mid (t_1, t_2) \in T_y\} \\ &\cup \{(t', t_1) \in T_m \times T_y \mid (t, t_1) \in \prec\} \cup \prec_m . \end{aligned}$$

The first rule removes a grounded primitive task t from tn and applies the effects of t to the current state, while the second rule uses a grounded method m to replace a grounded compound task t with tn_m while leaving the state unchanged. If there is a finite sequence of decompositions from (s_1, tn_1) to (s_n, tn_n) we write $(s_1, tn_1) \rightarrow_D^* (s_n, tn_n)$. An HTN instance s is solvable if and only if $(\text{init}, tn_I) \rightarrow_D^* (s_n, (\emptyset, \emptyset))$ for some state s_n , i.e. the initial HTN state (init, tn_I) is decomposed into an empty task network. Let π be the sequence of grounded actions extracted during such a decomposition; π corresponds to a *plan* that results from solving s .

Invariants

In STRIPS planning, an exactly-1 invariant is a subset of fluents $F' \subseteq P_\Omega$ such that exactly one fluent in F' is true at any moment. Formally, $|F' \cap \text{init}| = 1$ and any grounded action $a \in A_\Omega$ that adds a fluent in F' deletes another. The Fast Downward planning system [10] uses the domain description of a STRIPS domain to detect lifted invariant candidates. Unlike Fast Downward, which grounds lifted invariants on actual instances, our algorithm operates directly on the lifted invariants.

In LOGISTICS, Fast Downward finds a single lifted invariant candidate $\{(\text{in } ?o \ ?v), (\text{at } ?o \ ?p)\}$, i.e. a set of predicates with associated arguments. In the given invariant, variable $?o$ is *bound* while variables $?v$ and $?p$ are *free*. To ground the lifted invariant on an instance p , we should create one mutex invariant F' for each assignment of objects to the bound variables, obtaining each fluent in F' by assigning objects to the free variables. In our running example, assigning the package $p1$ to $?o$ results in the following grounded mutex invariant:

$$\{(\text{at } p1 \ \text{ap1}), (\text{at } p1 \ \text{ap2}), (\text{at } p1 \ \text{l1}), (\text{at } p1 \ \text{l2}), (\text{in } p1 \ \text{t1}), (\text{in } p1 \ \text{t2}), (\text{in } p1 \ \text{a1})\}.$$

The meaning of the invariant is that across all LOGISTICS instances, a given object $?o$ is either in a vehicle or at a location.

If a predicate $p \in P$ is not part of any invariant but there are actions that add and/or delete p , we create a new lifted invariant $\{(p \ ?o1 \ \dots \ ?ok), (\neg p \ ?o1 \ \dots \ ?ok)\}$. In this invariant, all variables $?o1, \dots, ?ok$ are bound and an associated fluent can either be true or false.

Given a lifted invariant, our algorithm generates one or several invariant graphs. We do so by iterating over the actions of the domain and identifying which actions add and delete predicates in the same lifted invariant. When grounded, such actions have the effect of *changing* the fluent of an exactly-1 invariant that is currently true. An invariant graph is a representation of a lifted invariant in which the nodes are the predicates of the invariant and the edges are the actions used to change the predicate that is currently true. We use invariant graphs to infer which actions to perform in order to achieve a particular fluent of an exactly-1 invariant.

The reason why a given lifted invariant can generate multiple invariant graphs is that the *type* of the bound objects may be differ-

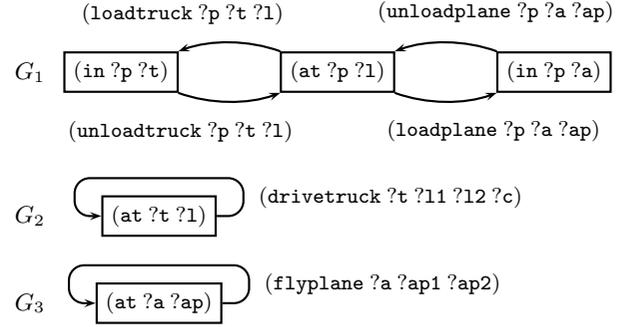


Figure 1. Invariant graphs G_1 , G_2 and G_3 in LOGISTICS.

ent for different actions. For example, in the LOGISTICS domain, all actions affect the lone invariant above. However, in the actions for loading or unloading a package, the bound object $?o$ is a package, in the action for driving a truck $?o$ is a truck, and in the action for flying an airplane $?o$ is an airplane. Moreover, we can either load a package into a truck or an airplane. We use the actions to differentiate between types, possibly generating multiple invariant graphs for each lifted invariant.

To generate the invariant graphs induced by lifted invariants we go through each action, find each transition of each invariant that it induces (by pairing add and delete effects and testing whether the bound objects are identical), and map the types of the predicates to the invariant. We then either create a new invariant graph for the bound types or add nodes to an existing graph corresponding to the mapped predicate arguments.

Figure 1 shows the invariant graphs that we generate in LOGISTICS. In the top graph (G_1), the bound object is a package $?p$, in the middle graph (G_2) it is a truck $?t$, and in the bottom graph (G_3) it is an airplane $?a$. Note that the predicate *in* is not actually part of the two bottom graphs, since trucks and planes cannot be inside other vehicles. Nevertheless, the invariant still applies: a truck or plane can only be at a single place at once.

Each edge of an invariant graph corresponds to an action that deletes one predicate of the invariant and adds another. To do so, the arguments of the action have to include the arguments of both predicates, including the bound objects. In the figure, the invariant notation is extended to actions on edges such that each argument of an action is either bound or free.

Even if actions preserve the invariant property, the initial state of a planning instance may violate the condition $|F' \cap \text{init}| = 1$, in which case F' is not an exactly-1 invariant. To verify that a lifted invariant candidate corresponds to actual exactly-1 invariants, our algorithm needs access to the initial state of an example planning instance p of the domain. If this verification fails, the lifted invariant is not considered by the algorithm.

Generating HTNs

In this section we describe our algorithm for automatically generating HTN domains. The idea is to construct a hierarchy of tasks that traverse the invariant graphs to achieve certain fluents. In doing so there are two types of interleaved tasks: one that achieves a fluent in a given invariant (which involves applying a series of actions to

traverse the edges of the graph), and one that applies the action on a given edge (which involves achieving the preconditions of the action).

Formally, our algorithm takes as input a STRIPS planning domain $\mathbf{d} = \langle \mathcal{T}, <, P, A \rangle$ and a planning instance $\mathbf{p} = \langle \Omega, \text{init}, \text{goal} \rangle$ and outputs an HTN domain $\mathbf{h} = \langle P', A', C, M \rangle$. The HTN domain \mathbf{h} can then be used to solve any other instance of the domain. Specifically, for each instance \mathbf{p}' of the planning domain \mathbf{d} , we construct an HTN instance s . Solving the HTN induced by \mathbf{d} and s returns a plan that can be adapted to solve \mathbf{p}' .

The input planning instance \mathbf{p} is used for three purposes:

1. To verify that an invariant candidate is actually an invariant by testing the condition $|F' \cap \text{init}| = 1$.
2. To extract a subset of predicates $P_G \subseteq P$ that are part of the goal.
3. To perform goal ordering as described in a subsequent section.

The algorithm first constructs the invariant graphs G_1, \dots, G_k described above. In what follows we describe the components of the HTN domain \mathbf{h} .

The set $P' \supseteq P$ extends P with the following predicates:

- For each predicate $p \in P$, a predicate `visited- p` with arity $\alpha(p)$ indicating that p has already been visited during search.
- For each predicate $p \in P$, a predicate `achieving- p` with arity $\alpha(p)$ indicating that p or another predicate in the same invariant are already being achieved.
- For each goal predicate $p \in P_G$, a predicate `goal- p` with arity $\alpha(p)$ indicating that a fluent derived from p is a goal state.
- For each type $\tau \in \mathcal{T}$, a type predicate τ with arity 1.

The set A' contains the following actions:

- Each action $a \in A$. For each element $\beta_k(a) \in \mathcal{T}$ of the type list of a , we add an additional precondition $(\beta_k(a), \varphi_k)$, where the argument map φ_k maps the argument x_k of a to the lone argument of the type predicate $\beta_k(a)$, ensuring that argument x_k has the correct type.
- For each $p \in P$, an action `visit- p` with arity $\alpha(p)$ that marks p as visited by adding `visited- p` .
- For each invariant graph G_i , an action `occupy- i` whose arity equals the number of bound objects in G_i , and that marks each predicate in G_i as being achieved.
- For each invariant graph G_i , an action `clear- i` whose arity equals the number of bound objects in G_i , and that deletes `visited- p` and `achieving- p` for each predicate p of G_i .
- For each goal predicate $p \in P_G$, an action `test- p` with arity 0 and no effects, whose precondition tests if *all* goal fluents derived from p hold.

Note that only actions in A add or delete predicates in the original set P . The set C contains four types of compound tasks:

- For each predicate $p \in P$, a task `achieve- p` with arity $\alpha(p)$.
- For each invariant graph G_i and each $p \in P$ that is positive in G_i , a task `achieve- $p-i$` with arity $\alpha(p)$.
- For each invariant graph G_i , each predicate p in G_i , and each outgoing edge of p (corresponding to an action $a \in A$), a task `do- $p-a-i$` with arity $\alpha(a)$.
- A task `solve` with arity 0.

The task `achieve- p` is a wrapper task that uses a task `achieve- $p-i$` to achieve p by traversing the edges of the invariant graph G_i . To

traverse each edge of G_i , `achieve- $p-i$` has to use a task of type `do- $p-a-i$` , which in turn uses tasks of type `achieve- p'` to achieve the preconditions of a . The task `solve` serves as the root task of the HTN and recursively achieves the goal by applying one task of type `achieve- p` at a time.

Finally, the set M contains the following decomposition methods. We describe methods in pseudo-SHOP2 syntax in the following format:

```
(:method ((name)[(arguments)])
  ((precondition))
  ((tasklist)))
```

For each method in the first line we specify name and arguments, in the second line we give precondition list, and finally in the third we specify task list to which method decomposes. For clarity, we add an exclamation mark in front of primitive tasks. The first type of compound task, `achieve- p` , has one associated method for each invariant graph G_i in which p appears. An outline of this method is given by

```
(:method (achieve- $p[x]$ )
  ((¬achieving- $p[x]$ )
  (!!occupy- $i[\varphi_i(x)]$ ) (achieve- $p-i[x]$ ) (!clear- $i[\varphi_i(x)]$ )).
```

The argument map φ_i maps the arguments of p to the bound variables of the invariant graph G_i . Intuitively this method delegates achieving p to the task `achieve- $p-i$` for some invariant graph G_i . The method first adds `achieving- p'` for each predicate p' in G_i , and clears the flags after achieving p . The precondition $(\neg\text{achieving-}p[x])$ prevents us from achieving p if it is part of an occupied invariant graph, which could potentially lead to an infinite recursion.

The second type of compound task, `achieve- $p-i$` , has one associated method for each predicate p' in the invariant graph G_i and each outgoing edge of p' (corresponding to an action a):

```
(:method (achieve- $p-i[x]$ )
  (( $p'[\varphi'(x)]$ ) (¬visited- $p'[\varphi'(x)]$ )
  (!!visit- $p'[\varphi'(x)]$ ) (do- $p'-a-i[\varphi_a(x)]$ ) (achieve- $p-i[x]$ )).
```

Action a appears on an outgoing edge from p' , i.e. a deletes p' . Intuitively, one way to achieve p in G_i , given that we are currently at some different node p' , is to traverse the edge associated with a using the compound task `do- $p'-a-i$` . Before doing so we mark p' as visited to prevent us from visiting p' again. After traversing the edge we recursively achieve p from the resulting node. The argument map φ' should set the bound objects of p' while leaving other arguments of p' as free variables. Likewise, the argument map φ_a should set the bound objects of a . The precondition $(\neg\text{visited-}p'[\varphi'(x)])$ prevents us from visiting the same node p' twice. In essence, the result is a depth-first search through the invariant graph G_i , which stops when we reach p . Recall that the flags `visited- p` and `achieving- p` are cleared by the parent method once we reach p .

To stop the recursion we define a “base case” method:

```
(:method (achieve- $p-i[x]$ )
  (( $p[x]$ ))
  ()).
```

This method is applicable when p already holds and has empty task list.

The third type of compound task, `do- $p-a-i$` , has only one associated method. The aim is to apply action a to traverse an outgoing edge of p in the invariant graph G_i . To do so, the task list has to ensure that all preconditions p_1, \dots, p_k of a hold (excluding p , which holds by definition, as well as any static preconditions of a). We define the method as

```
(:method (do-p-a-i[x])
  ((p[φ(x)]))
  (((achieve-p1[φ1(x)])) ··· (achieve-pk[φk(x)])) (!a[x]))
```

Here, (p, φ) is the precondition of a associated with p , while (p_i, φ_i) , $1 \leq i \leq k$, are the remaining preconditions of a . The decomposition achieves all preconditions of a , then applies a . Note that if action a has no preconditions except p , the mutual recursion stops since the decomposition does not contain any task of type `achieve- p_i` . In this case our approach is to simplify the definition of other methods by replacing any instance of `do-p-a-i` with the action a itself.

The fourth type of compound task, `solve`, has one method for each predicate $p \in P_G$ that appears in the goal state:

```
(:method (solve)
  ((goal-p[x])(¬(p[x])))
  ((achieve-p[x])(solve)))
```

Here, x are free parameters. Whenever a predicate in the goal state does not hold, we achieve it and recursively call `solve`. Again, we need a base case method to stop the recursion:

```
(:method (solve)
  ()
  ((!test-p1) ··· (!test-pk))
```

Here, p_1, \dots, p_k are the predicates in P_G , i.e. those that appear in the goal state. Since each action `test- p_i` , $1 \leq i \leq k$, checks whether all goal fluents associated with p_i hold, this method is only applicable when the goal state holds. The reason this check is not made in the precondition of the method itself is that SHOP2 only supports forall clauses in action preconditions, not method preconditions.

To restrict the choices made when traversing the HTN, we impose a total order on all task lists of methods, except the tasks for achieving the preconditions of a in `do-p-a-i`. The reason is that it may be difficult to determine in which order to achieve the preconditions of an action.

Planning Instances

Once we have generated the HTN domain h we can apply it to any instance of the domain. Given a STRIPS instance $\mathbf{p} = \langle \Omega, \text{init}, \text{goal} \rangle$, we construct an HTN instance $s = \langle \Omega, \text{init}', \langle \text{solve}, \emptyset \rangle \rangle$ as follows. The set of objects $\Omega = \Omega_1 \cup \dots \cup \Omega_n$ is identical to that of \mathbf{p} . The initial state init' is defined as

$$\text{init}' = \text{init} \cup \{\tau_j[\omega] : \tau_j \in \mathcal{T}, \omega \in \Omega_j\} \cup \{\text{goal-p}[x] : p[x] \in \text{goal}\}.$$

We thus mark the type τ_j of each object ω using the fluent $\tau_j[\omega]$, and we mark all fluents $p[x]$ in the goal state using the fluent `goal- $p[x]$` . The initial task network contains the single grounded compound task `solve`.

We show that the HTN translation is sound. The translation is only complete if we allow goals and preconditions to be achieved in any order; we violate this condition below.

Theorem 1 *Let π be a plan that results from a valid decomposition for s , and construct π' by removing grounded actions of type `visit- p` , `test- p` , `occupy- i` and `clear- i` . Then π' solves \mathbf{p} .*

Proof sketch Recall that an HTN state is given by (s, tn) for a planning state s and task network tn . When we decompose an HTN state, s is only changed if we use primitive tasks, i.e. changes to s are caused precisely by the elements of π . Consider the fluent set P_Ω induced by \mathbf{p} . The grounded actions of the types removed do not add or delete fluents in P_Ω . Conversely, the grounded actions that remain

in π' are those of the original action set A_Ω , which *only* add or delete fluents in P_Ω .

Let $\pi' = \langle a_1, \dots, a_m \rangle$ be the sequence of grounded actions from A_Ω . We can only decompose an HTN state (s, tn) using a primitive task a_j , $1 \leq j \leq m$, if the precondition of a_j holds in s . It follows that $\text{pre}(a_j) \subseteq \text{init} \times a_1 \times \dots \times a_{j-1}$ for each j , $1 \leq j \leq m$. Hence π' is a plan for \mathbf{p} .

To prove that the plan π' solves \mathbf{p} , it remains to show that the goal state `goal` of \mathbf{p} holds after applying π' in `init`. Since `solve` is recursively applied during HTN decomposition, the sequence π has to have suffix $\langle \text{test-}p_1, \dots, \text{test-}p_k \rangle$, the decomposition of the only base case method for `solve`. This action sequence has no effects and is only applicable if goal holds. Hence goal has to hold after the last action in π' since the removed actions have no effects on fluents in P_Ω .

Example

In LOGISTICS, our algorithm generates two wrapper tasks `achieve-in` and `achieve-at`, and four tasks `achieve-in-1`, `achieve-at-1`, `achieve-at-2`, and `achieve-at-3`, corresponding to the predicates in the three invariant graphs. The task `achieve-at-1` has five associated methods: one for each edge of the graph G_1 , plus the base case method.

The algorithm also generates six tasks `do-at-loadtruck-1`, `do-at-loadplane-1`, `do-at-unloadtruck-1`, `do-at-unloadplane-1`, `do-at-drivetruck-2`, and `do-at-flyplane-3`, corresponding to the six edges of the graphs. The latter two do not have preconditions besides `at` (the predicate `incity` in the precondition of `drivetruck` is static). The remaining four tasks each achieve a single precondition: the truck or plane being at the associated place.

To illustrate the tasks and associated methods we sketch the task expansions of the HTN instance generated from our running example. The only goal is `(at p1 ap1)`, so the task `solve` has a single valid decomposition that contains the task `(achieve-at p1 ap1)`. Table 1 shows the first five task expansions of the HTN instance. In each case, the compound task to be decomposed is underlined, and the new tasks inserted as a result of the decomposition are colored in the next step.

The second decomposition is produced by the lone method for `(achieve-at p1 ap1)`. The current node associated with `p1` in G_1 is `(at p1 l1)`, with two outgoing edges, corresponding to actions `loadtruck` and `loadplane`. Applying the method for `(achieve-at-1 p1 ap1)` associated with `(at p1 l1)` and `loadtruck` produces the third expansion. The only method for `(do-at-loadtruck-1 p1 t1 l1)` expands to `(achieve-at t1 l1)`, which in turn expands to `(achieve-at-2 t1 l1)` (the last expansion shown).

Optimizations

In this section we discuss several optimizations of the base algorithm for generating HTNs.

Ordering Preconditions

Achieving the preconditions of an action a in any order is inefficient since an algorithm solving the HTN instance may have to backtrack repeatedly to find a correct order. For this reason, we include an extension of our algorithm that uses a simple inference technique to compute a partial order in which to achieve the preconditions of a .

<u>(solve)</u>	(achieve-at p1 ap1)	(!occupy-1 p1)	(!occupy-1 p1)	(!occupy-1 p1)	(!occupy-1 p1)
(solve)	(achieve-at-1 p1 ap1)	(!visit-at p1 l1)	(!visit-at p1 l1)	(!visit-at p1 l1)	(!visit-at p1 l1)
	(clear-1 p1)	(do-at-loadtruck-1 p1 t1 l1)	(achieve-at t1 l1)	(!occupy-2 t1)	
	(solve)	(achieve-at-1 p1 ap1)	(!loadtruck p1 t1 l1)	(achieve-at-2 t1 l1)	
		(!clear-1 p1)	(achieve-at-1 p1 ap1)	(!clear-2 t1)	
		(solve)	(!clear-1 p1)	(!loadtruck p1 t1 l1)	
			(solve)	(achieve-at-1 p1 ap1)	
				(!clear-1 p1)	
				(solve)	

Table 1. The first five task expansions of the HTN instance generated from the running example in LOGISTICS. The colored tasks are those added by the decomposition of the underlined task in the previous step.

```

1: function ORDER( $a, p$ )
2:    $V \leftarrow \text{pre}(a) \setminus \{p\}, Z \leftarrow \langle \rangle$ 
3:   repeat
4:     for  $p' \in V$  do
5:        $W \leftarrow \{p'\} \cup V \setminus \{p'\}$ 
6:       for each invariant graph  $G_j$  containing  $p'$  do
7:         Perform a backwards BFS in  $G_j$  from  $p'$ 
8:         Test if paths applicable when  $W$  persists
9:       end for
10:      if each path achieving  $p'$  is applicable then
11:         $V \leftarrow V \setminus \{p'\}$ 
12:         $Z \leftarrow \langle p', Z \rangle$ 
13:      end if
14:    end for
15:  until  $V, Z$  converge
16:  return ( $V, Z$ )
17: end function

```

Figure 2. Algorithm ordering all preconditions of a except p .

We define a subset of predicates whose value is supposed to persist, and check whether a path through an invariant graph is applicable given these persisting predicates. While doing so, only the values of the bound variables are known, while free variables can take on any value. Matching the bound variables of predicates and actions enables us to determine whether an action allows a predicate to persist.

Consider a task of type *do-p-a-i*, i.e. the purpose of the task is to apply action a in order to delete p . Figure 2 shows how to order all preconditions of action a except p . In the algorithm, V is the set of preconditions to be ordered, while Z is a sequence of preconditions, initially empty. The algorithm considers one precondition $p' \in V$ at a time and checks if it is possible to achieve p' while all remaining preconditions persist. If so, we remove p' from V and place it first in Z . We then iterate until no more preconditions can be removed from V , and return (V, Z) . In the method m associated with *do-p-a-i*, the preconditions in Z can be achieved in order. On the other hand, we cannot say anything about the order of preconditions that remain in V .

Goal Ordering

Just as for preconditions, achieving goals in any order results in significant backtracking. To order the goals we implement an algorithm similar to the one for ordering preconditions. While the ordered pre-

conditions are coded into the HTN, the goals are different for each instance of the domain. Since HTNs are instance-independent, our approach is to define new tasks that compute a goal ordering as a preprocessing step.

To accomplish this, we first order the goals of the representative planning instance p passed as input to the algorithm. We run the precondition ordering algorithm on the set of goal predicates $P_G \subseteq P$, i.e. predicates whose associated fluents appear in the goal. Given an ordering of the predicates in P_G , we then order the set of fluents of each predicate $p \in P_G$ using a similar algorithm. To do so, the invariant graphs need to be partially grounded on each pair of fluents to be ordered.

For each $p \in P_G$ and each pair of fluents $p[x]$ and $p[y]$, we check if $p[y]$ is achievable when $p[x]$ persists (i.e. we are not allowed to delete $p[x]$). Each invariant graph that contains p is partially grounded on $p[x]$, while the preconditions of actions that directly achieve p are grounded on $p[y]$. If this grounding violates the invariant, $p[y]$ should be ordered before $p[x]$. Once the invariant is invalidated by partial grounding, the algorithm stores the indices of the parameters of p that invalidated the invariant.

As an example, in the BLOCKS domain, $P_G = \{\text{on}\}$, so the method for *solve* always decomposes to *achieve-on*. Figure 3 shows one of the invariant graphs in BLOCKS that contains the predicate *on*. We test each pair of goal fluents to establish an order among them. Consider two goal fluents $(\text{on } a \ b)$ and $(\text{on } b \ c)$ that both refer to block b . If we fix $(\text{on } a \ b)$ and attempt to achieve $(\text{on } b \ c)$, the only operator $(\text{stack } b \ c)$ that directly achieves $(\text{on } b \ c)$ has precondition $(\text{holding } b)$, which violates the invariant since the fluent $(\text{on } a \ b)$ should persist. Thus fluent $(\text{on } b \ c)$ should be ordered before fluent $(\text{on } a \ b)$. We can generalize this knowledge and derive a rule that whenever two goal fluents of type *on* have the same object as the first and second parameter, respectively, the former should be ordered before the latter.

To implement the goal ordering mechanism we introduce a new compound task *order* with arity 1 whose associated method uses the ordering rule from above to order the goal fluents of a single predicate $p \in P_G$. The argument of *order* is a new object that encodes a *counter* that starts at 0, and the result of decomposing *order* is assigning a count to each fluent in the goal. We force *order* to be the first task expanded. We then add an argument to the *solve* task that represents a counter, and change the initial task network to contain $(\text{solve } 0)$, i.e. the initial count is 0. The recursive method for *solve* requires us to always achieve the goal fluent corresponding to the current count, and we are only allowed to increment the count whenever the current goal fluent already holds. To ensure that all goal fluents are eventually achieved we reset the counter to 0 each time we achieve a goal fluent.

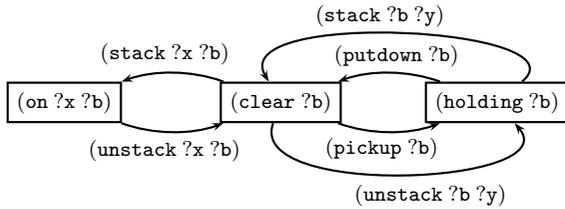


Figure 3. Invariant graph in the BLOCKS domain.

Sorting the Bindings of Free Variables

Apart from deciding which method to use to decompose a compound task and in which order to decompose partially ordered tasks in a task network (whenever the task network does not impose a total order on tasks), another non-deterministic choice made by SHOP2 is how to assign objects to the free variables of a method (e.g. the argument x of the predicate p in the recursive method for decomposing the `solve` task). Since SHOP2 performs blind search, it backtracks over all possible bindings to these free variables.

In the recursive method for decomposing the task `achieve-p-i`, the free variables determine which action to perform from the current node of the corresponding invariant graph. If we do not impose any order on variable bindings, SHOP2 iterates over such actions in an arbitrary fashion. As a consequence, the decompositions of `do-p-a-i` tasks do not immediately aim for the target node, even if this node is just one step away. We exploit the ability of SHOP2 to order the bindings of free variables by imposing an order that first attempts to traverse an edge to the target node. If this is not possible, blind search later explores all other possibilities.

Results

We ran our algorithm on all instances in the training and test set of HTN-MAKER (LOGISTICS, SATELLITE and BLOCKS). We also tested the algorithm in other STRIPS planning domains from the IPC-2000 and IPC-2002. The experiments were performed with two versions of our algorithm. The base algorithm that achieves the preconditions and goals in any order was slow in testing, so we activated precondition ordering in both versions. The first version, HTN-Prec, achieves the goals in the order they appear in the PDDL definition. The second version, HTN-Goal, implements our goal ordering and free variable sorting strategies in addition to precondition ordering. In all experiments we used JSHOP (the Java implementation of SHOP2), which uses blind search to compute a valid expansion. We used a memory limit of 4GB and a timeout of 1,800 seconds. We compare to the latest results of HTN-MAKER (WeakS), which uses a C++ implementation of SHOP2 and one hour of CPU time to solve the instances [12]. However, the instances from that paper are not publicly available, so we could not compare directly, and we just show their reported coverage results for reference.

Table 2 shows the coverage results on instances from HTN-MAKER’s experiments. We tested HTN-MAKER’s output domains from the fifth and final trial [11] and we ran it over the test set only, while we ran our algorithm over both the test and training sets of the experiment instances. To achieve the reported results, HTN-MAKER needed 420 training instances in BLOCKS and 75 training instances

	HTN-Prec	HTN-Goal	HTN-MAKER	WeakS
LOGISTICS	100 %	100%	100%	93,6%
SATELLITE	100 %	100%	92%	100%
BLOCKS	100 %	100%	63,5%	99%
ROVERS	100 % *	100% *	-	100%
ZENOTRAVEL	20% *	100% *	-	99,8%

Table 2. Coverage of instances from HTN-MAKER’s experiments. Scores marked with an asterisk (*) are taken from IPC instances.

in LOGISTICS and SATELLITE, while our algorithm used the domain and a single example instance with no need to solve the instance or annotating a plan beforehand. For ROVERS and ZENOTRAVEL we report coverage over the IPC-2002 instances for HTN-Prec and HTN-Goal. The improvement of HTN-Goal over HTN-Prec in ZENOTRAVEL is due to free variable sorting (since airplanes can fly directly to the target destination). HTN-Goal performs better in other domains as well; however, in terms of coverage of HTN-MAKER’s instances there is little difference.

The instances used in HTN-MAKER’s experiments are generated by a random generator with very few objects (e.g. a maximum of five blocks in BlocksWorld), while our HTNs can be used to solve much larger instances. Therefore we ran both the HTN-Prec and HTN-Goal versions of our algorithm on other benchmark instances from IPC-2000 and IPC-2002. The results from these experiments appear in Table 3. For example, in BLOCKS, IPC benchmark instances include up to 50 blocks. On those benchmark instances HTN-Goal achieves full coverage of BLOCKS, SATELLITE, ROVERS, LOGISTICS, ZENOTRAVEL and MICONIC. The improvement in average number of backtracks is most clearly visible in the BLOCKS domain, and is due to our goal ordering strategy. It also allowed for an increase in number of instances solved in the DEPOTS domain, as these two domains are the most sensitive to goal ordering. The combination of goal ordering and free variable sorting shows an increase in performance in other domains as well. This allows HTN-Goal to solve all instances in many domains, with very few backtracks. These domains however tend to be the ones with a lower branching factor in the invariant graphs.

It is important to note that unsolved instances are not due to failed decompositions. Rather, the allotted time was insufficient to complete the search. While the results of our approach are comparable to those of HTN-MAKER, in some domains the generated HTNs do not perform well due to excessive backtracking (e.g. we do not solve any instances of the IPC-2002 FREECCELL domain, which is more puzzle-like and therefore harder to serialize as our HTNs do by achieving one goal fluent at a time). On the other hand in DRIVERLOG, the algorithm does not solve many instances due to our goal ordering strategy. In this domain, using the lifted invariant graphs finds only goal predicate orderings, which is insufficient, and thus achieved goals often have to be unachieved by the `solve` task. Apart from that, in this domain, a system of “link-paths” is used for both drivers and trucks, which means that a path needs to be found (if one exists) for each location that happens to be a goal or subgoal location.

Related Work

Our approach to generating HTNs from a single planning instance and using them to solve larger instances of the same planning domain can be viewed as a form of generalized planning, which has received

Domain	Instances	HTNPrec			HTNGoal		
		#s	t	#b	#s	t	#b
Freecell	60	0	-	-	0	-	-
Blocks	103	24	91	8118	103	0.6	0.4
Rovers	20	20	0.6	1.2	20	0.5	1.1
Logistics	80	80	2.9	44	80	2.4	23.7
Driverlog	20	0	-	-	3	0.4	1.3
Zenotravel	20	4	25.6	4101	20	0.5	0.3
Miconic	150	150	0.66	0	150	0.63	0
Satellite	20	7	0.59	1.2	20	0.37	0.04
Depots	22	8	22.6	1867	17	88.4	8108

Table 3. Results in the IPC-2000 and IPC-2002 domains, with the total number of instances of each domain shown in brackets. For each solver we report number of solved instances (#s), average time in seconds (t) and average number of backtracks in thousands (#b) respectively

a lot of recent attention, most notably in the form of the learning track of the IPC. One popular approach to generalized planning is to identify macros [2, 20, 25, 28], i.e. sequences of operators that frequently appear in the solutions to example instances. Once identified, such macros can then be inserted into the action space of larger instances in order to speed up search. However, even though macros can be parameterized, they do not offer the same flexibility as HTNs in terms of representing a solution to all instances of a planning domain.

Another approach to generalized planning is to learn reactive policies for planning domains [15, 17, 22, 33]. A third approach is to learn domain-specific knowledge in order to improve heuristic estimates computed during search [4, 34]. In contrast to most of these techniques, which are *inductive*, ours is a *generative* approach, as we construct HTNs directly by analyzing the domain model.

Achieving fluents by traversing the edges of domain transition graphs is the strategy used by DTGPlan [3] and similar algorithms. There also exist other inference techniques that can solve many individual instances backtrack-free [18, 19]. The novelty of our approach compared to previous work is the ability to do this in an instance-independent way.

The most popular approaches for generating hierarchical task models are learning from examples. However these approaches not only need to learn from examples but also rely on some given task-subtask decompositions [14], annotated plans [11], or added concepts [16]. In contrast, our algorithm requires less semantic information, and although we could use the representative instance (or multiple instances) to generate solution plans for learning, these plans would still have to be annotated to be used by HTN-MAKER.

A basic compilation from STRIPS to HTN was defined by Erol et al [6]. This compilation constructs primitive tasks for each STRIPS operator and a single compound task. However this compilation is used only for theoretical purposes, as it does not impose more restrictions on the task network. Our work is also related to other approaches to hierarchical planning [13, 21, 5], with the difference that we generate the hierarchies automatically.

Conclusion

In this paper we present a domain-independent algorithm for generating HTNs. All the algorithm needs is a PDDL description of the planning domain and a single representative instance, which is needed for three reasons: 1) to validate invariant candidates; 2) to determine which predicates appear in the goal state; and 3) to establish a goal ordering. While other approaches learn from solved

instances, ours can be viewed as a form of compilation. We show that our algorithm is competitive with state-of-the-art algorithms for automatically learning HTNs from solved instances. The algorithm is not complete if we do not allow for preconditions and goals to be achieved in any order, which is the case in our experimental setting.

While the results of our approach are comparable to those of HTN-MAKER, in some domains due to the branching factor of the invariant graphs the generated HTNs do not perform well. In FREECCELL, apart from a high branching factor, the domain used in the competition is encoded so that it uses auxiliary predicates like `number` and `successor`. This causes a high arity of the actions (implying that there are more possible bindings of objects to the argument of actions), and is used to impose an order for many different stacks. It is possible that our approach would solve at least a few instances if a different representation were used.

Although the success of the algorithm is limited in some domains, we believe that there are still many potential benefits. The algorithm takes a fraction of a second to generate HTNs given a PDDL domain and a single example instance. The example instance does not need to be solved, and no plan traces are required. Since the algorithm is domain-independent it does not require any intervention and is easy to run. Therefore the resulting HTN could potentially be useful even in cases where it does not perform well right after the compilation, e.g. by extracting useful subtasks.

The avenue for future research that we find most promising is to test different restrictions on the invariant graphs. If the representative instance can still be solved under some restriction, the resulting HTN may still be able to solve other instances, and the restriction has the effect of reducing the branching factor. In essence, this mechanism would reduce the number of ways to traverse the invariant graphs. Another possible extension is to identify and prune methods that are not needed to solve the HTN instances.

User specified heuristics for HTNs have shown useful for automatically generated HTNs [31]. For example such heuristic would easily resolve path-finding problems in domains like DRIVERLOG. Therefore another option is to construct heuristics, which would be used to guide task-subtask decompositions and sort the bindings of free variables in order to direct search more efficiently.

Acknowledgment

This work is partially supported by the MINECO/FEDER grant TIN2015-67959 and the Maria de Maeztu Units of Excellence Programme MDM-2015-0502, MEC, Spain.

REFERENCES

- [1] C. Bäckström, A. Jonsson, and P. Jonsson, 'Automaton Plans', *Journal of Artificial Intelligence Research*, **51**, 255–291, (2014).
- [2] A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer, 'Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators', *Journal of Artificial Intelligence Research*, **24**, 581–621, (2005).
- [3] Y. Chen, R. Huang, and W. Zhang, 'Fast Planning by Search in Domain Transition Graphs', in *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI'08)*, pp. 886–891, (2008).
- [4] T. de la Rosa, S. Jiménez, R. Fuentetaja, and D. Borrajo, 'Scaling up Heuristic Planning with Relational Decision Trees', *Journal of Artificial Intelligence Research*, **40**, 767–813, (2011).
- [5] M. Elkawagy, P. Bercher, B. Schattenberg, and S. Biundo, 'Improving Hierarchical Planning Performance by the Use of Landmarks', in *Proceedings of the 26th National Conference on Artificial Intelligence (AAAI'12)*, (2012).
- [6] K. Erol, J. Hendler, and D. Nau, 'HTN planning: Complexity and expressivity', in *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, pp. 1123–1128, (1994).
- [7] Maria Fox and Derek Long, 'Pddl2. 1: An extension to pddl for expressing temporal planning domains.', *J. Artif. Intell. Res.(JAIR)*, **20**, 61–124, (2003).
- [8] T. Geier and P. Bercher, 'On the Decidability of HTN Planning with Task Insertion', in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, pp. 1955–1961, (2011).
- [9] A. González-Ferrer, J. Fernández-Olivares, and L. Castillo, 'From Business Process Models to Hierarchical Task Network Planning Domains', *Knowledge Engineering Review*, **28**(2), 175–193, (2013).
- [10] M. Helmert, 'Concise finite-domain representations for PDDL planning tasks', *Artificial Intelligence*, **173**, 503–535, (2009).
- [11] C. Hogg, H. Munoz-Avila, and U. Kuter, 'HTN-MAKER: Learning HTNs with Minimal Additional Knowledge Engineering Required', in *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI'08)*, pp. 950–956, (2008).
- [12] Chad Hogg, Héctor Muñoz-Avila, and Ugur Kuter, 'Learning hierarchical task models from input traces', *Computational Intelligence*, (2014).
- [13] R. Holte, M. Perez, R. Zimmer, and A. MacDonald, 'Hierarchical A*: Searching Abstraction Hierarchies Efficiently', in *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*, pp. 530–535, (1996).
- [14] Okhtay Ilghami, Dana S Nau, Héctor Muñoz-Avila, and David W Aha, 'Camel: Learning method preconditions for htn planning', in *AIPS*, pp. 131–142, (2002).
- [15] R. Khardon, 'Learning Action Strategies for Planning Domains', *Artificial Intelligence*, **113**(1-2), 125–148, (1999).
- [16] Pat Langley and Dongkyu Choi, 'Learning recursive control programs from problem solving', *The Journal of Machine Learning Research*, **7**, 493–518, (2006).
- [17] J. Levine and D. Humphreys, 'Learning Action Strategies for Planning Domains Using Genetic Programming', in *EvoWorkshops*, volume 2611 of *Lecture Notes in Computer Science*, pp. 684–695, (2003).
- [18] N. Lipovetzky and H. Geffner, 'Inference and Decomposition in Planning Using Causal Consistent Chains', in *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, (2009).
- [19] N. Lipovetzky and H. Geffner, 'Searching for Plans with Carefully Designed Probes', in *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS'11)*, (2011).
- [20] James MacGlashan, 'Hierarchical Skill Learning for High-Level Planning', in *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI'10)*, (2010).
- [21] B. Marthi, S. Russell, and J. Wolfe, 'Angelic Hierarchical Planning: Optimal and Online Algorithms', in *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS'08)*, pp. 222–231, (2008).
- [22] M. Martin and H. Geffner, 'Learning Generalized Policies in Planning Using Concept Languages', in *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, pp. 667–677, (2000).
- [23] A. Menif, C. Guettier, and T. Cazenave, 'Planning and Execution Control Architecture for Infantry Serious Gaming', in *Proceedings of the 3rd International Planning in Games Workshop (PG'13)*, pp. 31–34, (2013).
- [24] C. Miller, R. Goldman, H. Funk, P. Wu, and B. Pate, 'A Playbook Approach to Variable Autonomy Control: Application for Control of Multiple, Heterogeneous Unmanned Air Vehicles', in *Annual Meeting of the American Helicopter Society*, (2004).
- [25] C. Muise, S. McIlraith, J. Baier, and M. Reimer, 'Exploiting N-Gram Analysis to Predict Operator Sequences', in *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, (2009).
- [26] H. Munoz-Avila, D. Aha, L. Breslow, and D. Nau, 'HICAP: An Interactive Case-Based Planning Architecture and its Application to Non-combatant Evacuation Operations', in *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99)*, pp. 870–875, (1999).
- [27] D. Nau, T. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F. Yaman, 'SHOP2: An HTN Planning System', *Journal of Artificial Intelligence Research*, **20**, 379–404, (2003).
- [28] M. Hakim Newton, J. Levine, M. Fox, and D. Long, 'Learning Macro-Actions for Arbitrary Planners and Domains', in *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, pp. 256–263, (2007).
- [29] I. Sánchez-Garzón, J. Fernández-Olivares, and L. Castillo, 'An Approach for Representing and Managing Medical Exceptions in Care Pathways Based on Temporal Hierarchical Planning Techniques', in *Process Support and Knowledge Representation in Health Care (Pro-Health'12)*, *Lecture Notes in Computer Science* 7738, pp. 168–182, (2013).
- [30] W. van der Sterren, 'Multi-Unit Planning with HTN and A*', in *AIGameDev Paris Game AI Conference*, (2009).
- [31] Nathaniel Waisbrot, Ugur Kuter, and Tolga Könik, 'Combining heuristic search with hierarchical task-network planning: A preliminary report.', in *FLAIRS Conference*, pp. 577–578, (2008).
- [32] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau, 'Automating DAML-S Web Services Composition Using SHOP2', in *Proceedings of the 2nd International Semantic Web Conference (ISWC'03)*, pp. 195–210, (2003).
- [33] S. Yoon, A. Fern, and R. Givan, 'Inductive Policy Selection for First-Order Markov Decision Processes', in *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI'02)*, pp. 568–576, (2002).
- [34] S. Yoon, A. Fern, and R. Givan, 'Learning Control Knowledge for Forward Search Planning', *Journal of Machine Learning Research*, **9**, 683–718, (2008).
- [35] H. Zhuo, D. Hu, C. Hogg, Q. Yang, and H. Munoz-Avila, 'Learning HTN Method Preconditions and Action Models from Partial Observations', in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pp. 1804–1809, (2009).

Learning the Structure of Dynamic Hybrid Relational Models

Davide Nitti^{1,2} and Irma Ravkic^{1,2} and Jesse Davis² and Luc De Raedt²

Abstract. Typical approaches to relational MDPs consider only discrete variables or else discretize the continuous variables prior to inference or learning. In contrast, we consider hybrid relational MDPs, which are represented as probabilistic programs and specify the probability density function of the continuous variables. Our key contribution is that we introduce a technique for learning their structure (and parameters) from data. The learned models contain rich relational descriptions as well as mathematical equations. We demonstrate the utility of our approach by learning a model that accurately predicts the effects of robot-arm actions. The learned model is then used for planning tasks.

1 Introduction

Markov Decision Processes (MDPs) are the standard representation used in probabilistic planning and reinforcement learning [19, 24, 22]. Relational MDPs integrate these processes with principles of statistical relational artificial intelligence [6], resulting in models that make abstraction of sets of states, transitions and can compactly represent and generalize across states having a variable number of objects through the use of relations. While relational MDPs are popular in planning and learning and there has been steady progress, their application to domains such as robotics is still severely limited by the emphasis on purely symbolic representations and the lack of general methods for dealing with subsymbolic (e.g., numeric) information. Approaches that have combined relational MDPs and robotics include learning probabilistic relational planning rules [17, 16], relational reinforcement learning [20, 13, 3], imitation learning [12], inverse reinforcement learning [11], and learning relational affordance models in multi-object manipulation tasks [10]. However, in all these cases the low-level numeric information had to be converted into a symbolic representation, often prior to learning and inference, leading to a loss of information, potentially affecting the quality of the obtained solutions. In another case, the approach required experts to provide partial code for the relational MDP [9].

We propose a different “hybrid” approach to relational MDPs in which numeric features are first-class citizens and the models are fully learned from data. Rather than discretizing numeric features and relations, we explicitly represent their probability density in an expressive probabilistic programming language, the dynamic distributional clauses (DDCs) [15], which has already been used in a robotics context and for which a planner, called HYPE [14], exists. Specifically, we address the problem of learning a state transition model from a set of trajectories collected from a robot-arm performing actions. Our central contribution is an algorithm to learn

the structure and the parameters of a DDC model representing a hybrid relational MDP. In order to realize this, we leverage statistical relational learning (SRL) techniques for learning hybrid probabilistic relational models (e.g., [23, 5, 18]). One novelty from an SRL perspective is that the learned DDCs include expressive features defined as mathematical equations involving the continuous variables, which is useful for finding, for instance, the object closest to the moved one. We demonstrate the utility of our DDC-TL algorithm (DDC Tree Learner) by applying the learned model to perform planning with HYPE [14] in a simple robotics scenario.

By making continuous features first class citizens in relational MDPs, we hope to contribute towards bridging the gap between symbolic and numeric approaches and to facilitate the application of relational MDPs to robotics.

2 Background

Next we introduce the necessary background needed to explain our learner of hybrid dynamic relational models and how the learned hybrid models are used for planning.

2.1 MDP

The problem of planning under uncertainty can be modeled as a Markov decision process (MDP). In an MDP, an agent interacts with its environment, described using a set of *states* S , a set of *actions* A that the agent can perform, a *transition function* $p : S \times A \times S \rightarrow [0, 1]$, and a *reward function* $R : S \times A \rightarrow \mathbb{R}$. That is, when in state s_t and performing action a_t , the probability of reaching s_{t+1} is given by $p(s_{t+1}|s_t, a_t)$, for which the agent receives the reward $R(s_t, a_t)$. It is assumed that the agent operates over a finite number of time steps $t = 0, 1, \dots, T$, with the goal of maximizing the expected reward: $\mathbb{E}[\sum_{t=0}^T \gamma^t R(s_t, a_t)]$, where s_0 is the start state, a_0 the first action, and $\gamma \in [0, 1]$ is a discount factor. In this paper we consider goal-oriented MDPs where there is a goal $g \subset S$ to reach, and the reward is high if we reach the goal, and low otherwise.

For planning, we shall use HYPE [14], a recently proposed planner for hybrid relational MDPs. It is a sample-based planner that exploits the model to simulate action effects and to determine the action in each state that maximizes the expected total reward. As input, HYPE requires both the starting state s_0 and the MDP specification of the domain described using DDCs. That is, it must be given the state transition model and the reward function.

2.2 Distributional Clauses

We assume some familiarity with standard terminology of statistical relational learning and logic programming [4]. Briefly, in logic

¹ These authors contributed equally to this work.

² Department of Computer Science, KU Leuven, Belgium

programming symbols can be terms and predicates (often called relations). A term is a constant, a logical variable (logvar) or an n -ary functor f applied to a tuple of terms t_1, t_2, \dots, t_n , that is, $f(t_1, t_2, \dots, t_n)$. A constant term refers to a single object in the domain of interest. A logical variable (logvar) X is a variable ranging over terms $x \in \mathcal{C}$, where \mathcal{C} is the set of possible ground terms (Herbrand universe). An atom is of the form $P(\tau_1, \dots, \tau_n)$ where P/n is a predicate with arity n and each τ_i is a term. For example $\text{near}(1, 2)$ is an atom that describes that objects 1 and 2 are close to each other. A ground atom (or expression) does not contain logvars. A literal is an atom or its negation.

In distributional clauses (DCs) [5, 15], an interpretation I assigns a value $I(\mathbf{a})$ to each random variable \mathbf{a} . While in logic programming that value of ground atoms will be true or false, in DCs the values can also be discrete or numeric. A substitution $\theta = \{v_1/\tau_1, \dots, v_n/\tau_n\}$ assigns terms τ_i to variables. A substitution θ can be applied to an expression E yielding the expression $E\theta$ where all variables v_i in E are simultaneously replaced by the corresponding terms τ_i in θ . A grounding substitution for an expression maps each logvar occurring in that expression to a term without logvars. The set of all grounding substitutions for an expression E is denoted $\text{grsub}(E)$.

Formally, a *distributional clause* (DC) is a formula of the form $\mathbf{h} \sim \mathcal{D} \leftarrow \mathbf{b}_1, \dots, \mathbf{b}_n$, where the \mathbf{b}_i are literals and \sim is a binary predicate written in infix notation. The name of the random variable \mathbf{h} and the distribution \mathcal{D} are formally terms. The intended meaning of a distributional clause is that each ground instance of the clause $(\mathbf{h} \sim \mathcal{D} \leftarrow \mathbf{b}_1, \dots, \mathbf{b}_n)\theta$ defines the random variable $\mathbf{h}\theta$ with distribution $\mathcal{D}\theta$ whenever all the $\mathbf{b}_i\theta$ are true, where θ is a substitution. A distributional clause is a template to define conditional probabilities: $p(\mathbf{h}\theta | (\mathbf{b}_1, \dots, \mathbf{b}_n)\theta) = \mathcal{D}\theta$. The term \mathcal{D} can be nonground, i.e., values, probabilities, or distribution parameters can be related to conditions in the body. Furthermore, given a random variable \mathbf{r} , the term $\simeq(\mathbf{r})$ constructed from the reserved functor $\simeq/1$ represents the value of \mathbf{r} .

Dynamic distributional clauses (DDCs) associate a time index to each random variable to capture temporal information. It is easy to specify an MDP using DDC as described in [14].

Example 1. Let us consider a scenario of pushing an object, where there is an object on the table and the robot has to move it in a given region. This scenario is modeled with the following MDP:

$$\text{pos}(\text{ID})_{t+1} \sim \text{gaussian}(\simeq(\text{pos}(\text{ID})_t) + (\text{DX}, \text{DY}), \Sigma) \leftarrow \text{push}(\text{ID}, (\text{DX}, \text{DY})). \quad (1)$$

$$\text{stop}_t \leftarrow \text{dist}(\simeq(\text{pos}(\text{ID})_t), (0.6, 1.0)) < 0.1. \quad (2)$$

$$\text{reward}(100)_t \leftarrow \text{stop}_t. \quad (3)$$

$$\text{reward}(-1)_t \leftarrow \text{not}(\text{stop}_t). \quad (4)$$

The DDC clause (1) defines the state transition model, i.e. the next position $\text{pos}(\text{ID})_{t+1}$ of an object ID after a push action with displacement (DX, DY) . The deterministic clause (2) defines when the goal is reached (e.g., an object is close to point $(0.6, 1.0)$) and the remaining clauses define the reward function.

Note that $\text{pos}(1)_{t+1}$ represents a random variable, i.e., the position of object 1 at time $t + 1$, with predicate-like notation. But unlike standard relational representations, a random variable can have a continuous (or categorical) range. Thus, in DCs and DDCs an interpretation assigns each random variable a value in its range.

Static inference in DCs is performed using importance sampling, while filtering in dynamic models is performed using particle filtering methods [15].

This paper explores how to learn a DDC program that describes the state-transition model by using DDC-TL, a relational tree learner inspired by learner of local models (LLM) [18] for hybrid relational dependency networks.

3 Learning State Transition Models

We propose an algorithm (DDC-TL) for learning a state-transition model, expressed as a DDC, from data described by continuous variables, discrete variables, and relations (e.g., `nextTo`). Concretely, given a set of discrete-time trajectories $(s_0, a_0, s_1, a_1, \dots, s_{T-1}, a_{T-1}, s_T)$, where s_t is the state (i.e., an interpretation that could describe object properties and relations such as position, orientation, type, and color) and a_t is an action at time t , the goal is to learn DDCs that define a state transition model of the following form:

$$Q_{t+1} \sim \mathcal{D}(f_c(s_t)) \leftarrow \text{body}_t \quad (5)$$

Each such clause defines the distribution $\mathcal{D}(f_c(s_t))$ of a relational random variable $Q_{t+1} \in s_{t+1}$ in terms of a set of (continuous) relational features $f_c(s_t)$ whenever body_t holds, where body_t is a conjunction of literals (discrete conditions) that refer to the current state s_t and action a_t . The relational features are essentially the random variables defined in the DDC. Note that DDCs defined in this manner result in a stratification where predicates at time $t+1$ only depend on predicates from the previous time step t . Stratification is required for DDCs to be well-defined [5].

In this paper, we consider a robot arm that moves objects on a table. The actions considered are grasping followed by a vertical or horizontal movement, pushing or tapping, while the features will be positions of objects and all the derived relationships. However, unlike related work, concepts like `rightOf`, `closeTo` or `aboveOf` are not manually defined, but are indirectly learned with equational features, which we will describe later. We assume inverse kinematics and a motion planner are available to execute the described actions.

3.1 The Learned Model

The key insight to develop an algorithm for learning DDCs is that we can leverage ideas from the learner of local models (LLM) approach for learning hybrid relational dependency networks (HRDNs) [18]. HRDNs are able to model dependencies in relational domains that have both continuous and discrete variables. Like DDCs, HRDNs use a variant of first-order logic as a template language for defining conditional probability distributions. An HRDN uses a set of local distributions to approximate a joint distribution over a set of random variables defined by grounding atoms constructed over a set of predicates \mathcal{P} and terms in \mathcal{C} . In this paper, we are interested in learning the dependencies that represent Formula (5). Hence, Q_{t+1} denote the set of predicates for which we learn dependencies. We use \mathcal{P}_t to denote the set of predicates describing the previous time step, which are used to construct the features that appear in $\text{Parents}(Q_{t+1})$. Each local distribution is quantified by a dependency $Q_{t+1} | \text{Parents}(Q_{t+1})$, where $Q_{t+1} \in Q_{t+1}$ is a predicate and $\text{Parents}(Q_{t+1})$ is a set of relational features that describe how Q_{t+1} depends on the other predicates in the domain. Such local distributions can be learned using relational regression trees [2], but unlike standard (relational) regression trees, each leaf does not contain a constant but instead has a linear or logistic regression model. The key observation is that the distribution modeled by a relational regression tree can be represented with DDCs. Therefore, techniques for learning relational regression

trees can be adapted for learning (certain classes of) distributional clauses.

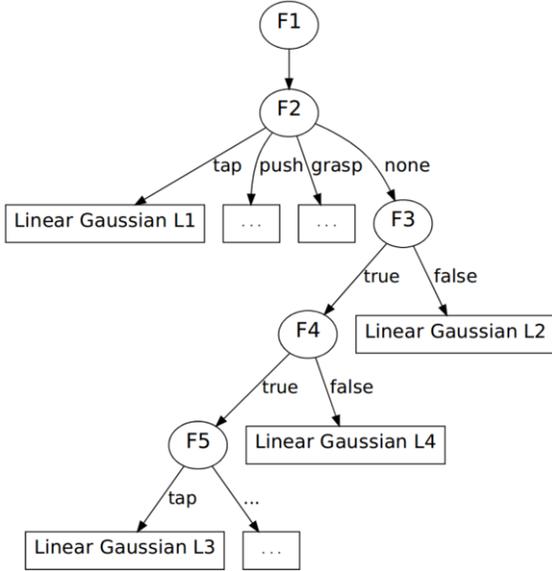


Figure 1. A simplified snapshot of a relational regression tree for predicting $\text{pos}(\text{ID})_{t+1}$. Ellipses represent internal nodes corresponding to learned relational features, and rectangles represent density functions for $\text{pos}(\text{ID})_{t+1}$ as defined in Example 2.

Example 2. Figure 1 shows a simplified example of a relational regression tree learned by DDC-TL. Each root-to-leaf path can be mapped onto one DDC, where each internal node defines a condition b_i that appears in the body of the rule or a numerical feature $f \in f_c$, and the leaf contains the probability distribution (density) $\mathcal{D}(f_c)$ that defines the random variable Q_{t+1} in terms of the features f along the path.

The example tree in Figure 1 corresponds to the following set of DDCs (some clauses are omitted):

$$\text{L1} : \text{pos}_x(\text{ID})_{t+1} \sim \text{gaussian}(P + \text{DX}, .02) \leftarrow \quad (6)$$

$$\text{F1} : P \text{ is } \simeq(\text{pos}_x(\text{ID})_t),$$

$$\text{F2-Tap} : \text{action}(\text{ID}, \text{tap}, \text{DX}, \text{DY})_t.$$

$$\text{L2} : \text{pos}_x(\text{ID})_{t+1} \sim \text{gaussian}(P, .004) \leftarrow \quad (7)$$

$$\text{F1} : P \text{ is } \simeq(\text{pos}_x(\text{ID})_t),$$

$$\text{F2-None} : \text{not}(\text{action}(\text{ID}, \text{Action}, \text{DX}, \text{DY})_t),$$

$$\text{F3-False} : \text{not}(\min\{\simeq(\text{pos}_y(\text{ID})_t) - \simeq(\text{pos}_y(\text{ID}_2)_t), \\ \text{ID} \neq \text{ID}_2\} < 0.07).$$

$$\text{L3} : \text{pos}_x(\text{ID})_{t+1} \sim \text{gaussian}(P + \text{DX}_2, .03) \leftarrow \quad (8)$$

$$\text{F1} : P \text{ is } \simeq(\text{pos}_x(\text{ID})_t),$$

$$\text{F2-None} : \text{not}(\text{action}(\text{ID}, \text{Action}, \text{DX}, \text{DY})_t),$$

$$\text{F3-True} : \min\{\simeq(\text{pos}_y(\text{ID})_t) - \simeq(\text{pos}_y(\text{ID}_2)_t), \\ \text{ID} \neq \text{ID}_2\} < 0.07,$$

$$\text{F4-True} : \min\{\simeq(\text{pos}_y(\text{ID})_t) - \simeq(\text{pos}_y(\text{ID}_2)_t), \\ \text{ID} \neq \text{ID}_2\} > -0.07,$$

$$\text{F5-Tap} : \text{action}(\text{ID}_2, \text{tap}, \text{DX}_2, \text{DY}_2)_t.$$

$$\text{L4} : \text{pos}_x(\text{ID})_{t+1} \sim \text{gaussian}(P, .004) \leftarrow \quad (9)$$

$$\text{F1} : P \text{ is } \simeq(\text{pos}_x(\text{ID})_t),$$

$$\text{F2-None} : \text{not}(\text{action}(\text{ID}, \text{Action}, \text{DX}, \text{DY})_t),$$

$$\text{F3-True} : \min\{\simeq(\text{pos}_y(\text{ID})_t) - \simeq(\text{pos}_y(\text{ID}_2)_t), \\ \text{ID} \neq \text{ID}_2\} < 0.07,$$

$$\text{F4-False} : \text{not}(\min\{\simeq(\text{pos}_y(\text{ID})_t) - \simeq(\text{pos}_y(\text{ID}_2)_t), \\ \text{ID} \neq \text{ID}_2\} > -0.07).$$

The aggregation function $\min\{U, C\}$, where U is a mathematical equation, returns the minimum of the U 's values obtained by applying all possible substitutions θ for the unbound logvars in U (i.e., logvars not in the target) such that the condition $C\theta$ holds.

In a nutshell, this program models that the next x position is the current position plus the action displacement of a tap action if one occurred (clause (6)). If there is no tap action on the object and if its y position is 7 cm higher (or lower) than that of any other object, then the object's x position will basically not change (clauses (7) and (9)). If the object is close to another object that is tapped, then its position will be affected as well (clause (8)).

One advantage of using relational regression trees to learn DDCs is that they satisfy the mutual exclusiveness property required by DCs, which states that if there are two distributional clauses defining the same continuous random variable, their bodies must be mutually exclusive. This is guaranteed by relational regression trees.

We shall now first introduce the type of features and distributions used by our learning algorithm, and then provide a description of the relational regression tree learning algorithm.

3.2 Relational Features

Each learned DDC will have the form shown in Formula (5) and will correspond to a root-to-leaf path in a relational regression tree. There are two types of internal nodes in such a tree:

- logical conditions which are tests that evaluate to true or false as in standard decision trees. These nodes contribute a condition to body_t ; and
- continuous features which specify a parameter that appears in the conditional distribution $\mathcal{D}(f_c(s_t))$ contained in all leaf nodes below this node.

In contrast to standard decision and regression trees, nodes containing a continuous feature do not specify a test. Hence, they do not split the data and only have one child in the tree.

We now define the distributions and features used in our DDC-TL algorithm.

Distributions. Leaf nodes, and hence $\mathcal{D}(f_c(s_t))$ in Formula (5), contains one of the following distributions:

- A **linear Gaussian distribution** defining $\mathcal{D}(f_c(s_t)) = \mathcal{N}(\mu, \sigma)$ where $\mu = \alpha + \sum_{f \in f_c(s_t)} f \cdot \beta_f$ if Q_{t+1} 's range is continuous
- A **softmax** or normalized exponential model defining $\mathcal{D}(f_c(s_t))$ as a $\text{softmax}(\alpha^{(j)} + \sum_{f \in f_c(s_t)} f \cdot \beta_f^{(j)})$ if Q_{t+1} ranges over discrete values j , where the $\beta_f^{(j)}$ are the weights for the features $f \in f_c(s_t)$ and value j in Q_{t+1} 's range.

The set $f_c(s_t)$ contains the numeric (continuous) features encountered on the root-to-leaf path. Notice that these distributions degenerate into a $\mathcal{N}(\alpha, \sigma)$ and a probability mass function when there are no numeric features, that is, when $f_c(s_t) = \emptyset$.

Features. Given the (target) random variable $Q_{t+1}(A_0, \dots, A_n)$ with logical variables $A = \{A_0, \dots, A_n\}$, any term U representing a random variable with logical variables B_i such that $\forall B_i \in A$, can be used as a feature in a distributional clause. For example, given the target random variable $\text{pos}_x(\text{ID})_{t+1}$, any random variable with logical variable ID is an admissible feature in f_c (if continuous) or in body_t (if discrete), e.g., $\text{pos}_x(\text{ID})_t$, $\text{pos}_y(\text{ID})_t$, $\text{color}(\text{ID})$. Continuous features can be discretized and added as conditions in body_t , e.g., $\text{pos}_x(\text{ID})_t > 2$.

If the random variable U has some logical variable B_i that does not appear in A , then we allow aggregations of the form:

$$\text{agg}(U, C)$$

where agg is an aggregation function, U is a term representing a random variable, and C is a conjunction of atoms that evaluates to true or false (i.e., a condition). The feature computes the specified aggregate agg over values returned as the result of aggregation over all possible groundings of $U\theta$ that satisfy $C\theta$. Sometimes the aggregation might not be defined. For example, if there are no objects satisfying a specific condition C , then aggregation is over an empty set. In case the node represents discretization of an aggregation feature applied to an empty set, the discretized feature is assumed to be *false* as, for example, F3-False in Example 2. If at some node a continuous feature evaluates to *undefined* during learning, that feature is discarded from the set of candidate features for that node and all its children.

Constructing high-level concepts from low-level numeric sensor data often requires performing mathematical operations (e.g., addition, multiplication, etc.) on the raw data. As our goal is to enable automated discovery of these types of concepts, we allow features involving two atoms U_1 and U_2 that both have numeric ranges. These features have the following form:

$$\text{agg}\{(U_1 \text{ op } U_2), C\}$$

where op is a mathematical operator ($+$, $-$, $*$, $/$), and agg and C are defined as before.

Note that an admissible aggregation over U needs at least one logical variable B_i bound with a logical variable in the target $Q_{t+1}(A_0, \dots, A_n)$, i.e. $\exists i, j : B_i = A_j$.

Example 3. An example of feature that includes an equation is:

$$\min\{(\text{pos}_y(\text{ID})_t - \text{pos}_y(\text{ID}_2)_t), \text{ID} \neq \text{ID}_2\}$$

For a specific grounding of ID bound with the target, this feature calculates the minimum distance in the y -plane between that object and other objects around it. By placing a threshold on this feature, we

could distinguish when two objects are close. Thus, these features can represent important concepts that are not explicitly encoded in the raw data.

The condition C can also be a conjunction, for example:

$$\min\{(\text{pos}_y(\text{ID})_t - \text{pos}_y(\text{ID}_2)_t), \\ (\text{ID} \neq \text{ID}_2, \text{action}(\text{ID}_2, \text{push}, \text{DX}, \text{DY})_t)\},$$

which calculates the minimum distance between two distinctive objects if the pushing action is performed on the object ID_2 . If the condition is not satisfied, the feature's value is *false*.

Logical Conditions. Features can easily be turned into logical conditions. If a feature f is discrete, the condition $f = v$ (with a value v in the range of f) can serve as an atom in the body of a distributional clause. When a discrete feature is included in a node, there will be one subtree for each possible value v of that node in the decision tree, as usual. If f is continuous, we can discretize it by picking some threshold v in the range of f and building a feature such as $f > v$ or $f \leq v$.

3.3 Learning Relational Regression Trees

Algorithm 1 DDC-TL

```

1: function FEATURESPACE( $Q_{t+1}, data, \mathcal{P}, op, aggrs$ )
2:    $\mathcal{F}_{Q_{t+1}} \leftarrow \text{CONSTRUCT}(Q_{t+1}, data, \mathcal{P}, op, aggrs)$ 
3: end function
4: function GROWTREE( $tree, Q_{t+1}, data, score, \mathcal{F}_{Q_{t+1}}$ )
5:    $f, score_f \leftarrow \text{FINDBESTFEATURE}(Q_{t+1}, data, \mathcal{F}_{Q_{t+1}})$ 
6:   if stopcond( $score_f$ ) then
7:     ADDLEAF( $tree$ )
8:   end if
9:   if is_continuous( $f$ ) then ▷ Continuous range
10:     $tree \leftarrow \text{ADDNODE}(f, tree)$ 
11:    GROWTREE( $tree, Q_{t+1}, data, score, \mathcal{F}_{Q_{t+1}} \setminus f$ )
12:  else
13:     $tree \leftarrow \text{ADDNODE}(f, tree)$  ▷ Discrete range
14:    for each  $a$  in domain( $f$ ) do
15:       $filtered \leftarrow \text{filter}(data, f, a)$  ▷ data such that  $f = a$ 
16:      GROWTREE( $tree[a], Q_{t+1}, filtered, score, \mathcal{F}_{Q_{t+1}} \setminus f$ )
17:    end for
18:  end if
19: end function

```

Algorithm. For each predicate Q_{t+1} occurring in a state, we learn a relational regression tree that defines the distribution according to a set of clauses of the form defined in Formula (5). Algorithm 1 outlines a top-down procedure for learning the tree. First, the function FEATURESPACE constructs a set of candidate relational features $\mathcal{F}_{Q_{t+1}}$ that can appear in the internal nodes when learning the tree for a target predicate Q_{t+1} (legal features will be described in detail later). Starting from the empty tree, it adds internal nodes as follows. It calls FINDBESTFEATURE which iterates through the set of candidate features and tries using each feature as the current internal node. It calculates the difference in score between the new tree and the old tree using five fold internal cross validation on the training data (the score function is discussed in detail later). If no feature improves the score (that is, stopcond($score_f$) is true), then a leaf node is added. Otherwise, the algorithm greedily selects the highest scoring feature

f to include as the internal node. The selected feature is removed from the set of candidate features. If the selected feature has a continuous range and never evaluates to undefined in the current branch, the procedure recurses and passes all data to the next node. If the selected feature has a discrete range, one branch is constructed for each value of the discrete feature (yielding a logical condition). The data is divided over the branches according to the feature’s value, and the procedure recurses along each branch. When no feature improves the score, the recursion stops, a leaf node is added, and the parameters of the leaf’s probability distribution or density function are estimated.

Defining Legal Features. We now describe in more detail the CONSTRUCT function in Algorithm 1, which receives as input a target predicate Q_{t+1} , a set of aggregation functions (*aggrs*), mathematical operators *op*, and *data* from which the ranges of predicates are extracted. The features of the form $\text{agg}(U, C)$ are constructed by exhaustively enumerating all combinations of $\text{agg} \in \text{aggrs}$, U , and C that meet the following constraints. The atom U does not appear in C , and C is a conjunction with fewer than $k \leq N$ conjuncts. Each conjunct in C is either a randvar-value test or an (in)equality constraint between two logvars. Each condition in C is “linked” to U via a path of shared logvars, that may pass also through other conditions. As aggregation functions, we consider \min , \max , avg , and \simeq . Recall, that $\simeq(U)$ simply returns the value of U in the data and that it is only applicable if there is exactly one grounding substitution of U for each grounding substitution θ of C for which $C\theta$ is true in the data.

We construct candidate features of the form $\text{agg}\{(U_1 \text{ op } U_2), C\}$ in an analogous manner. Both U_1 and U_2 must have continuous ranges. We consider $\text{op} \in \{+, -, *, /\}$ and the aforementioned aggregation functions.

For all features that return a real value, we construct two variants: 1) the feature itself to be used as a parameter of the distribution, and 2) discretized version that can be used as a logical condition in the body of a rule. The second alternative is implemented by discretizing the feature into a number of different bins where bin widths are set based on the training data. In the experiments we used five bins.

Example 4. To illustrate how to threshold a continuous value to create a discrete feature, consider again the following feature

$$\min\{(\text{pos}_y(\text{ID})_t - \text{pos}_y(\text{ID}_2)_t), \text{ID} \neq \text{ID}_2\}$$

which calculates the minimum distance in the y -plane between two distinct objects. We also consider features with the following template:

$$\min\{(\text{pos}_y(\text{ID})_t - \text{pos}_y(\text{ID}_2)_t), \text{ID} \neq \text{ID}_2\} \bowtie \text{Thresh}$$

where $\bowtie \in \{<, >\}$, and Thresh is a threshold determined from the training data. If \bowtie is $<$ and $\text{Thresh} = 0.07$, we get Feature F3 from Example 2.

Score Function. In Algorithm 1 we use the loglikelihood as the scoring function. The data consists of a set of traces of the form $(s_0, a_0, s_1, a_1, \dots, s_{T-1}, a_{T-1}, s_T)$, and since we are interested in learning the state transition model $p(s_{t+1}|s_t, a_t)$ with the Markov assumption, we split the traces into triples of the form (s_t, a_t, s_{t+1}) .

Each triple can be viewed as an interpretation e that assigns values to each of the logical atoms and random variables that appear in the triplet at times $t + 1$ and t . The loglikelihood of a triple

$e = (s_t, a_t, s_{t+1})$ for a DDC program \mathbb{P} is

$$\begin{aligned} \text{score}(\mathbb{P}, e) &= \log p_{\mathbb{P}}(s_{t+1}|s_t, a_t) = \\ &= \sum_{(h_{t+1} \sim \mathcal{D} \leftarrow \text{body}_t) \in \mathbb{P}} \sum_{\theta: \text{body}_t \theta \cup h_{t+1} \theta \subseteq e} \log p_{\mathcal{D}}(e(h_{t+1} \theta) | e(\text{body}_t \theta)), \end{aligned}$$

where $e(a)$ denotes the value assignment of random variable a in interpretation e . In this definition we assume that the transition probability factorizes as follows: $p(s_{t+1}|s_t, a_t) = \prod_{q_i \in s_{t+1}} p(q_i|s_t, a_t)$, that is, every variable in the next state $q_i \in s_{t+1}$ depends only on the previous state and action.

Scoring a DDC program first requires estimating the parameters for the CPDs $\mathcal{D}(f_c(s_t))$. For linear Gaussian distributions, parameter learning requires estimating the weight vector for the linear regression model. This can be done via standard maximum likelihood techniques (e.g., ridge regression [1]). Similarly, softmax parameter estimation requires learning the weight vectors for the logistic regression model. We follow standard gradient ascent approach to maximize the loglikelihood [1].

3.4 Related Work

In the literature there are several relational approaches that try to learn a model and use it for planning. However, most approaches only support relational (binary) random variables. For example, Pasula et al. [17, 16] learn noisy indeterministic deictic (NID) rules that define the state in terms of facts (true or false statements). Such representation has been used with the planner PRADA [8]. Similarly, Moldovan et al. [10] learn relational affordance models in multi-object manipulation tasks. In such works, the training data is discretized and the model is relational, without the possibility to include continuous variables. This makes the model abstract but useful low level information is lost. Moreover, in robotics applications the discretization is generally handcrafted. In contrast, our approach tries to learn the best features for the prediction task.

Other approaches, based on RL, try to directly learn the Q-function or the policy without learning the state transition model. Among the works with a relational representation there is an approach for performing policy gradient boosting [7], and approach for imitation learning [12]. Both methods require a regression tree learner, thus our approach can be easily used in such settings.

Few works consider learning hybrid relational domains. One simplified attempt is the one of Moldovan et al. [9], that learns the structure of a Bayesian network to model two object effects and convert it into DDC clauses. However, this fixed conversion might not generalize well on more than two objects. In contrast, our approach directly learns a hybrid relational model, which allows to combine data collected with a different number of objects, and thus provides a better generalization. Moreover, their work considers only simple conditional linear Gaussian models without aggregation or equational features and they evaluate only plans of horizon one.

This work is based on the learner of local models (LLM), an approach for learning the structure of HRDNs [18]. Our approach differs from the LLM algorithm used for HRDNs in several crucial ways. Most importantly, we provide support for automatically learning relationships that involve equational features. This is a key capability in terms of being able to model spatial relationships based on low-level continuous data. Additional differences include that we take into account time (and dynamics) and use a tree-based representation for the distributions/densities. Furthermore, the trees are represented using a set of DDCs, enabling their direct use for planning with HYPE.

Beyond incorporating mathematical equations, our tree representation also differs from existing relational regression trees such as TILDE [2] and its extension towards learning aggregate functions by [21]. Another difference is that we use distributions in the leaves. Finally, there may be features on a path that are not used to “split” the data but will be used as features in the distributions.

4 Experiments

This section empirically evaluates our approach for learning dynamic hybrid transition models. Specifically we want to answer the following questions:

- Q1) How accurate are our learned state transition models for prediction tasks?
- Q2) How does DDC-TL compare to the propositional hybrid learners?
- Q3) What is the performance of the planner that executes our learned models?

To evaluate these questions, we consider a robotics scenario that consists of a table with cubes and a Kinova MICO robot arm that can manipulate the cubes as shown in Figure 2. The learner will only have access to the x, y position of each object. Hence, the learned model must automatically discover important intermediate concepts like `closeTo` by constructing features that build upon the low-level positional data. The learned model is then used for planning.

The learning was performed on Intel(R) Xeon(R) CPU 2.40GHz machines with 128 Gb memory. The planning was performed on a laptop Intel(R) i7 CPU 2.40GHz with 4 Gb memory.

4.1 Experimental Setup and Data Generation

To generate data, a sequence of actions is randomly generated and executed on the objects and their effects (positions) are stored, that is, (s_t, a_t, s_{t+1}) triples. The state is the x, y position of each object. The number of objects used in the experiments are two, three and four. The actions considered are grasping followed by moving horizontally, grasping followed by moving vertically, pushing and tapping. Each action refers to an object and is parameterized by the displacement, a continuous value. The action might fail, e.g., when the object is out of the range of the robot, or the object is too close to other objects. In such cases, the objects will not move or move differently than expected from a successful action execution. The concept of failure is not explicitly encoded and the learner must learn how to distinguish the different effects according to the state and the action.

4.2 Methodology

We perform the experiments with the following learners:

- **Basic Propositionalization.** We propositionalize the state by constructing one feature for each fact. We denote these with B .
- **Advanced Propositionalization.** We propositionalize the state by using exactly the candidate features considered by DDC-TL. We denote these with A .
- **DDC-TL.** Our approach for learning hybrid dynamic state transition models as introduced in Section 3.
- **DDC-TL-50.** A version of DDC-TL using feature selection when learning the tree. This means that when deciding on the split the learner uses only the top 50 selected features.

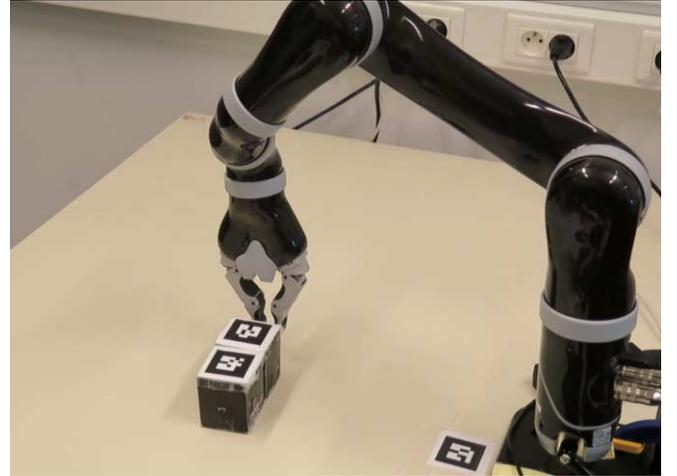


Figure 2. MICO arm performing a left tap action of the object on the right. This action moves the other object as well.

In Basic Propositionalization, each interpretation with n objects is converted in n training examples, where the target is the next position of an object $\text{pos}(x)_{t+1}$, and the features are the positions of all objects at time t . In this scheme, we need to ensure that for each training example with target $\text{pos}(x)_{t+1}$, the features corresponding to object x are in the same position.

Example 5. Given the triplet interpretation $\{\text{pos}(1)_{t+1}, \text{pos}(2)_{t+1}, \text{pos}(3)_{t+1}, \text{pos}(1)_t, \text{pos}(2)_t, \text{pos}(3)_t\}$ (the values and the actions are omitted for compactness), the Basic Propositionalization extracts three training examples:

$$\begin{aligned} \text{pos}(1)_{t+1} &| \text{pos}(1)_t, \text{pos}(2)_t, \text{pos}(3)_t \\ \text{pos}(2)_{t+1} &| \text{pos}(2)_t, \text{pos}(1)_t, \text{pos}(3)_t \\ \text{pos}(3)_{t+1} &| \text{pos}(3)_t, \text{pos}(2)_t, \text{pos}(1)_t \end{aligned}$$

Note that the feature $\text{pos}(x)_t$ of the target object x is always the first feature selected. The non-target object features do not have a fixed order.

In DDC-TL we use all the available features for learning. In contrast, DDC-TL-50 uses feature selection in each node to overcome some of the known issues of greedy search and control against overfitting. To select a subset of the features, we use Lasso regression, ARD linear regression, and Random Forests for regression. We pick the top 50 features according to the absolute value of the weights in the linear models, and the feature importance metric of Random Forests.

We use the following propositional learners for our experiments: Lasso, regression trees, and gradient boosted regression trees. We consider three different setups: the data only contains two objects, the data only contains three objects, and the data containing two or three objects. We also evaluate the models learned on two and three objects by applying them to test data that contain four objects. In each case, the goal of the model is to predict the x and y positions of each object in the next time step.

4.3 Evaluation Settings

We consider two different evaluation setups. First, we perform 10-fold-cross-validation, and report the average root-mean-squared-error (RMSE) over all held-out folds and learning time. For the

propositional learners, hyper-parameter optimization is performed using a grid search with internal cross-validated on the training set.

Second, we use the DDC-TL models with the best performance on 10-fold-cross-validation to perform planning using HYPE. We use a reward of 100 if the goal is achieved and -1 otherwise, and consider the following goals:

Region This requires moving any object in a specific region (distance around 20 *cm* from the closest object).

Swap This entails moving an object to the right (or left) side of another object.

The presence of additional objects can make actions fail. Moreover, if an object is out of reach it can be moved only indirectly by acting first on the other objects that can influence its position. For this reason, it is important that the action effects and the interactions between objects are captured in the learned model. For each goal, a set of 15 experiments is performed from different starting positions and the average number of steps is provided.

We assume that every action is applicable (i.e., executable), even when the action does not provide an effect. Obviously, the planner will need to select the actions that are useful to reach the goal. Actions that do not produce movement (i.e., fail) will probably not be selected.

Videos of actions executed by the robot are available at <https://dtai.cs.kuleuven.be/ml/systems/DC/>

4.4 Results and Discussion

Table 1 presents the RMSE of the x and y positions for all approaches, and Table 2 summarizes the learning time. Note that the basic propositionalization approach is not applicable when combining the two and three objects data as this approach only works for a fixed number of input variables and hence a fixed number of objects. DDC-TL learns a model that has an error smaller than most of propositional models tested (Q1). In the two objects case, DDC-TL-50 beats all the propositional models (Q2). On the three objects setting, DDC-TL has the best result. On the dataset combining the data of both two and three objects, DDC-TL and Gradient Boosting have the same performance. Note that Gradient Boosting has access to the same features as DDC-TL and is an ensemble approach which provides it with an advantage.

Moreover, for the propositional models learned with the advanced feature set and DDC-TL, we used the models learned on data about two and three objects data and evaluated them on test data containing a number of objects (four) that was never seen in the training data. The result for this setting is shown in the last column of Table 1. The superior performance in this setting shows an important advantage of our approach, which is that it is able to generalize to new scenarios involving a different number of objects.

Table 2 shows the learning time. We report the time needed to calculate the values of all the features in the feature space and the time to actually learn the model. The calculated features are also used when learning propositional approaches. The proposed DDC-TL is more complex, thus generally slower than propositional learning methods. However, our approach has not been optimized and there are number of ways to improve performance (e.g., use feature selection, use a beam search, etc.).

The models have been also qualitatively tested on some actions. For example, Figure 3 shows the predicted positions of two objects (based on sampling) after a tap action. The visual inspection confirms

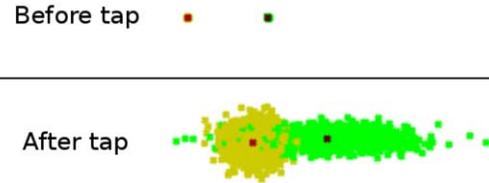


Figure 3. Prediction of tap action with two objects. The object on the left is tapped on the right, this moves the other object. 1000 samples of object positions are shown with the mean in dark color.

that the model captures the interactions between objects. In particular, the model contains relevant features such as the closeness of an object to the object being moved (as shown in Figure 1) and the concept ‘not reachable’, that is, the object is far away from the arm.

Table 3 presents results about using the learned model with the HYPE planner. For the experiments we use the model learned by DDC-TL with the full feature set. For the **Region** task it has an average success rate of 61% and needs 3.4 steps on average (when it succeeds); whereas for the **Swap** task, it has an average success rate of 62% and needs 3.8 steps on average (Q3). These results confirm that DDC-TL is able to learn a model that is sufficiently accurate for performing simple planning tasks. However, the plans do not always succeed for two reasons. First, there are some scenarios where the objects interactions are not properly modeled. This is expected given that we have a limited amount of training data (196 interpretations for the setup with two objects, and 376 interpretations for the setup with three objects). Second, to keep the planning time reasonable, we limited the number of samples used by the planner to 250. Using more samples could improve the planner’s performance.

Learners	RMSE			
	Num. of objects			
	2	3	2 + 3	2 + 3 \rightarrow 4
DDC-TL	0.029	0.022	0.024	0.023
DDC-TL-50	0.027	0.026	0.026	0.019
Lasso-B	0.030	0.026	NA	NA
Regression Tree-B	0.037	0.030	NA	NA
Gradient Boosting-B	0.029	0.025	NA	NA
Lasso-A	0.031	0.026	0.027	0.023
Regression Tree-A	0.037	0.029	0.032	0.030
Gradient Boosting-A	0.029	0.024	0.024	0.023

Table 1. The RMSEs of predicting the next x and y positions, based on the learned models, averaged over 10 folds.

5 Conclusions

To the best of our knowledge, this paper is the first approach that can learn a dynamic statistical relational state transition model in a hybrid domain and then apply an MDP planner to the learned model. The central contributions of the paper are: adapting HRDN learning

Learners	Runtime (seconds)		
	Num. of objects		
	2	3	2 + 3
Feature computation	1072.5	2099.5	3186
DDC-TL	3501.5	58322.0	100159.0
DDC-TL-50	1436.0	14846.5	25646.0
Lasso-B	1.8	2.7	NA
Regression Tree-B	0.2	0.7	NA
Gradient Boosting-B	19.9	75.2	NA
Lasso-A	33.7	98.1	160.6
Regression Tree-A	7.2	19.6	26.6
Gradient Boosting-A	636.5	1824.5	2490.5

Table 2. The learning time measured in seconds for learning the models for predicting the next x and y positions averaged over 10 folds. Feature computation is the time DDC-TL needs to calculate the values of all the features in the feature space. These calculated feature values are also used for learning propositional models.

	Avg. # steps (max 5)	Avg. Reward	Success Rate
Region	3.4	56.8	61%
Swap	3.8	57.3	62%

Table 3. Planning results using planner HYPE with the model learned by DDC-TL. ‘Avg. # steps’ refers to the average number of steps when the plan succeeds.

for learning dynamic distributional clauses, extending the rich relational feature space to include mathematical equations involving the continuous variables, planning with the learned hybrid models, and identifying a potential robotics application for SRL. Empirically, we demonstrated the merits of our approach using scenario involving a real robotic arm. One of the future directions is to perform learning with partial observability or an unknown number of objects. Another future direction is to explore whether the features that we select in one scenario and that are deemed as interesting can be re-used in future scenarios. This might speed up the learning and increase the expressivity and compression of the models as the number of scenarios grows.

Acknowledgements

This work has been supported by the Research Foundation Flanders, by the Chist-Era ReGround project, and by the BOF funds of the KULeuven.

References

- [1] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
- [2] Hendrik Blockeel and Luc De Raedt, ‘Top-down induction of first-order logical decision trees’, *Artificial intelligence*, **101**(1), 285–297, (1998).
- [3] Tom Croonenborghs, Jan Ramon, Hendrik Blockeel, and Maurice Bruynooghe, ‘Online learning and exploiting relational models in reinforcement learning’, in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 726–731, (2007).
- [4] *Probabilistic Inductive Logic Programming — Theory and Applications*, eds., L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, volume 4911 of *Lecture Notes in Artificial Intelligence*, Springer, 2008.
- [5] B. Gutmann, I. Thon, A. Kimmig, M. Bruynooghe, and L. De Raedt, ‘The magic of logical inference in probabilistic programming’, *Theory and Practice of Logic Programming*, **11**(4-5), 663–680, (2011).
- [6] Saket Joshi, Roni Khardon, Prasad Tadepalli, Alan Fern, and Aswin Raghavan, ‘Relational Markov decision processes: Promise and prospects’, in *AAAI Workshop: Statistical Relational Artificial Intelligence*, (2013).
- [7] Kristian Kersting and Kurt Driessens, ‘Non-parametric policy gradients: A unified treatment of propositional and relational domains’, in *Proceedings of the 25th International Conference on Machine Learning*, pp. 456–463, (2008).
- [8] Tobias Lang and Marc Toussaint, ‘Planning with Noisy Probabilistic Relational Rules’, *Journal of Artificial Intelligence Research*, **39**, 1–49, (2010).
- [9] Bogdan Moldovan and Luc De Raedt, ‘Learning relational affordance models for two-arm robots’, in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 2916–2922. IEEE, (2014).
- [10] Bogdan Moldovan, Plinio Moreno, Martijn van Otterlo, José Santos-Victor, and Luc De Raedt, ‘Learning relational affordance models for robots in multi-object manipulation tasks’, in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4373–4378. IEEE, (2012).
- [11] Thibaut Munzer, Bilal Piot, Matthieu Geist, Olivier Pietquin, and Manuel Lopes, ‘Inverse reinforcement learning in relational domains’, in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, (2015).
- [12] Sriraam Natarajan, Saket Joshi, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik, ‘Imitation learning in relational domains: A functional-gradient boosting approach’, in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, volume 22, p. 1414, (2011).
- [13] Vien Ngo and Marc Toussaint, ‘Model-based relational reinforcement learning when object existence is partially observable’, in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 559–567, (2014).
- [14] Davide Nitti, Vaishak Belle, and Luc De Raedt, in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD) 2015, Part II*, volume 9285.
- [15] Davide Nitti, Tinne De Laet, and Luc De Raedt, ‘A particle filter for hybrid relational domains’, in *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2764–2771, (2013).
- [16] Hanna Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling, ‘Learning probabilistic relational planning rules’, in *International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 73–82, (2004).
- [17] Hanna M Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling, ‘Learning symbolic models of stochastic domains’, *Journal of Artificial Intelligence Research*, 309–352, (2007).
- [18] Irma Ravkic, Jan Ramon, and Jesse Davis, ‘Learning relational dependency networks in hybrid domains’, *Machine Learning*, **100**(2), 217–254, (2015).
- [19] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.
- [20] Prasad Tadepalli, Robert Givan, and Kurt Driessens, ‘Relational reinforcement learning: An overview’, in *Proceedings of the ICML-2004 Workshop on Relational Reinforcement Learning*, pp. 1–9, (2004).
- [21] Anneleen Van Assche, Celine Vens, Hendrik Blockeel, and Sašo Džeroski, ‘First order random forests: Learning relational classifiers with complex aggregates’, *Machine Learning*, **64**(1-3), 149–182, (2006).
- [22] Luis Gustavo Rocha Vianna, Leliane N. de Barros, and Scott Sanner, ‘Real-time symbolic dynamic programming’, in *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*, pp. 3402–3408, (2015).
- [23] Jue Wang and Pedro Domingos, ‘Hybrid Markov Logic Networks’, in *Proceedings of the 23rd Conference on Artificial intelligence (AAAI)*, volume 2, pp. 1106–1111, (2008).
- [24] M. Wiering and M. van Otterlo, *Reinforcement Learning: State-of-the-Art*, Adaptation, Learning, and Optimization, Springer, 2012.

A Distributed Asynchronous Solver for Nash Equilibria in Hypergraphical Games ¹

Mohamed Wahbi and Kenneth N. Brown ²

Abstract. Hypergraphical games provides a compact model of a network of self-interested agents, each involved in simultaneous sub-games with its neighbors. The overall aim is for the agents in the network to reach a Nash Equilibrium, in which no agent has an incentive to change their response, but without revealing all their private information. Asymmetric Distributed constraint satisfaction (ADisCSP) has been proposed as a solution to this search problem. In this paper, we propose a new model of hypergraphical games as an ADisCSP based on a new global constraint, and a new asynchronous algorithm for solving ADisCSP that is able to find a Nash Equilibrium. We show empirically that we significantly reduce both message passing and computation time, achieving an order of magnitude improvement in messaging and in non-concurrent computation time on dense problems compared to state-of-the art algorithms.

1 Introduction

In many multi-agent problems, agents must interact with each other to achieve a global goal while maximising their own individual preferences. The *hypergraphical games* model [13] provides a compact representation of the problem, in which agent interactions are represented as normal-form strategic subgames, and the relationship topology between the agents is represented as a hypergraph. Each agent has a set of strategies, and a utility function specifying the agent's payoff under each possible combination of its own and neighboring agents' strategies in each subgame. A solution to a hypergraphical game is the selection of a strategy for each agent, such that the network is in equilibrium. Typically, the aim is to find a Nash Equilibrium (NE), in which no agent can improve its payoff by changing its strategy. To model more realistic problems, ϵ -approximate Nash Equilibria (ϵ -NE) are considered, in which no agent can improve its payoff by more than some minimum threshold ϵ .

Given the multi-agent setting, algorithms to compute solutions should be distributed, to avoid the need for agents to reveal potentially private information. Early work focused on identifying graph topologies which allowed polynomial-time solutions, based on algorithms for Bayesian Network inference. More recent approaches focus on arbitrary graphs with cyclic dependencies, and represent the problem as *asymmetric distributed constraint satisfaction* (ADisCSP), maintaining the individual utility functions as extensional table constraints [5]. ADisCSP allows agents to keep

their strategic information private while optimizing their local utility, and allows the system to stabilize at an equilibrium by coordinating agents' decisions. However, for dense graphical games with large strategy sets, the encoding of the table constraints ([5]) becomes expensive in the number or size of messages that need to be exchanged.

Our contributions in this paper are as follows. We develop new approaches to finding approximate Nash Equilibria for hypergraphical games using the distributed constraint satisfaction framework. We develop a new model of a hypergraphical game as ADisCSP using a new global constraint, ϵ -BRConstraint, to represent an agent's requirement to find an approximate best strategy given the other decisions in its neighborhood. We introduce asymmetric asynchronous backtracking, AABT, a new algorithm for solving ADisCSP with global constraints using intelligent backtracking to avoid thrashing. AABT is then used to solve the problem of finding an ϵ -NE in hypergraphical games formulated as an ADisCSP. We compare the new model and algorithm empirically to previous state-of-the-art algorithms, and we show that we achieve significant reductions in non-concurrent computation time and an order of magnitude improvement in message passing.

The paper is organized as follows. Section 2 gives a brief overview of related works on game theory and distributed CSP. Section 3 introduces the necessary background, basic notation and terminology. We present our model of the problem of finding ϵ -NE in hypergraphical games as ADisCSP in Section 4. Section 5 introduces our new algorithm, AABT, for solving ADisCSP with global constraints, and we show our empirical results in Section 6.

2 Related Work

[8] introduced *graphical games*, a compact representation of n -player normal-form games and proposed NashTree, a dynamic programming algorithm for computing Nash equilibria in graphical games for which the underlying graph is a tree. NashTree consists of two phases: a table passing phase from the leaves to the root and an assignment passing phase from the root to the leaves. [15] proposed a constraint satisfaction generalization of NashTree for general graphical games using variable elimination. They transform the graphical game into a tree via triangulation, and then subsequently run NashTree algorithm on the resulting junction tree.

[12] introduced the NashProp algorithm, another generalization of NashTree for general graphical games based on belief propagation requiring no triangulation. In the table passing phase, NashProp proceeds in a series of rounds to maintain (generalized) arc consistency in the constraints network. In each round, every node will send a different binary-valued table to each of its neighbors in the graph. Each table represents the value combination of the sender and the receiver

¹ This work is funded by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

² Insight Centre for Data Analytics, School of Computer Science and IT, University College Cork, Ireland
email: {mohamed.wahbi,ken.brown}@insight-centre.org

that the sender believes could be in an ϵ -NE. In the assignment passing phase, NashProp performs a synchronous search to find ϵ -NE [5, Section 5.1].

The pioneering algorithm for symmetric DisCSP was *asynchronous backtracking* (ABT) [18, 2]. ABT is an asynchronous algorithm executed autonomously by each agent, and is guaranteed to converge to a global consistent solution (or detect inconsistency) in finite time. [3] proposed two varieties of ABT for solving ADisCSP, namely ABT-2ph and ABT-1ph. ABT-2ph alternates the execution of ABT considering constraints in one direction following a total ordering on agents until the problem is solved or inconsistency is proved. ABT-1ph checks constraints asynchronously in both directions, by agents sending their proposed assignments to all their neighbors. In ABT-1ph, an agent only changes its assignment when it is inconsistent with a higher neighbor assignment. When it is inconsistent with a lower neighbor assignment, the conflict is reported to the lower neighbor to change its assignment. However, both algorithms were restricted to solving problems with binary constraints.

Grubshtein and Meisels proposed in [5] a model of graphical games as an ADisCSP with a unique private (global) table constraint for each agent. The table constraint contains all tuples (joint strategies) of the neighbors that satisfy the ϵ -NE condition. They also proposed *asynchronous Nash backtracking* (ANT), the first asynchronous algorithm for solving ADisCSP with global constraints capable of finding ϵ -NE. ANT is an extension of ABT-1ph that handles asymmetric global (non-binary) constraints. ANT achieves orders of magnitude improvements over NashProp. However, ANT only uses the global constraints as checkers. In addition its handling of global constraints produces chronological backtracks (thrashing), as all value assignments for agents in the constraint are considered, even if they are not the cause of the conflict. Recent developments in DisCSP have shown how to make more effective use of global constraints, exploiting their pruning power, and backjumping closer to the point of conflict [1, 16].

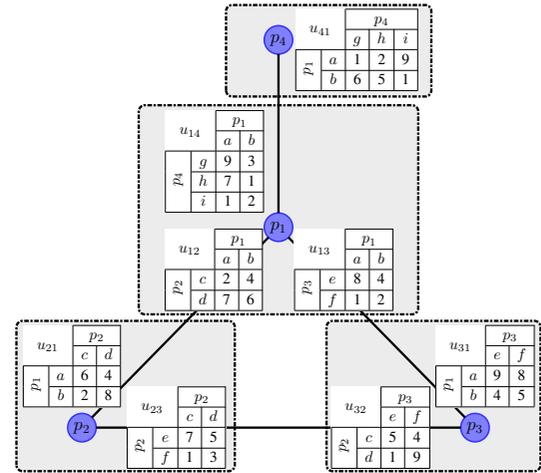
3 Preliminaries

In this section, we introduce some basic notation and terminology for game theory, and describe the framework of hypergraphical games before presenting the distributed constraint satisfaction formalism.

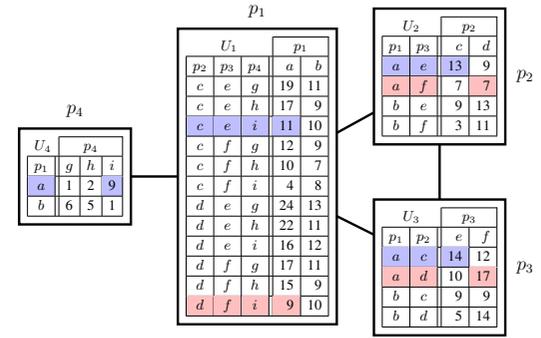
3.1 Game Theory

An n -player game in *normal-form* is a tuple $(\mathcal{P}, \{S_i, U_i\}_{p_i \in \mathcal{P}})$ where $\mathcal{P} = \{p_1, \dots, p_n\}$ is a set of n players (agents). For each agent $p_i \in \mathcal{P}$, S_i is a finite set of actions or pure strategies, and $U_i : S \rightarrow \mathbb{R}$ where $S = \prod_{i=1}^n S_i$, is a local utility hypermatrix/function that specifies the payoff for player p_i under each strategy profile $s \in S$. The joint strategic choice of all agents other than agent p_i is denoted by s_{-i} , and $s = (s_i, s_{-i})$. $U_i(s)$ is the utility that player p_i receives when, for all players $p_j \in \mathcal{P}$, p_j plays $s_j \in s$. The neighbors of agent p_i are all other agents in the normal-form game, i. e., $\mathcal{N}_i = \mathcal{P} \setminus p_i$. In the rest of the paper, we assume all agents have the same number of strategies, i. e., $\forall p_i \in \mathcal{P}, |S_i| = d$. The representation size of each local utility hypermatrix is exponential in the number of players, i. e., $O(d^n)$. Motivated by scenarios where an agent's utility is directly dependent on only a subset of the total number of agents, researchers devoted a considerable effort to develop compact representations following the graphical game model [8].

A *hypergraph* is a pair (V, E) where V is a set of vertices, and E is a set of non-empty subsets of V called hyperedges. [13] introduced



(a) Graphical polymatrix games



(b) Graphical game representation

Figure 1. An example of a hypergraphical game.

hypergraphical games, in which each agent p_i is involved in simultaneous (local) normal-form subgames, \mathcal{G}_i . A hypergraphical game is described by a hypergraph (\mathcal{P}, E) where each hyperedge $h \in E$ represents an explicit subgame involving the players in $h \subseteq \mathcal{P}$. The strategy set of each agent p_i is the same in all subgames in \mathcal{G}_i . The payoff function of p_i is the sum of all p_i 's payoffs in all \mathcal{G}_i . The neighbors of agent p_i , \mathcal{N}_i , is the union of its neighbors in all subgames in \mathcal{G}_i . The degree of agent p_i is denoted by $\kappa_i = |\mathcal{N}_i|$. The utility of an agent is directly dependent on its neighbors, i. e., $U_i : S_i \times S_j \rightarrow \mathbb{R}$. From now on, s_{-i} is the joint strategy of \mathcal{N}_i .

Hypergraphical games is a generalization of *graphical games* [8] where each agent is involved in exactly one subgame. In a graphical game, the representation size of p_i ' utility is exponential in the degree of the agent $O(d^{\kappa_i})$. Hypergraphical games is also a generalization of *graphical polymatrix games* where each agent is involved in simultaneous 2-player games [6]. The representation size of p_i 's utilities is $O(\kappa_i \cdot d^2)$.

In graphical polymatrix games, we can represent the utility function of agent p_i , U_i , by a set of (binary) utility functions u_{ij} for each $p_j \in \mathcal{N}_i$. The utility function $u_{ij}(s_i, s_j)$ represents the gain in utility of agent p_i when playing strategy $s_i \in S_i$ and agent p_j plays strategy $s_j \in S_j$, and p_i 's overall utility function becomes:

$$U_i(s) = U_i(s_{-i}, s_i) = \sum_{p_j \in \mathcal{N}_i, s_j \in s} u_{ij}(s_i, s_j) \quad (1)$$

For a strategy profile s , the *regret* $\delta_i(s)$ of an agent p_i is the highest additional reward p_i could have gained by changing its strategy,

assuming its neighbors' strategy choices remain the same:

$$\delta_i(s) = \max_{s'_i \in S_i} \{U_i(s_{-i}, s'_i) - U_i(s)\} \quad (2)$$

In strategic games, each rational agent would ideally play its best strategy given a fixed joint strategy of other agents. A configuration in which all agents selected a best strategy is called a *Nash Equilibrium* (NE). Specifically, a NE is a strategy profile s where each player's regret is 0 (i. e., $\forall p_i \in \mathcal{P}, \delta_i(s) = 0$). Given a joint strategy of other players s_{-i} , the *best response* of agent p_i , $BR_i(s_{-i})$, is the strategies which produces the maximal gain for agent p_i (i. e., $BR_i(s_{-i}) = \{s_i | \delta_i(s_{-i}, s_i) = 0\}$).

An *approximate Nash Equilibrium* (ϵ -NE) represents scenarios where agents are satisfied with choices whose payoff is sufficiently close to the maximum. Formally, a strategy profile s is an ϵ -NE if each player's regret is at most ϵ , i. e., $\forall p_i \in \mathcal{P}, \delta_i(s) \leq \epsilon$. Given a joint strategy of other players s_{-i} , the approximate best response set of agent p_i is defined as: $\epsilon\text{-BR}_i(s_{-i}) = \{s_i | \delta_i(s_{-i}, s_i) \leq \epsilon\}$.

Figure 1 shows a simple hypergraphical game with 4 agents: p_1, p_2, p_3 and p_4 having the following strategy sets $S_1 = \{a, b\}$, $S_2 = \{c, d\}$, $S_3 = \{e, f\}$ and $S_4 = \{g, h, i\}$. Figure 1(a) shows the original representation in graphical polymatrix games and Figure 1(b) shows a representation of the same instance in graphical games. Agent p_1 is involved in three 2-player subgames with p_2, p_3 and p_4 represented respectively by the utilities u_{12}, u_{13} , and u_{14} . Agent p_2 is involved in two subgames with p_1 (u_{21}) and p_3 (u_{23}). Agent p_3 is involved in two subgames with p_1 (u_{31}) and p_2 (u_{32}). Agent p_4 is involved in one 2-player subgame with p_1 represented by utility u_{41} . This problem has one NE [$p_1 = a, p_2 = c, p_3 = e, p_4 = i$] and one ϵ -NE in addition to the NE where $\epsilon = 3$, that is, [$p_1 = a, p_2 = d, p_3 = f, p_4 = i$].

3.2 CSP & Asymmetric Distributed CSP

The *constraint satisfaction problem* (CSP) is a triple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, where $\mathcal{X} = \{x_1, \dots, x_n\}$ is a set of n variables, $\mathcal{D} = \{D(x_1), \dots, D(x_n)\}$ is a set of domains, where $D(x_i)$ is a finite set of values from which one value must be assigned to variable x_i , and \mathcal{C} is a set of constraints. A constraint $c_k(X) \in \mathcal{C}$, on the ordered subset of variables $X = (x_{j_1}, \dots, x_{j_k}) \subseteq \mathcal{X}$, is $c_k(X) \subseteq D(x_{j_1}) \times \dots \times D(x_{j_k})$, and specifies the tuples of values which may be assigned simultaneously to the variables in X . $c_k(X)$ can be represented extensionally or intensionally. $|X|$ is the *arity* of $c_k(X)$, and X is its *scope*. A global constraint is defined on a set of variables and thus an instance of the constraint may have arbitrary arity. A *solution* is an assignment to each variable of a value from its domain, satisfying all the constraints.

Asymmetric distributed CSP (ADisCSP) [3] models problems where variables and constraints are held by distinct agents. ADisCSP is a 4-tuple $(\mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C})$, where \mathcal{X}, \mathcal{D} and \mathcal{C} are as above, and $\mathcal{A} = \{a_1, \dots, a_m\}$ is a set of m agents. Each variable $x_i \in \mathcal{X}$ is controlled by a single agent in \mathcal{A} . During a solution process, only the agent which controls a variable can assign a value to this variable. In an ADisCSP, constraints are private and only the agent, a_i , that holds a constraint knows it while other agents involved in that constraint are only aware that a_i constrains their variable without knowing the nature of that constraint or its scope. We denote by $\mathcal{C}_i \subseteq \mathcal{C}$ all constraints held by a_i . As in CSP, a *solution* to an ADisCSP is an assignment to each variable of a value from its domain, satisfying all the constraints. For simplicity and without loss of generality, we assume each agent controls exactly one variable and use the two terms interchangeably (i. e., $m = n$).

4 Hypergraphical Games as ADisCSP

We now present a model of the problem of finding ϵ -NE in hypergraphical games as ADisCSP, where agents can control a local CSP that allows them to maintain the approximate best responses.

The straightforward modeling of a hypergraphical game $(\mathcal{P}, \{S_i, U_i\}_{p_i \in \mathcal{P}})$ into an ADisCSP is to represent each player $p_i \in \mathcal{P}$ by an agent $a_i \in \mathcal{A}$ having a single local variable x_i that can take its value from the strategy set of player p_i , i. e., $D(x_i) = S_i$. In the following we use the terms agent, player and variable interchangeably, (i. e., $a_k = p_k = x_k$). In addition, we consider a generic agent a_i . $\mathcal{X}_i = \{x_i, x_j | x_j \in \mathcal{N}_i\}$ denotes the variables (or copies) maintained by a_i . Agent a_i encodes the problem of finding an ϵ -NE (finding an ϵ -approximate best strategy) by a single constraint $c_i(\mathcal{X}_i)$ requiring that the regret, Eq. (2), is less than or equals ϵ , i. e., $c_i(\mathcal{X}_i) : \delta_i(s) \leq \epsilon$ where $s = (s_i, s_{-i})$ is a joint strategy (the assignments) of agents in \mathcal{X}_i . Thus, $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_m\}$ where $\mathcal{C}_i = \{c_i(\mathcal{X}_i)\}$. The agent a_i that holds the constraint $c_i(\mathcal{X}_i)$ is the only agent that knows it, and must ensure it is satisfied, given the joint assignment of all its neighbors. To evaluate this constraint agent a_i needs to maintain local copies of its neighbors' variables in \mathcal{X}_i . Finding ϵ -NE distributively in a hypergraphical game is then equivalent to solving the ADisCSP model above.

Proposition 1. *Let \mathcal{M} be an ADisCSP model of a hypergraphical game \mathcal{H} . A solution of \mathcal{M} is an ϵ -NE of \mathcal{H} .*

Proof. (Sketch) A solution to \mathcal{M} is an assignment to each variable of a value from its domain, satisfying all the constraints. Thus, each player is assigned a strategy and every agent a_i 's private constraint, $c_i(\mathcal{X}_i)$, is satisfied. Satisfying each constraint $c_i(\mathcal{X}_i)$ means that each agent regret is less than ϵ ($\delta_i(s) \leq \epsilon$). Thus, each agent assignment is an approximate best response to its neighbors assignments. Hence, a solution of \mathcal{M} is an ϵ -NE of \mathcal{H} . \square

If agents are allowed mixed strategies (i. e. a probability distribution over deterministic strategies), then the discretization scheme proposed in [8] guarantees that an ϵ -NE always exists. Thus, the ADisCSP model of the hypergraphical game is always satisfiable. For more details about mixed strategies and the existence of an ϵ -NE, we refer the reader to [8, 12, 11, 5]. If the agents are restricted to pure strategies, it is possible that no ϵ -NE exists. In such circumstances, the ADisCSP model would have no solution, and any algorithm should report failure.

In the following we propose ϵ -BRConstraint, a new incremental global constraint to ensure an agent's value is an approximate best response, that is efficient in memory and allows efficient propagation. The ϵ -BRConstraint enforces $\delta_i(s) \leq \epsilon$ for each agent a_i (i. e., it is an implementation of $c_i(\mathcal{X}_i)$). For simplicity, we restrict our attention to graphical polymatrix games. However, our constraint and algorithm will represent any hypergraphical game including graphical games and graphical polymatrix games. Our experiments are against the state-of-the-art distributed algorithm for graphical games, and we experiment with the same class of problems that that algorithm was tested on. These problems happen to be graphical polymatrix games.

The Constraint ϵ -BRConstraint

When the system is at an ϵ -NE, each agent has assigned to its variable an approximate best strategy with respect to its neighbors assignments. Thus, at each step in the distributed algorithm, an agent a_i should prune all strategies that are dominated by others. In agent

a_i , ϵ -BRConstraint filters the domain of x_i to prune dominated strategies with respect to the (subset of) decisions of other agents in \mathcal{N}_i . ϵ -BRConstraint ($\epsilon, x_i, \{x_j\}_{x_j \in \mathcal{N}_i}, \{u_{ij}\}_{x_j \in \mathcal{N}_i}$) takes as parameters all variables in \mathcal{X}_i and utility functions u_{ij} in addition to ϵ .³ Before presenting the constraint behavior we first consider the auxiliary variables used inside ϵ -BRConstraint to filter dominated strategies. In addition to x_i and copies of neighbors' variables $x_j \in \mathcal{N}_i$, a_i has in \mathcal{X}_i the following integer variables:

- $v_j[s_i]$ for each strategy $s_i \in D(x_i)$, and each utility u_{ij} . The variable $v_j[s_i]$ represents the gain in utility of agent a_i when playing strategy s_i with respect to x_j possible strategies. This variable ranges over the possible utilities in u_{ij} when a_i plays s_i , i.e., $D(v_j[s_i]) = \{u_{ij}(s_i, s_j) | s_j \in D(x_j)\}$.
- $y[s_i]$ for each strategy $s_i \in D(x_i)$. The variable $y[s_i]$ is used to maintain the minimum reward gained when a_i chooses to play strategy s_i .

$$y[s_i] = \sum_{x_j \in \mathcal{N}_i} \min\{D(v_j[s_i])\} \quad (3)$$

- $z[s_i]$ for each strategy $s_i \in D(x_i)$. The variable $z[s_i]$ is used to maintain the maximum reward gained when a_i chooses to play strategy s_i .

$$z[s_i] = \sum_{x_j \in \mathcal{N}_i} \max\{D(v_j[s_i])\} \quad (4)$$

For each $s_i \in D(x_i)$, we need to represent the relation between the values of the variables x_i , x_j and $v_j[s_i]$, and maintain $v_j[s_i] = u_{ij}(s_i, s_j)$. As soon as utility $v_j[s_i]$ is updated, some now inconsistent values of x_j can be removed from consideration. Similarly, when $D(x_j)$ is updated (e.g., a value is removed from $D(x_j)$), the utility variable $v_j[s_i]$ can be updated correspondingly. Each time a domain of x_j or $v_j[s_i]$ is changed, the domain of the other variable is updated to keep only values having a support on other variables, i.e., we need to ensure that $\exists s_j \in D(x_j) \wedge r \in D(v_j[s_i])$, $r = u_{ij}(s_i, s_j)$.⁴

We need also to make the correspondence between the values of $v_j[s_i]$ and those of $y[s_i]$ and $z[s_i]$ for each $s_i \in D(x_i)$ by ensuring Eq. (3) and Eq. (4). We maintain a support for $y[s_i]$ (resp. $z[s_i]$) on variable $v_j[s_i]$ and we only update that support and its corresponding reward on $y[s_i]$ (resp. $z[s_i]$) when the lower bound (resp. the upper bound) of the domain of $v_j[s_i]$ has changed.

By maintaining the above variables and properties on their domain changes, ϵ -BRConstraint is able to detect and remove from $D(x_i)$ the dominated strategies using the $y[s_i]$ and $z[s_i]$ variables. We say that a strategy $s_i \in D(x_i)$ is *dominated* if the largest utility that a_i can gain when choosing strategy s_i (i.e., $z[s_i]$) is lower than the minimal reward that a_i can gain by choosing another strategy s'_i where $s'_i = \arg \max_{s'_i \in D(x_i)} \{y[s'_i]\}$ taking ϵ in consideration. Specifically, a strategy $s_i \in D(x_i)$ is dominated iff:

$$z[s_i] < \max_{s'_i \in D(x_i)} \{y[s'_i]\} - \epsilon \quad (5)$$

Removing dominated strategies s_i from $D(x_i)$ is safe because a_i will never play s_i , Eq. (5). Filtering in ϵ -BRConstraint is based on pruning all dominated strategies each time they are detected with changes on domains of variables in \mathcal{X}_i without needing their full joint strategy (assignments).

³ In the general hypergraphical games, u_{ij} will be replaced by payoffs matrix of each subgame.

⁴ Maintaining $v_j[s_i]$ values follows the same scheme as the element constraint [14]. However, in the general case the index variable is a combination of indexes of all neighborhood values in the subgame.

x_2	x_3	x_4	x_1
c	e	g	b
c	e	h	b
c	e	i	b
c	f	g	b
c	f	h	b
c	f	i	a
d	e	g	b
d	e	h	b
d	e	i	b
d	f	g	b
d	f	h	b
d	f	i	a

$$\begin{aligned}
D(v_2[a]) &= \{2, 7\} \\
D(v_2[b]) &= \{4, 6\} & D(y[a]) &= \{4..24\} \\
D(v_3[a]) &= \{1, 8\} & D(y[b]) &= \{7..13\} \\
D(v_3[b]) &= \{2, 4\} & D(z[a]) &= \{4..24\} \\
D(v_4[a]) &= \{1, 7, 9\} & D(z[b]) &= \{7..13\} \\
D(v_4[b]) &= \{1, 2, 3\}
\end{aligned}$$

$$\begin{aligned}
v_2[a] &= u_{12}(a, s_2) = [2, 7] \\
v_2[b] &= u_{12}(b, s_2) = [4, 6] \\
v_3[a] &= u_{13}(a, s_3) = [8, 1] \\
v_3[b] &= u_{13}(b, s_3) = [4, 2] \\
v_4[a] &= u_{14}(a, s_4) = [9, 7, 1] \\
v_4[b] &= u_{14}(b, s_4) = [3, 1, 2]
\end{aligned}$$

(a) Table constraint encoding (b) ϵ -BRConstraint for $c_1(\mathcal{X}_1)$. $c_1(\mathcal{X}_1)$ used in [5].

Figure 2. The encoding of ANT and AABT in a_1 of the constraint $c_1(\mathcal{X}_1)$ in the problem shown in Figure 1.

4.1 Memory requirements

In a hypergraphical game, in [5], $c_i(\mathcal{X}_i)$ is represented in extensional form using a table constraint containing all tuples (joint strategy) $s \in D(x_i) \times \prod_{x_j \in \mathcal{N}_i} D(x_j)$ that satisfies $c_i(\mathcal{X}_i)$. A tuple s satisfies $c_i(\mathcal{X}_i)$ if it yields a maximal gain to agent a_i , i.e., $\delta_i(s) \leq \epsilon$. Thus, the utility functions of all subgames of a_i are encoded and represented by a large table constraint requiring a memory size of $O(d^{\kappa_i+1})$. The table constraint encoding the constraint $c_1(\mathcal{X}_1)$ of agent a_i of the example presented in Figure 1 is shown in Figure 2(a). All tuples satisfying $c_1(\mathcal{X}_1)$ are represented in that table.

In our new model, ϵ -BRConstraint maintains a_i 's utility in each subgame in \mathcal{G}_i . Thus, the representation size in ϵ -BRConstraint is similar to that required by the hypergraphical game in each agent, i.e., $O(|\mathcal{G}_i| \cdot d^{\gamma+1})$ where γ is largest neighborhood in subgames \mathcal{G}_i and $|\mathcal{G}_i|$ is the number of subgames in \mathcal{G}_i . In polymatrix games, this representation is polyspace $O(\kappa_i \cdot d^2)$. The encoding of the constraint $c_1(\mathcal{X}_1)$ of agent a_i of the example presented in Figure 1 is shown in Figure 2(b).

In ANT the handling of the global constraint does not exploit its filtering power. In addition, when a dead-end occurs, a no-good is produced from all neighbors' assignments and then sent to the lowest neighbor in the ordering. This produces chronological backtracks (thrashing). In the following, we propose a new algorithm for solving ADIS CSP with global constraints, that exploits the pruning power of global constraints and produces no-goods closer to the (real) point of conflict.

5 Asymmetric Asynchronous Backtracking

Asymmetric asynchronous backtracking (AABT) is an asynchronous algorithm for solving ADIS CSP that allows agents to keep their constraints private. In AABT agents operate asynchronously, but are subject to a known total priority order. AABT combines a distributed search procedure with a failure learning mechanism to perform intelligent backtracking. Intelligent backtracking techniques usually store an explanation for each value removal. Such explanations are computed on-the-fly on each domain reduction.

In AABT, each agent a_i tries to solve its local CSP _{i} defined by \mathcal{X}_i , their domains, and \mathcal{C}_i . Solving a CSP _{i} is achieved by interleaving search with propagation. The search can be regarded as the dy-

Algorithm 1: AABT algorithm running by agent a_i .

```

procedure AABT ()
1.  $E_i^+ \leftarrow \{a_j \mid a_i \in \mathcal{N}_j\}$ ;  $end \leftarrow \text{false}$ ;  $x_i \leftarrow nil$ ;
2. assignVariable ();
3. while ( $\neg end$ ) do
4.    $msg \leftarrow \text{getMsg}()$ ;
5.   switch ( $msg.type$ ) do
6.     ok?: processOk ( $msg.var$ ,  $msg.tag$ );
7.     ngd: processNogood ( $msg.sender$ ,  $msg.ngd$ );
8.     adl: processAddLink ( $msg.sender$ );
9.     stp:  $end \leftarrow \text{true}$ ;

procedure assignVariable ()
10. while ( $x_i = nil \wedge \neg end$ ) do
11.   propagate ( $x_i = D(x_i).peek()$ );
12.   if ( $\exists x_k \in \mathcal{X}_i \mid D(x_k) = \emptyset$ ) then
13.     repair (); // An empty domain has been found
14.   else
15.      $t_i \leftarrow t_i + 1$ ;
16.     sendMsg: ok? $\langle x_i = s_i, t_i \rangle$  to  $E_i^+$ ;

procedure propagate ( $x_k = s'_k$ )
17. remove ( $expl(x_l \neq s_l)$ ) s.t.  $x_k = s_k \in expl(x_l \neq s_l)$ ;
18.  $D(x_k) \leftarrow \{s'_k\}$  s.t.  $expl(x_k \neq s_k) \leftarrow x_k = s'_k$ ;
19.  $C_i.propagate()$ ;

procedure processOk ( $x'_j, t'_j$ )
20. if ( $t'_j \geq t_j$ ) then
21.   propagate ( $x_j = x'_j$ );
22.   if ( $\exists x_k \in \mathcal{X}_i \mid D(x_k) = \emptyset$ ) then repair ();
23.   assignVariable ();

procedure processNogood ( $a_j, ng$ )
24. if ( $\forall x_l \in \{ng \cap \mathcal{X}_i, ng[x_l] = \mathcal{X}_i[x_l]\}$ ) then
25.    $Links \leftarrow \{ng \cup \mathcal{X}_i\} \setminus \mathcal{X}_i$ ;
26.    $\mathcal{X}_i \leftarrow \mathcal{X}_i \cup Links$ ;
27.   sendMsg: adl $\langle \rangle$  to  $Links$ ;
28.   learn ( $ng$ );
29. else if ( $ng[x_i] = x_i$ ) then
30.   sendMsg: ok? $\langle x_i = s_i, t_i \rangle$  to  $a_j$ ;

procedure repair ()
31.  $ng \leftarrow \bigwedge_{s_k \in D(x_k)} expl(x_k \neq s_k)$ ; //  $D(x_k) = \emptyset$  *
32. if ( $ng = \emptyset$ ) then
33.   sendMsg: stp $\langle \rangle$  to  $A \setminus a_i$ ;
34.    $end \leftarrow \text{true}$ ;
35. else learn ( $ng$ );

procedure learn ( $ng$ )
36. Let  $x_t$  be the lowest variable in  $ng$ ;
37. if ( $x_t \neq x_i$ ) then sendMsg: ngd $\langle ng \rangle$  to  $a_t$ ;
38.  $expl(x_t \neq s_t) \leftarrow \{ng \setminus x_t = s_t\}$ ;
39. remove ( $expl(x_l \neq s_l)$ ) s.t.  $x_t = s_t \in expl(x_l \neq s_l)$ ;
40.  $C_i.propagate()$ ;
41. if ( $\exists x_k \in \mathcal{X}_i \mid D(x_k) = \emptyset$ ) then repair ();
42. assignVariable ();

procedure processAddLink ( $a_j$ )
43.  $E_i^+ \leftarrow E_i^+ \cup a_j$ ;
44. sendMsg: ok? $\langle x_i = s_i, t_i \rangle$  to  $a_j$ ;

```

dynamic addition of constraints (decision constraints) to C_i and retractions (backtracks) [7]. For simplicity, we restrict ourselves to decision constraints that are of the form $x_k = s_k$, i. e., assignments of values to variables.⁵ Each assignment is followed by the propagation of C_i with respect to \mathcal{X}_i 's domains. This propagation may result in a value removal ($x_k \neq s_k$) for a variable $x_k \in \mathcal{X}_i$. We define an *explanation*, $expl(x_k \neq s_k)$, of that value removal (i. e., $x_k \neq s_k$) by the set of decision constraints $x_j = s_j, \dots, x_l = s_l$ (assignments), such that $(x_j = s_j \wedge \dots \wedge x_l = s_l \wedge x_k = s_k)$ is globally inconsistent. During search, a failure (a domain wipeout) may occur leading to *no-goods* computations. A no-good can be regarded as a subset of the assignments made so far that caused a failure (i. e., $\bigwedge_{s_k \in D(x_k)} expl(x_k \neq s_k)$)

where $D(x_k) = \emptyset$.

AABT agents exchange the following message types:

ok?: used to notify its recipients of a new assignment associated with the current counter t_i of agent a_i , used to discard obsolete assignments.

ngd: used to report a no-good to another agent, requesting the removal of its value.

adl: used to request the additional of a link to the receiver.

The pseudo-code of AABT executed by each agent a_i is presented in Algorithm 1. In the following, s_i will represent the current value assigned to x_i and t_i the counter tagging s_i used for the timestamp mechanism. “ $x_i = nil$ ” means that x_i is unassigned. After initialization, each agent assigns a value and informs all agents in E_i^+ of its decision (assignVariable call, line 2) by sending them **ok?** messages. E_i^+ are agents having a constraint involving a_i 's variable (line 1). Then, a loop considers the reception of the possible message types. If no message is traveling through the network, the state of quiescence is reached meaning that a global solution is found. The solution is given by the current variables' assignments. The quiescence state can be detected by a specialized algorithm [4].

When an agent a_i receives an **ok?** message from a_j it calls procedure processOk (line 6). If the received assignment has a larger counter than that received beforehand, it is accepted, otherwise it is discarded (line 20). If the assignment is accepted, a_i calls procedure propagate to propagate C_i after adding the newly received assignment as a decision constraint $x_j = s'_j$, line 21. If the propagation results in a domain wipeout procedure repair is called to resolve the conflict, line 22. Finally, assignVariable is called (line 23) to assign a new consistent value to x_i if its value has been pruned by the propagation of C_i .

When calling procedure propagate ($x_k = s'_k$), the value of x_k is set to s'_k . Next, all explanations not relevant to this new assignment, i. e., $expl(x_k \neq s_k)$ containing $x_k = s_k$ where $s'_k \neq s_k$, are removed (line 17).⁶ Then, the domain of x_k is reduced to a singleton $D(x_k) = \{s'_k\}$ with $x_k = s'_k$ as explanation, line 18. Finally, the constraints in C_i that might be affected by the domains' changes above are propagated, line 19.

When every value (s_k) of a variable $x_k \in \mathcal{X}_i$ is ruled out by an explanation $expl(x_k \neq s_k)$, the procedure repair is called to resolve the conflict (lines 13, 22 and 41). The conflict is resolved by computing a new no-good ng from the conjunction of these explanations, i. e., $expl(x_k \neq s_k)$, line 31. If the new no-good ng is empty, a_i terminates execution after sending a **stp** message to all agents in the system meaning that the problem is unsolvable (line 32). Otherwise,

⁵ Agent a_i can not take decision for other agents variables x_j , i. e., $i \neq j$.

⁶ The removal of some explanations $expl(x_k \neq s_k)$ does not imply the restoration of s_k to $D(x_k)$ unless the propagation of constraints involving x_k allows that.

it calls procedure `learn` (ng), line 35. Let x_t be the variable having the lowest priority in ng . If x_t is different than x_i , ng is reported to a_t through a **ngd** message, lines 36 and 37. In AABT, the backtracking target x_t is always that having the lowest priority in ng . Agent x_t can be a higher (as in ABT) but also a lower priority agent. Thus, we guarantee that the conflict is always reported to the agent with lowest priority in the conflict. Next, a new explanation from ng is used to justify the removal of the value of x_t , i.e., $expl(x_t \neq s_t) \leftarrow \{ng \setminus x_t = s_t\}$, line 38. All explanations containing the assignment of x_t ($x_t = s_t$) are removed because they are not valid anymore, line 39. The constraints in \mathcal{C}_i are then propagated to check the new changes and if a failure occurs again `repair` is called, line 41. Finally, `assignVariable` is called to check if x_i needs to be assigned (line 42).

When a **ngd** message is received (line 7), a_i checks the validity of the received no-good (procedure `processNogood` call). If the assignments of the received no-good are consistent with those stored locally, this no-good is valid (line 24). Then, agent a_i sends a link request to non linked agents having variables in ng (lines 25 to 27). Next, a_i calls procedure `learn` (ng), line 28. If the ng is not valid but is consistent with the current assignment of x_i , a_i sends an **ok?** message to the generator of ng , lines 29 and 30.

When receiving an **adl** message (lines 43 and 44) a_i adds a new link to the request sender (a_j) and sends an **ok?** message to inform a_j of its assignment.

5.1 Privacy

In AABT, to use the power of the filtering algorithms of global constraints, each agent maintains a copy of its neighborhood domains. However, agents in AABT only require to know the initial domain of their neighbors and do not need their actual domain. For the case of the hypergraphical game as ADisCSP, an agent can construct the domain of a neighbor based on the payoff matrices of local sub-games. At each stage of AABT, the privacy of agents' current domains is preserved. In AABT, agents exchange their values with their neighbors. Thus, in AABT there is no privacy of assignments. In AABT, agents do not have to share their constraints/scopes to solve the problem. Thus, AABT preserves privacy of constraints and privacy of agents topologies. However, during the solving process some information is exchanged (value assignments and no-goods) between agents, thus leaking information about agents' constraints.

A method to evaluate the constraint privacy using entropy as a quantitative measure for privacy loss has been proposed in [10, 3]. This method measures the percentage of the conflicts in a table constraint held by an agent that are revealed to another agent involved in a conflict induced from **ok?** and **ngd** messages. However, this method only applies to ADisCSP with binary table constraints. Thus, this method can not be used to evaluate the privacy loss in AABT. We believe that studying the privacy loss in distributed algorithms for solving DisCSP with global constraints (e.g., ANT and AABT) is an open research area that needs to be investigated.

5.2 Theoretical Analysis

Here we prove that AABT is sound, complete and terminates.

Theorem 1. *AABT terminates.*

Proof. AABT stops its execution in two cases: when an empty no-good has been generated meaning that the problem is unsolvable or when the network reaches a quiescent state reporting a solution. To

prove that AABT terminates, we need to prove that AABT reaches one of these two cases in finite time, i.e., agents can never fall into an infinite loop cycling among their possible values. In the following, we prove by induction on the agent ordering that agents can never fall into an infinite loop. Let assume a lexicographic ordering on agents. The base case for induction ($i = 1$) is obvious. Unlike ABT, a_1 can receive **ok?** messages from its neighbors (having a lower priority). When receiving these **ok?** messages a_1 may generate new no-goods as results of propagating its constraints. Agent a_1 may generate three categories of no-goods. The first are empty no-goods. This category stops the algorithm execution. The second are no-goods containing a lower neighbor. Those no-goods are transmitted to the lowest agent involved in. The third category are singleton no-goods containing a value of a_1 . Agent a_1 may also receive **ngd** messages from lower priority agents. Now, all no-goods contained in **ngd** messages a_1 receives are singleton because in AABT the generated no-good is always sent to the agent having the lowest priority in it (lines 36 and 37). Hence, when agent a_1 proposes a possible value, it will not change it unless it receives or itself produces singleton no-goods ruling out this value. Values in singleton no-goods are removed once and for all from the domain of a_1 . Because its domain is finite, a_1 cannot fall into an infinite loop.

Now, assume that agents higher than agent a_i ($i > 2$), i.e., a_1 to a_{i-1} ($i > 2$), are in a stable state, i.e., they are all assigned values to their variables and do not change their values. In the following, we show that agent a_i never falls into an infinite loop. After processing **ok?** and **ngd** messages it receives, agent a_i may generate contradictions (no-goods) as results of propagating its local constraints. The categories of no-goods agent a_i may generate are: (i) empty no-goods, (ii) no-goods containing lower neighbors, (iii) no-goods containing the agents a_1 to a_{i-1} , and (iv) no-goods containing the agents a_1 to a_i . No-goods of (i) causes the algorithm to stop, those of (ii) are forwarded to a lower agent and can not cause a_i to fall in infinite loop. No-goods of (iii) breaks our assumption because they are forwarded to a higher agent than we assumed to be in a stable state, causing it to change its value. Thus, only (iv) may affect the termination of a_i . Agent a_i may also receive **ngd** messages from other agents (higher or lower). However, all no-goods contained in **ngd** messages a_i receives contain only the agents a_1 to a_i , i.e., category (iv), because in AABT no-goods are always sent to the lowest agent involved in. Since agents a_1 to a_{i-1} are in a stable state, these no-goods are valid for a_i (the assignments on these no-goods are consistent with those stored locally).⁷ Thus, a_i will remove its value and will not assign it again while at least one of agents a_1 to a_{i-1} does not change its value. Because its domain is finite, a_i will either eventually change its assignment with a different value or exhaust the possible values and send a no-good to a higher agent, i.e., one of $a_1 \dots a_{i-1}$. This no-good will cause an agent that we assumed to be in a stable state, not to be in a stable state. Hence, agent a_i can never fall into an infinite loop for a given stable state of a_1 to a_{i-1} . By induction we have that agents can never fall into an infinite loop and AABT is thus guaranteed to terminate. \square

A global solution is reported when the network has reached quiescence, meaning that all agents are idle and no message is transmitting through the network. To prove that AABT is sound, we should first establish the two following lemmas about when the network reaches the quiescent state σ .

⁷ Note here that we discuss only no-goods consistent with a_1 to a_i , because inconsistent ones are discarded.

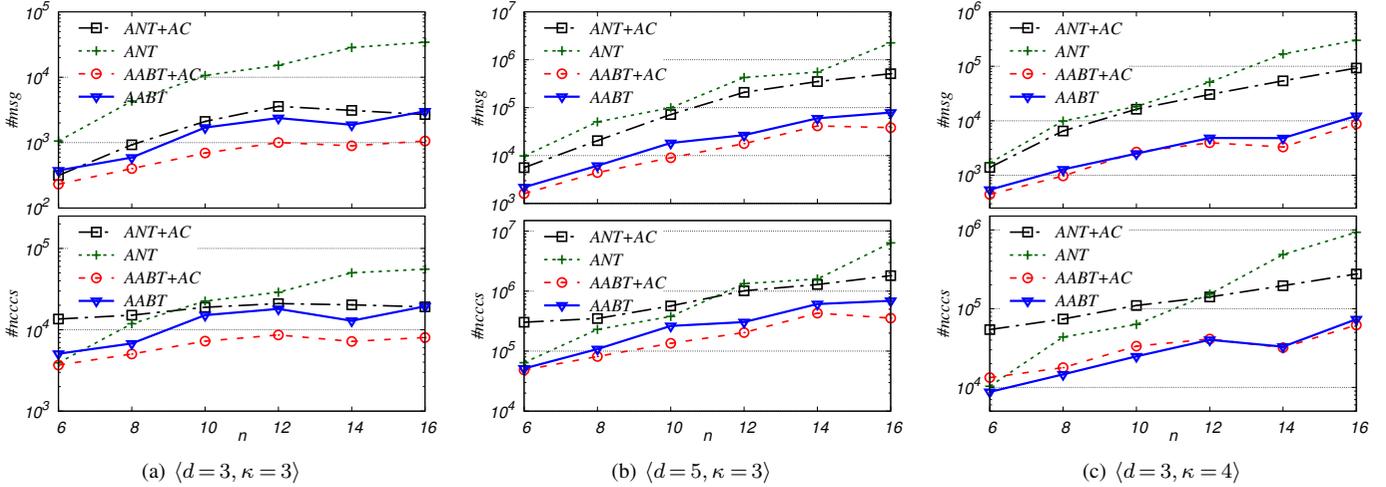


Figure 3. The $\#msg$ and $\#ncccs$ for solving random regular graphical polymatrix games when varying the number of agents in logarithmic scale.

Lemma 1. *When reaching a quiescent state σ , every agent is assigned, and its assignment is known by all its neighbors.*

Proof. (Sketch) Assume the system reaches a quiescent state σ . According to Algorithm 1, each time an agent a_i is unassigned, it immediately calls procedure `assignVariable` (lines 23 and 42). Before exiting its loop and being idle allowing the system to reach σ , `assignVariable` guarantees that either (i) an empty domain has been found when calling `repair` (line 13) meaning that the problem is unsolvable or (ii) a value is assigned to x_i . (i) contradicts our assumption. Thus, when reaching σ every agent a_i is assigned. Now, let v_i be the value assigned to x_i when reaching σ . Each neighbor of a_i , say a_j , should receive an `ok?` message from a_i containing its decision, i. e., $x_i = v_i$ (line 16). The only case where agent a_j removes $x_i = v_i$ is when it sends a no-good message to a_i . We can easily see that this message is either not obsolete, in which case a_i will change its value v_i and breaks our assumption, or obsolete, which means that an `ok?` message has not yet reached a_j which breaks our quiescence assumption. Hence, when reaching σ all agents know the assignments of all their neighbors and no two agents store different assignments for the same variable. \square

Lemma 2. *When reaching σ , all constraints are satisfied.*

Proof. In AABT, each time a new value is assigned or a new message is received by an agent a_i , it immediately propagates its constraints C_i . When generating an empty domain, a_i calls procedure `repair` to repair inconsistent assignments. Hence, after processing each message all assignments stored locally satisfy all constraints of the agent. \square

Theorem 2. *AABT is sound.*

Proof. Direct from Lemmas 1 and 2. \square

Theorem 3. *AABT is complete.*

Proof. In AABT, agents only store valid explanations and no-goods. In addition, all explanations and no-goods are generated by logical inferences from existing constraints. Thus, the empty no-good cannot be inferred if the network is satisfiable. \square

Corollary 1. *AABT is a correct and complete solver for hypergraphical games.*

Proof. From Proposition 1 and Theorems 1 to 3. \square

6 Empirical Study

In this section we experimentally compare AABT to ANT [5], AABT+AC and ANT+AC [5]. In AABT+AC and ANT+AC arc consistency (AC) is performed in a preprocessing phase before running AABT and ANT. This processing phase is the table passing phase of NashProp [12]. We do not compare the algorithms to NashProp because in [5] it has been shown that ANT achieves orders of magnitude improvements over NashProp. All the problems used for evaluating both algorithms were generated randomly and then modified to ensure that at least one pure strategy Nash equilibrium exists. All experiments were performed on the DisChoco 2.0 platform [17], in which agents are simulated by Java threads that communicate only through message passing. We evaluate the performance of the algorithms by communication load and computation effort. Communication load is measured by the total number of exchanged messages among agents during algorithm execution ($\#msg$) [9]. Computation effort is measured by the number of non-concurrent constraint checks ($\#ncccs$) [19]. $\#ncccs$ is the metric used in distributed constraint solving to simulate the computation time. Algorithms are evaluated on two random benchmarks: random regular graphical polymatrix games, and random graphical polymatrix games.

Random regular graphical polymatrix games: are characterized by $\langle n, d, \kappa \rangle$, where n is the number of players/agents, d is the number of strategies per agent, and κ represents the degree of each agent. For each value combination a utility was uniformly selected in $\{0, \dots, 9\}$. We solved and report the average over 25 instances of four settings. The first and the fourth settings cover the experiments presented in [5]. In the three first settings, we fixed the number of strategies and the degree of each agent and varied the number of agents in the range 6..16 by a step of 2 to guarantee the graphs are κ -regular. In the first setting we generated 3 random connections for each agent ($\kappa = 3$) and fixed the number of strategies of each agent at $d = 3$. We then increase the number of strategies to $d = 5$ in the second setting and increase the degree of each agent to $\kappa = 4$ in the third setting.

Results are presented in Figure 3. Regarding the number of required messages to solve the problem ($\#msg$), AABT improves ANT by an order of magnitude in almost all instances of Figure 3.

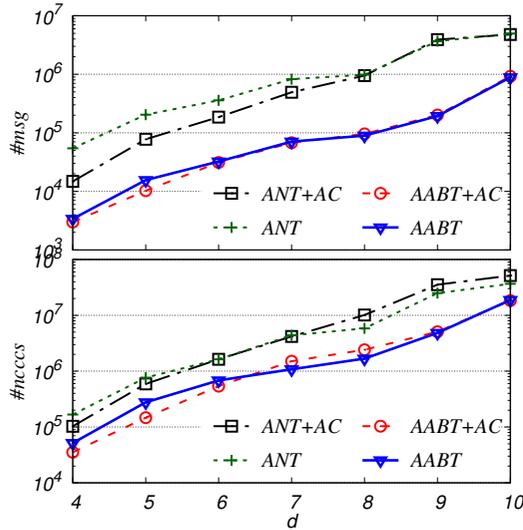


Figure 4. The $\#msg$ and $\#ncccs$ for solving random regular graphical polymatrix games when varying the number of strategies in logarithmic scale.

In the first setting (Figure 3(a)), the improvement factor ranges between 3 for small problems and 15 on larger ones. In the second setting (Figure 3(b)), the factor of improvement ranges between 5 for small problems and 29 on larger ones. In the third setting (Figure 3(c)), the factor of improvement ranges between 3 for small problems and 35 on larger ones. For $\#ncccs$, AABT outperforms ANT, although the improvement is less significant than in $\#msg$. In small problems the two algorithms perform similarly. However, in larger problems, the factor of improvement ranges from 4 in the first setting ($d=3, \kappa=3$) to 9 in the second ($d=5, \kappa=3$) and 15 in the third ($d=5, \kappa=4$). Regarding the AC, ANT+AC (resp. AABT+AC) always improves ANT (resp. AABT) in $\#msg$. For $\#ncccs$, AABT+AC always improves AABT, however, ANT+AC only improves ANT on problems with larger number of agents ($n > 10$). AABT always improves ANT+AC and the improvement is more significant (an order of magnitude improvement) in larger problems of the second and the third setting when increasing the number of strategies or the agents degrees. AABT+AC is clearly the best algorithm for solving all instances in the three settings.

In the fourth setting (Figure 4) we generated random 3-regular graphical polymatrix games ($\kappa=3$) where we fixed the number of agents at $n=10$ and varied the number of strategies of all agents in the range 4..10. Again, the results show that AABT outperforms ANT and ANT+AC in all instances. Regarding $\#ncccs$, the improvement factor ranges between 3 to 7. For $\#msg$, in instances with larger number of strategies the AABT improvement over ANT and ANT+AC is again an order of magnitude. AABT and AABT+AC algorithms perform almost similarly in all instances. ANT+AC shows small improvement over ANT on $\#msg$ for problems with smaller number of strategies.

Random graphical polymatrix games: are characterized by $\langle n, d, \rho \rangle$, where n is the number of players/agents, d is the number of strategies per player, ρ is the network connectivity defined as the ratio of existing binary utility functions. For each value combination a utility was uniformly selected from the set $\{0, \dots, 9\}$. For each $\rho \in \{0.4, \dots, 0.9\}$, we generated 25 instances in the class $\langle n=10, d=5, \rho \rangle$. We report the average over these instances in Figure 5. The results demonstrate that AABT improves ANT algorithm

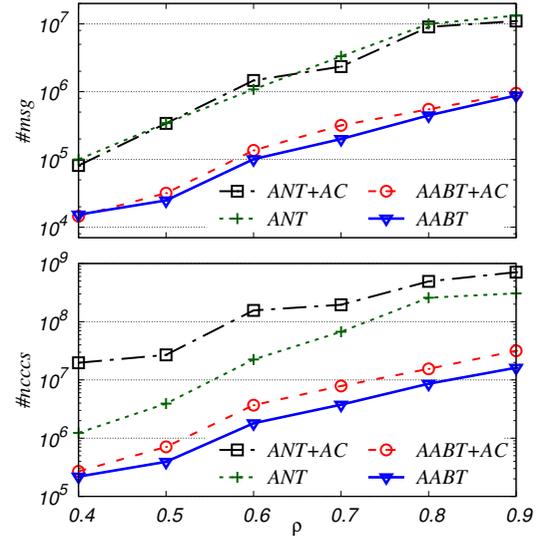


Figure 5. The $\#ncccs$ performed and $\#msg$ exchanged for solving problems where $\langle n=10, d=5, \rho \rangle$ in logarithmic scale.

in both $\#msg$ and $\#ncccs$. This improvement is over an order of magnitude on dense problems, $\rho \geq 6$. In contrast to κ -regular games, the improvement on $\#ncccs$ is more significant than on $\#msg$. Specifically, the improvement factor on $\#ncccs$ when $\rho=.8$ is 30 while it is 22 regarding the $\#msg$. Regarding AC, our results confirm those of [5] that if pruning is limited, AC can create a significant overhead.

Summary: In all our experimentation, AABT always improves ANT and ANT+AC in both metrics $\#msg$ and $\#ncccs$. This improvement is more significant on harder problems (when increasing the number of players/agents and/or the number of strategies of each player and/or the degree of each agent). In sparse problems of κ -regular graph $\kappa=3$, AABT improves the $\#ncccs$ slightly compared to ANT.⁸ However, this improvement is more significant on dense graphical polymatrix games. AABT+AC only improves AABT on sparse problems. AC is harmful for hard problems because it does not lead to domain filtering when problems are dense and/or large with a large number of strategies.

7 Conclusion

We studied the problem of finding an approximate Nash Equilibrium in hypergraphical games, an elegant framework for modeling collaboration in multi-agent systems within strategic environments. Our study is based on asymmetric distributed constraint satisfaction, an efficient tool for distributed problem solving allowing agents to keep their utilities private. We proposed a new model of hypergraphical games as an asymmetric DisCSP based on ϵ -BRConstraint, a new global constraint modeling hypergraphical games using the original compact representation of the subgames with a filtering algorithm of dominated strategies. Finally we introduced a new asynchronous algorithm for solving (asymmetric) DisCSP, in order to find Nash Equilibria for hypergraphical games. This achieves an order of magnitude improvement in messaging and in non-concurrent computation time on dense problems compared to state-of-the art algorithms.

⁸ In our implementation, table constraints are presented in lexicographic ordering of tuples and we use a dichotomic search ($\log(\kappa_i + 1)$) to check the consistency of each tuple.

REFERENCES

- [1] Christian Bessiere, Ismel Brito, Patricia Gutierrez, and Pedro Meseguer, 'Global constraints in distributed constraint satisfaction and optimization', *The Computer Journal*, (2013).
- [2] Christian Bessiere, Arnold Maestre, Ismel Brito, and Pedro Meseguer, 'Asynchronous backtracking without adding links: a new member in the ABT family', *Artif. Intel.*, **161**, 7–24, (2005).
- [3] Ismel Brito, Amnon Meisels, Pedro Meseguer, and Roie Zivan, 'Distributed Constraint Satisfaction with Partially Known Constraints', *Constraints*, **14**, 199–234, (2009).
- [4] K. Mani Chandy and Leslie Lamport, 'Distributed Snapshots: Determining Global States of Distributed Systems', *ACM Trans. on Computer Systems*, **3**(1), 63–75, (1985).
- [5] Alon Grubshtein and Amnon Meisels, 'Finding a nash equilibrium by asynchronous backtracking', in *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming*, CP'12, pp. 925–940, Berlin, Heidelberg, (2012). Springer-Verlag.
- [6] Albert Xin Jiang and Kevin Leyton-Brown, 'A general framework for computing optimal correlated equilibria in compact games', *CoRR*, **abs/1109.6064**, (2011).
- [7] Narendra Jussien, Romuald Debruyne, and Patrice Boizumault, 'Maintaining arc-consistency within dynamic backtracking', in *Proceedings of CP'00*, pp. 249–261. Springer Berlin Heidelberg, (2000).
- [8] Michael J. Kearns, Michael L. Littman, and Satinder P. Singh, 'Graphical models for game theory', in *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI'01, pp. 253–260, San Francisco, CA, USA, (2001). Morgan Kaufmann Publishers Inc.
- [9] Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Series, 1997.
- [10] Rajiv T. Maheswaran, Jonathan P. Pearce, Emma Bowring, Pradeep Varakantham, and Milind Tambe, 'Privacy loss in distributed constraint reasoning: A quantitative framework for analysis and its applications', *Autonomous Agents and Multi-Agent Systems*, **13**(1), 27–60, (2006).
- [11] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, *Algorithmic Game Theory*, Cambridge University Press, New York, NY, USA, 2007.
- [12] Luis E Ortiz and Michael Kearns, 'Nash propagation for loopy graphical games', in *Advances in Neural Information Processing Systems 15*, eds., S. Becker, S. Thrun, and K. Obermayer, 817–824, MIT Press, (2003).
- [13] Christos H. Papadimitriou and Tim Roughgarden, 'Computing correlated equilibria in multi-player games', *Journal of the ACM (JACM)*, **55**(3), 14:1–14:29, (August 2008).
- [14] Pascal Van Hentenryck and Jean-Philippe Carillon, 'Generality vs. specificity: an experience with ai and or techniques', in *Proceedings of AAAI'88*, pp. 660–664, (1988).
- [15] David Vickrey and Daphne Koller, 'Multi-agent algorithms for solving graphical games', in *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pp. 345–351, (2002).
- [16] Mohamed Wahbi and Kenneth N. Brown, 'Global constraints in distributed csp: Concurrent gac and explanations in abt', in *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming*, CP'2014, pp. 721–737, Lyon, France, (2014). Springer International Publishing.
- [17] Mohamed Wahbi, Redouane Ezzahir, Christian Bessiere, and El Housseine Bouyakhf, 'DisChoco 2: A Platform for Distributed Constraint Reasoning', in *Proceedings of workshop on DCR'11*, pp. 112–121, (2011).
- [18] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara, 'Distributed constraint satisfaction for formalizing distributed problem solving', in *Proceedings of ICDCS*, pp. 614–621, (1992).
- [19] Roie Zivan and Amnon Meisels, 'Message delay and DisCSP search algorithms', *AMAI*, **46**(4), 415–439, (2006).

Summary Information for Reasoning About Hierarchical Plans

Lavindra de Silva¹ and Sebastian Sardina² and Lin Padgham²

Abstract.

Hierarchically structured agent plans are important for efficient planning and acting, and they also serve (among other things) to produce “richer” classical plans, composed not just of a sequence of primitive actions, but also “abstract” ones representing the supplied hierarchies. A crucial step for this and other approaches is deriving precondition and effect “summaries” from a given plan hierarchy. This paper provides mechanisms to do this for more pragmatic and conventional hierarchies than in the past. To this end, we formally define the notion of a precondition and an effect for a hierarchical plan; we present data structures and algorithms for automatically deriving this information; and we analyse the properties of the presented algorithms. We conclude the paper by detailing how our algorithms may be used together with a classical planner in order to obtain abstract plans.

INTRODUCTION

This paper provides effective techniques for automatically extracting abstract actions and plans from a supplied hierarchical agent plan-library. Hierarchically structured agent plans are appealing for efficient acting and planning as they embody an expert’s domain control knowledge, which can greatly speed up the reasoning process and cater for non-functional requirements. Two popular approaches based on such representations are Hierarchical Task Network (HTN) [7, 10] planning and Belief-Desire-Intention (BDI) [18] agent-oriented programming. While HTN planners “look ahead” over a supplied collection of hierarchical plans to determine whether a task has a viable decomposition, BDI agents interleave this process with acting in the real world, thereby trading off solution quality for faster responsiveness to environmental changes. Despite these differences, HTN and BDI systems are closely related in both syntax and semantics, making it possible to translate between the two representations [19, 20].

While HTN planning and BDI execution are concerned with decomposing hierarchical structures (offline and online, respectively), one may perform other kinds of reasoning with them that do not necessarily require or amount to decompositions. For example, [5] and [21, 22] perform reasoning to coordinate the execution of abstract steps, so as to preempt potential negative interactions or exploit positive ones.

In [6], the authors propose a novel application of such hierarchies to produce “richer” classical plans composed not just of sequences of primitive actions, but also of “abstract” steps. Such abstract plans are particularly appealing in the context of BDI and HTN systems because they respect and re-use the domain control knowledge inherent in such systems, and they provide flexibility and robustness: if a refinement of one abstract step happens to fail, another option may be tried to achieve the step.

A pre-requisite for these kinds of reasoning is the availability of meaningful preconditions and effects for abstract steps (i.e., compound tasks in HTN systems or event-goals in BDI languages). Generally, this information is not supplied explicitly, but embedded within the decompositions of an abstract step. This paper provides techniques for extracting this information automatically. The algorithms we develop are built upon those of [5] and [21, 22], which calculate offline the precondition and effect “summaries” of HTN-like hierarchical structures that define the agents in a multi-agent system, and use these summaries at runtime to coordinate the agents or their plans. The most important difference between these existing techniques and ours is that the former are framed in a propositional language, whereas ours allow for first-order variables. This is fundamental when it comes to practical applicability, as any realistic BDI agent program will make use of variables. A nuance worth mentioning between our work and that of Clement et al. is that the preconditions we synthesise are standard classical precondition formulas (with disjunction), whereas their preconditions are (essentially) two sets of literals: the ones that must hold at the start of *any* successful execution of the entity, and the ones that must hold at the start of *at least one* such execution. Yao et al. [25] extend the above two strands of work to allow for concurrent steps within an agent’s plan, though still not first-order variables.

Perhaps the only work that computes summaries (“external conditions”) of hierarchies specifying first-order variables is [23]. The authors automatically extract a weaker form of summary information (what we call “mentioned” literals) to inform the task selection strategy of the UMCP HTN planner: tasks that can possibly establish or threaten the applicability of other tasks are explored first. They show that even weak summary information can significantly reduce backtracking and increase planning speed. However, the authors only provide insights into their algorithms for computing summaries.

We note that we are only concerned here with how to extract abstract actions (with corresponding preconditions and effects), and eventually abstract plans, from a hierarchical

¹ Institute for Advanced Manufacturing, University of Nottingham, Nottingham, UK, e-mail: lavindra.desilva@nottingham.ac.uk

² RMIT University, Melbourne, Australia, e-mail: {ssardina, linpa}@cs.rmit.edu.au

know-how structure. Consequently, unlike existing useful and interesting work [11, 8, 2, 1], our approach does not directly involve *guiding a planner* toward finding a suitable primitive plan. We also do not aim to build new “macro” actions from sample primitive solution plans, as done in [3], for example.

Thus, the contributions of this paper are as follows. First, we develop formal definitions for the notions of a precondition and an effect of an (abstract) event-goal. Second, we develop algorithms and data structures for deriving precondition and effect summaries from an event-goal’s hierarchy. Unlike past work, we use a typical BDI agent programming language framework; in doing so, we allow for variables in agent programs—an important requirement in practical systems. Our chosen BDI agent programming language cleanly incorporates the syntax and semantics of HTN planning as a built-in feature, making our work immediately accessible to both communities. Finally, we show how derived event-goal summaries may be used together with a classical planner in order to obtain abstract plans (which can later be further refined, if desired, to meet certain properties [6]).

THE HIERARCHICAL FRAMEWORK

Our definitions, algorithms, and results are based on the formal machinery provided by the CANPlan [20] language and operational semantics. While designed to capture the essence of BDI agent-oriented languages, it directly relates to other hierarchical representations of procedural knowledge, such as HTN planning [7, 10], both in syntax and semantics.

A CANPlan BDI agent is created by the specification of a *belief base* \mathcal{B} , i.e., a set of ground atoms, a *plan-library* Π , i.e., a set of plan-rules, and an *action-library* Λ , i.e., a set of action-rules. A plan-rule is of the form $e(\mathbf{v}) : \psi \leftarrow P$, where $e(\mathbf{v})$ is an *event-goal*, \mathbf{v} is a vector of distinct variables, ψ is the *context condition*, and P is a *plan-body* or *program*.³ The latter is made up of the following components: primitive actions (*act*) that the agent can execute directly; operations to add (+*b*) and remove (−*b*) beliefs; tests for conditions (? ϕ); and event-goal programs (!*e*), which are simply event-goals combined with the label “!”. These components are composed using the sequencing construct $P_1; P_2$. While the original definition of a plan-rule also included declarative goals and the ability to specify partially ordered programs [24], we leave out these constructs here and focus only on an AgentSpeak-like [17], typical BDI agent programming language.

There are also additional constructs used by CANPlan internally when attaching semantics to constructs. These are the programs *nil*, $P_1 \triangleright P_2$, and $(\psi_1 : P_1, \dots, \psi_n : P_n)$. Intuitively, *nil* is the empty program, which indicates that there is nothing left to execute; program $(\psi_1 : P_1, \dots, \psi_n : P_n)$ represents the plan-rules that are relevant for some event-goal; and program $P_1 \triangleright P_2$ realises failure recovery: program P_1 should be tried first, failing which P_2 should be tried. The complete language of CAN, then, is described by the grammar

$$P ::= \text{nil} \mid \text{act} \mid ?\phi \mid +b \mid -b \mid !e \mid P_1; P_2 \mid P_1 \triangleright P_2 \mid (\psi_1 : P_1, \dots, \psi_n : P_n).$$

The behaviour of a CANPlan agent is defined by a set of derivation rules in the style of Plotkin’s structural single-step

³ In [20] an event-goal is of the form $e(\mathbf{t})$ where \mathbf{t} is a vector of terms. Here, we replace \mathbf{t} with \mathbf{v} and assume WLOG that $\forall t_i \in \mathbf{t}, \psi \supset (t_i = v_i)$, where $v_i \in \mathbf{v}$.

operational semantics [15]. The *transition relation* on a configuration is defined using one or more derivation rules. Derivation rules have an *antecedent* and a *conclusion*: the antecedent can either be empty, or it can have transitions and auxiliary conditions; the conclusion is a single transition. A *transition* $C \longrightarrow C'$ within a rule denotes that configuration C yields configuration C' in a single execution step, where a configuration is the tuple $\langle \mathcal{B}, \mathcal{A}, P \rangle$ composed of a belief base \mathcal{B} , a program P , and the sequence \mathcal{A} of actions executed so far. Construct $C \xrightarrow{\mathbf{t}} C'$ denotes a transition of type \mathbf{t} , where $\mathbf{t} \in \{\text{bdi}, \text{plan}\}$; when no label is specified on a transition both types apply. Intuitively, *bdi*-type transitions are used for the standard BDI execution cycle, and *plan*-type transitions for (internal) deliberation steps within a *planning* context. By distinguishing between these two types of transitions, certain rules can be disallowed from being used in a planning context, such as those dealing with BDI-style failure handling.

We shall describe three of the CANPlan derivation rules. The rule below states that a configuration $\langle \mathcal{B}, \mathcal{A}, !e \rangle$ evolves into a configuration $\langle \mathcal{B}, \mathcal{A}, (\Delta) \rangle$ (with no changes to \mathcal{B} and \mathcal{A}) in one *bdi*- or *plan*-type execution step, with (Δ) being the set of all relevant plan-rules for e , i.e., the ones whose handling event-goal unifies with e ; *mgu* stands for “most general unifier” [12]. From (Δ) , an applicable plan-rule—one whose context condition holds in \mathcal{B} —is selected by another derivation rule and the associated plan-body scheduled for execution.

$$\frac{\Delta = \{\psi_i \theta : P_i \theta \mid e' : \psi_i \leftarrow P_i \in \Pi \wedge \theta = \text{mgu}(e, e')\}}{\langle \mathcal{B}, \mathcal{A}, !e \rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, (\Delta) \rangle}$$

The *Plan* construct incorporates HTN planning as a built-in feature of the semantics. The main rule defining the construct states that a configuration $\langle \mathcal{B}, \mathcal{A}, \text{Plan}(P) \rangle$ evolves into a configuration $\langle \mathcal{B}', \mathcal{A}', \text{Plan}(P') \rangle$ in one *bdi*-type execution step if the following two conditions hold: (i) configuration $\langle \mathcal{B}, \mathcal{A}, P \rangle$ yields configuration $\langle \mathcal{B}', \mathcal{A}', P' \rangle$ in one *plan*-type execution step, and (ii) it is possible to reach a *final* configuration $\langle \mathcal{B}', \mathcal{A}', \text{nil} \rangle$ from $\langle \mathcal{B}', \mathcal{A}', P' \rangle$ in a finite number of *plan*-type execution steps. Thus, executing the single *bdi*-type step necessitates zero or more internal “look ahead” steps that check for a successful HTN execution of P .

Unlike plan-rules, any given action program will have exactly one associated action-rule in the action-library Λ . Like a STRIPS operator, an action-rule $\text{act} : \psi \leftarrow \Phi^+; \Phi^-$ is such that *act* is a symbol followed by a vector of distinct variables, and all variables free in ψ , Φ^+ (the add list) and Φ^- (the delete list) are also free in *act*. We additionally expect any action-rule $\text{act} : \psi \leftarrow \Phi^-; \Phi^+$ to be *coherent*: that is, for all ground instances $\text{act}\theta$ of *act*, if $\psi\theta$ is consistent, then $\Phi^+\theta \cup \{-b \mid b \in \Phi^-\theta\}$ is consistent. For example, while the rule R corresponding to an action *move*(X, Y) with precondition $\text{at}(X) \wedge \neg \text{at}(Y)$ (or $X \neq Y$) and postcondition $\neg \text{at}(X) \wedge \text{at}(Y)$ is coherent, the same rule with precondition *true* is not, as there will then be a ground instance of R such that its precondition is consistent but its postcondition is not: both its add and delete lists contain the same atom.

ASSUMPTIONS

We shall now introduce some of the definitions used in the rest of the paper and concretise the rest of our assumptions. As usual, we use \mathbf{x} and \mathbf{y} to denote vectors of distinct variables,

and \mathbf{t} to denote a vector of (not necessarily distinct) terms. Moreover, since the language of CANPlan allows variables in programs, we shall frequently make use of the standard notions related to *substitutions* [12].

We assume that the plan-library does not have recursion. Formally, we assume that a *ranking* exists for the plan-library, i.e., that it is always possible to give a child a smaller rank (number) than its parent. We define a ranking as follows.

Definition 1 (RANKING) A *ranking* for a plan-library Π is a function $\mathcal{R}_\Pi : E_\Pi \mapsto \mathbb{N}_0$ from event-goal types mentioned in Π to natural numbers, such that for all event-goals $e_1, e_2 \in E_\Pi$ where e_2 is the same type as some $e_3 \in \text{children}(e_1, \Pi)$, we have that $\mathcal{R}_\Pi(e_1) > \mathcal{R}_\Pi(e_2)$.⁴ ■

In addition, we define the following two related notions: first, given an event-goal type e , $\mathcal{R}_\Pi(e)$ denotes the *rank* of e in Π ; and second, given any event-goal $e(\mathbf{t})$ mentioned in Π , we define $\mathcal{R}_\Pi(e(\mathbf{t})) = \mathcal{R}_\Pi(e(\mathbf{x}))$ (where $|\mathbf{x}| = |\mathbf{t}|$), i.e., the rank of an event-goal is equivalent to the rank of its type. In order that these and other definitions also apply to event-goal *programs*, we sometimes blur the distinction between event-goals e and event-goal programs $!e$.

Finally, we assume that context conditions are written with appropriate care. Specifically, if there is no environmental interference, whenever a plan-rule is applicable it should be possible to successfully execute the associated plan-body without any failure and recovery; this disallows rules such as $e : \text{true} \leftarrow ?\text{false}$. Our definition makes use of the notion of a *projection*: given any configuration $\langle \mathcal{B}, \mathcal{A}, P \rangle$, we define the *projection* of the first component of the tuple as $C|_{\mathcal{B}}$, the second as $C|_{\mathcal{A}}$, and the third as $C|_P$.

Definition 2 (COHERENT LIBRARY) A plan-library Π is *coherent* if for all rules $e : \psi \leftarrow P \in \Pi$, ground instances $e\theta$ of e , and belief bases \mathcal{B} , whenever $\mathcal{B} \models \psi\theta\theta'$ (where $\psi\theta\theta'$ is ground) there is a successful HTN execution $C_1 \dots C_n$ of $P\theta\theta'$ (relative to Π) with $C_1|_{\mathcal{B}} = \mathcal{B}$. A *successful HTN execution* of a program P relative to a plan-library is any finite sequence of configurations $C_1 \dots C_n$ such that $C_1|_P = P$, $C_n|_P = \text{nil}$, and for all $0 < i < n$, $C_i \xrightarrow{\text{plan}} C_{i+1}$. ■

Intuitively, the term *HTN execution* simply denotes a BDI execution in which certain BDI-specific derivation rules associated with failure and recovery have not been used.

SUMMARY INFORMATION

We can now start to define what we mean by preconditions and postconditions/effects of event-goals; some of these definitions are also used later in the algorithms. As a first step we define these notions for our most basic programs.

A basic program is either an *atomic program* or a *primitive program*. Formally, a program P is an *atomic program* (or simply *atomic*) if $P = !e \mid \text{act} \mid +b \mid -b \mid ?\phi$, and P is a *primitive program* if P is an atomic program that is not an event-goal program. Then, like the postcondition of a STRIPS

action, the *postcondition* of a primitive program is simply the atoms that will be added to and removed from the belief base upon executing the program. Formally, the *postcondition* of a primitive program P relative to an action-library Λ , denoted $\text{post}(P, \Lambda)$, is the set of literals $\text{post}(P, \Lambda) =$

$$\begin{cases} \emptyset & \text{if } P = ?\phi, \\ \{b\} & \text{if } P = +b, \\ \{-b\} & \text{if } P = -b, \\ \Phi^+ \theta \cup \{-b \mid b \in \Phi^- \theta\} & \text{if } P = \text{act} \text{ and there exists} \\ & \text{an } \text{act}' : \psi \leftarrow \Phi^+; \Phi^- \in \Lambda \\ & \text{such that } \text{act} = \text{act}'\theta. \end{cases}$$

The postcondition of a test condition is the empty set because executing a test condition does not result in an update to the belief base. The postcondition of an action program is the combination of the add list and delete list of the associated action-rule, after applying the appropriate substitution.

While this notion of a postcondition as applied to a primitive program is necessary for our algorithms later, we do not also need the matching notion of a *precondition* of a primitive program. Such preconditions are already accounted for in context conditions of plan-rules, by virtue of our assumption (Definition 2) that the latter are coherent. What we do require, however, is the notion of a *precondition* as applied to an event-goal. This is defined as any formula such that whenever it holds in some state there is at least one successful HTN execution of the event-goal from that state.

Definition 3 (PRECONDITION) A formula ϕ is said to be a *precondition* of an event-goal $!e$ (relative to a plan- and an action-library) if for all ground instances $!e\theta$ of $!e$ and belief bases \mathcal{B} , whenever $\mathcal{B} \models \phi\theta$, there exists a successful HTN execution $C_1 \dots C_n$ of $!e\theta$, where $C_1|_{\mathcal{B}} = \mathcal{B}$. ■

Unlike the postcondition of a primitive program, the postcondition of an event-goal program—and indeed any arbitrary program P —is non-deterministic: it depends on what plan-rules are chosen to decompose P . There are, nonetheless, certain effects that will be brought about irrespective of such choices. We call these *must literals*: literals that hold at the end of *every* successful HTN execution of P .

Definition 4 (MUST LITERAL) Let P be a program and l a literal where its variables are free in P . Then, l is a *must literal* of P (relative to a plan- and action-library) if for any ground instance $P\theta$ of P and successful HTN execution $C_1 \dots C_n$ of $P\theta$, we have that $C_n|_{\mathcal{B}} \models l\theta$. ■

A desirable consequence of the two definitions above is that any given set of must literals of an event-goal, like the postcondition of an action, is consistent whenever the event-goal's precondition is consistent.

Theorem 1 Let e be an event-goal, ϕ a precondition of e (relative to a plan-library Π and an action-library Λ), and L^{mt} a set of must literals of e (relative to Π and Λ). Then, for all ground instances $e\theta$ of e , if $\phi\theta\theta'$ is consistent for some ground substitution θ' , then so is $L^{mt}\theta$.

PROOF SKETCH. We prove this by contradiction. First, note that since $L^{mt}\theta$ is a set of ground literals (by Definition 4 and because $e\theta$ is ground), if $L^{mt}\theta$ is consistent, then for all

⁴ We define the function $\text{children}(\hat{e}, \Pi) = \{e \mid e' : \psi \leftarrow P \in \Pi, \hat{e} \text{ and } e' \text{ are the same type, } P \text{ mentions } !e\}$, where two event-goals are the *same type* if they have the same predicate symbol and arity. The *type* of an event-goal $e(\mathbf{t})$ is defined as $e(\mathbf{x})$, where $|\mathbf{x}| = |\mathbf{t}|$.

literals $l, l' \in L^{mt}$ it is the case that $l\theta \neq \bar{l}\theta$.⁵ If we assume that the theorem does not hold, then there must be a ground instance $e\theta$ of e such that $\phi\theta\theta'$ is consistent for some ground substitution θ' , but $l\theta = \bar{l}\theta$ for some $l, l' \in L^{mt}$.

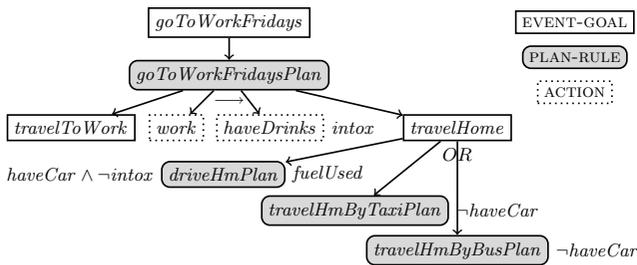
Since $\phi\theta\theta'$ is consistent, it is not difficult to show there is a belief base \mathcal{B} such that $\mathcal{B} \models \phi\theta\theta'$. Then, by Definition 3 there is also a successful HTN execution $C_1 \dots C_n$ of $e\theta$ with $C_1|_{\mathcal{B}} = \mathcal{B}$. Moreover, since l, l' are must literals of e and $e\theta$ is ground, by Definition 4 we know that (i) $l\theta$ and $l'\theta$ are also ground, and (ii) $C_n|_{\mathcal{B}} \models l\theta$ and $C_n|_{\mathcal{B}} \models l'\theta$. This leads to a contradiction of our assumption that $l\theta = \bar{l}\theta$. \square

In addition to must literals, there are two related notions. The first, called *may summary conditions* in [5], defines literals that hold at the end of at least one successful HTN execution of the program, and the second, weaker notion defines literals that are simply mentioned in the program or in one of its “descendant” programs; such literals may or may not be brought about when the program executes. It is this second notion, called *mentioned literals*, that we use.

Definition 5 (MENTIONED LITERAL) If P is a program, its *mentioned literals* (relative to a plan-library Π and an action-library Λ), denoted $mnt(P)$, is the set $mnt(P) =$

$$\left\{ \begin{array}{ll} post(P, \Lambda) & \text{if } P = +b \mid -b \mid act \mid ?\phi, \\ mnt(P_1) \cup mnt(P_2) & \text{if } P = P_1; P_2, \\ \{l\theta' \mid e' : \psi \leftarrow P' \in \Pi, \\ e = e'\theta', l \in mnt(P'), \\ \theta' \text{ is any substitution}\} & \text{if } P = !e. \end{array} \right.$$

We use this weaker notion because the stronger notion of a may summary condition in [5] is not suitable for our approach, which reasons about plans that will not be interleaved with one another—i.e., plans that will be scheduled as a sequence. For example, consider the figure below, which shows a plan-library for going to work on Fridays, possibly one belonging to a larger library from an agent-based simulation. The expressions to the left and right sides of actions/plan-rules are their preconditions and postconditions, respectively.



Observe that *fuelUsed* is actually never asserted in the context of the hierarchy shown, because literal $\neg intox$ (“not intoxicated”) in the context condition of *driveHmPlan* is contradicted by literal *intox*. However, the algorithms in [5] will still classify *fuelUsed* as a may summary condition of plan *goToWorkFridaysPlan*, because some other plan may have a step asserting $\neg intox$ —perhaps a step that involves staying

⁵ The *complement* of a literal $l \in \{a, \neg a\}$, denoted by \bar{l} , is a if $l = \neg a$, and $\neg a$ otherwise.

overnight in a hotel nearby—that can be ordered to occur between *haveDrinks* and *travelHome*.

Since we cannot rely on such steps, we settle for a weaker notion—mentioned literals—than the corresponding definition of a may summary condition. By our definition there can be literals that are mentioned in some plan-body but in fact can never be asserted, because of interactions that preclude the particular plan-body which asserts that literal from being applied. We avoid the approach of disallowing interactions like the one shown above in order to use the stronger notion of a may summary condition because such interactions are natural in BDI and HTN domains: event-goals such as *travelHome* are, intuitively, meant to be self-contained “modules” that can be “plugged” into any relevant part of a hierarchical structure in order to derive all or just some of their capabilities.

Finally, we conclude this section by combining the above definitions of must and mentioned literals to form the definition of the *summary information* of a program.

Definition 6 (SUMMARY INFORMATION) If P is a program, its *summary information* (relative to a plan-library and an action-library) is a tuple $\langle P, \phi, L^{mt}, L^{mn} \rangle$, where ϕ is a precondition of P if P is an event-goal program, and $\phi = \epsilon$ otherwise; L^{mt} is a set of must literals of P ; and L^{mn} is a set of mentioned literals of P . \blacksquare

EXTRACTING SUMMARY INFORMATION

With the formal definitions now in place, in this section we provide algorithms to extract summary information for event-goals in a plan-library. Moreover, we illustrate the algorithms with an example, and analyse their properties.

Basically, we extract summary information from a given plan-library and action-library by propagating up the summary information of lower-level programs, starting from the leaf-level ones in the plan-library, until we eventually obtain the summary information of all the top-level event-goals.

To be able to identify must literals, we need to be able to determine whether a given literal is definitely undone, or *must undone*, and possibly undone, or *may undone* in a program. Informally, a literal l is *must undone* in a sequence P of atomic programs if the literal’s negation is a must literal of some atomic program in P . Formally, then, given a program P and the set Δ of summary information of all atomic programs in P , a literal l is *must undone* in P relative to Δ , denoted $\text{Must-Undone}(l, P, \Delta)$, if there exists an atomic program P' in P and a literal $l' \in L^{mt}$, with $\langle P', \phi, L^{mt}, L^{mn} \rangle \in \Delta$, such that $l = \bar{l}'$, that is, l is the complement of l' .

Similarly, we can informally say that a literal l is *may undone* in a program P if there is a literal l' that is a mentioned (or must) literal of some atomic program in P such that l' may become the negation of l after variable substitutions. Formally, given a program P and the set Δ of summary information of all atomic programs in P , a literal l is *may undone* in P relative to Δ , denoted $\text{May-Undone}(l, P, \Delta)$, if there exists an atomic program P' in P , a substitution θ , and a literal $l' \in L^{mn}$,⁶ with $\langle P', \phi, L^{mt}, L^{mn} \rangle \in \Delta$, such that $l\theta = \bar{l}'\theta$.

Algorithm 1. This is the top-level algorithm for computing the summary information Δ of event-goal types occurring

⁶ variables occurring in l' are renamed to those not occurring in l

Algorithm 1 Summ(Π, Λ)**Require:** Plan-library Π and action-library Λ .**Ensure:** Set of summary info. of event-goal types in Π .

- 1: $\Delta \leftarrow \{(P, \epsilon, \text{post}(P, \Lambda), \text{post}(P, \Lambda)) \mid$
 $P \text{ is a primitive program mentioned in } \Pi\}$
- 2: $E \leftarrow \{e(\mathbf{x}) \mid e \text{ is an event-goal mentioned in } \Pi\}$
- 3: **for** $i \leftarrow \min(R)$ **to** $\max(R)$ **where**
 $R = \{\mathcal{R}_\Pi(e) \mid e \in E\}$ **do** // Recall $\mathcal{R}_\Pi(e)$ is the rank of e
- 4: **for** each $e \in E$ such that $\mathcal{R}_\Pi(e) = i$ **do**
- 5: $\Delta \leftarrow \Delta \cup$
 $\{\text{SummPlan}(P, \Pi, \Lambda, \Delta) \mid e' : \psi \leftarrow P \in \Pi, e' = e\theta\}$
- 6: $\Delta \leftarrow \Delta \cup \{\text{SummEvent}(e, \Pi, \Delta)\}$
- 7: **return** $\Delta \setminus \{u \mid u \in \Delta,$
 $u \text{ is not the summary information of an event-goal}\}$

Algorithm 2 SummPlan($P, \Pi, \Lambda, \Delta_{in}$)**Require:** Plan-body P ; plan-library Π ; action-library Λ ; and the set Δ_{in} of summary information of primitive programs and event-goal types mentioned in P .**Ensure:** The summary information of P .

- 1: $\Delta \leftarrow \Delta_{in} \cup \{(e(\mathbf{x}), \phi, L^{mt}, L^{mn})\theta \mid !e(\mathbf{t}) \text{ occurs in } P,$
 $\langle e(\mathbf{x}), \phi, L^{mt}, L^{mn} \rangle \in \Delta_{in}, e(\mathbf{t}) = e(\mathbf{x})\theta\}$
 // We assume variables in L^{mn} are appropriately renamed
- 2: Let $P = P_1; P_2; \dots; P_n$ where each P_i is atomic
- 3: $L_P^{mt} \leftarrow \{l \mid l \in L^{mt}, \langle P_i, \phi, L^{mt}, L^{mn} \rangle \in \Delta,$
 $i \in \{1, \dots, n\}, \neg \text{May-Undone}(l, P_{i+1}; \dots; P_n, \Delta)\}$
- 4: $L_P^{mn} \leftarrow \{l \mid l \in L^{mn} \cup L^{mn}, \langle P_i, \phi, L^{mt}, L^{mn} \rangle \in \Delta,$
 $i \in \{1, \dots, n\}, \neg \text{Must-Undone}(l, P_{i+1}; \dots; P_n, \Delta)\}$
- 5: **return** $\langle P, \epsilon, L_P^{mt}, L_P^{mn} \rangle$

Algorithm 3 SummEvent($e(\mathbf{x}), \Pi, \Delta$)**Require:** Event-goal type $e(\mathbf{x})$; plan-library Π ; and the set Δ of summary information of plan-bodies of plan-rules $e' : \psi \leftarrow P \in \Pi$ such that $e' = e(\mathbf{x})\theta$.**Ensure:** The summary information of $e(\mathbf{x})$.

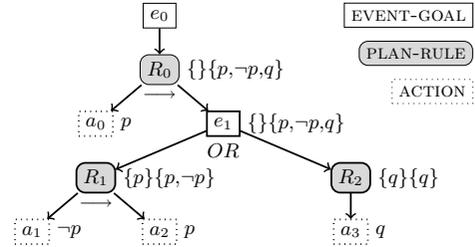
- 1: $\phi \leftarrow \text{false}, L^{mt} \leftarrow \emptyset, L^{mn} \leftarrow \emptyset,$ and $S \leftarrow \emptyset$
 // L^{mt}, L^{mn} are sets of literals and S is a set of sets of literals
- 2: **for** each $e(\mathbf{y}) : \psi \leftarrow P \in \Pi$ such that $e(\mathbf{x}) = e(\mathbf{y})\theta$ **do**
- 3: $\phi \leftarrow \phi \vee \psi\theta$
 // Relevant variables in ψ and L_P^{mt}, L_P^{mn} below are renamed
- 4: $S \leftarrow S \cup \{L_P^{mt}\theta\}$, where $\langle P, \epsilon, L_P^{mt}, L_P^{mn} \rangle \in \Delta$
- 5: $L^{mn} \leftarrow L^{mn} \cup L_P^{mn}\theta$
- 6: **if** $S \neq \emptyset$ **then** // Obtain the must literals of $e(\mathbf{x})$
- 7: $L^{mt} \leftarrow \bigcap S$
- 8: $L^{mt} \leftarrow \{l \mid l \in L^{mt},$
 variables occurring in l also occur in $e(\mathbf{x})\}$
- 9: **return** $\langle e(\mathbf{x}), \phi, L^{mt}, L^{mn} \rangle$

in the plan-library. The algorithm works bottom up, by summarising first the leaf-level entities of the plan-library—the primitive programs (line 1)—and then repetitively summarising plan-bodies (Algorithm 2) and event-goals (Algorithm 3) in increasing order of their levels of abstraction (lines 3-6).

Algorithm 2. This algorithm summarises the given plan-body P by referring to the set Δ_{in} containing the summary information tuples of programs in P . First, the algorithm obtains the summary information of each event-goal program in the plan-body from the summary information of the corresponding event-goal types in Δ_{in} (line 1). This involves substituting variables occurring in relevant summary information tuples in Δ with the corresponding terms occurring in the event-goal program being considered. Second, the algorithm computes the set of must literals (L_P^{mt}) and the set of mentioned literals (L_P^{mn}) of the given plan-body P , by determining, from the must and mentioned literals of atomic programs in P , which literals will definitely hold and which ones will

only possibly hold on successful executions of P (lines 3 and 4). More precisely, a must literal l of an atomic program P_i in $P = P_1; \dots; P_n$ is classified as a must literal of P only if l is not may undone in $P_{i+1}; \dots; P_n$ (line 3). Otherwise, l is classified as only a mentioned literal of P , provided l is not also must undone in $P_{i+1}; \dots; P_n$ (line 4). The reason we do not summarise literals that are must undone is to avoid missing must literals in cases where they are possibly undone but then later (definitely) reintroduced, as we illustrate below.

Suppose, on the contrary, that the algorithm *does* summarise mentioned literals that are must undone. Then, given the plan-library below, the algorithm would (hypothetically) compute the summary information denoted by the two sets attached to each node, the one on the left being its set of must literals and the one on the right its set of mentioned literals.



Observe that literal p asserted by a_0 is not recognised as a must literal of R_0 simply because it is may undone by mentioned literal $\neg p$ of e_1 (asserted by a_1), despite the fact that action a_2 of R_1 also subsequently adds p . On the other hand, our algorithm does recognise p as a must literal of R_0 by not including $\neg p$ in the set of mentioned literals of R_1 (line 4).

Algorithm 3. This algorithm summarises the given event-goal type $e(\mathbf{x})$ by referring to the set Δ containing the summary information tuples associated with the plan-bodies of plan-rules handling $e(\mathbf{x})$. In lines 2 and 3, the algorithm takes the precondition of the event-goal as the disjunction of the context conditions of all associated plan-rules.⁷ Then, the algorithm obtains the must and mentioned literals of the event-goal by respectively taking the intersection of the must literals of associated plan-rules (lines 4 and 7), and the union of the mentioned literals of associated plan-rules (line 5). Applying substitution θ in line 4 helps recognise must literals of $e(\mathbf{x})$, by ensuring that variables occurring in the summary information of its associated plan-bodies have consistent names with respect to $e(\mathbf{x})$.

An illustrative example

We shall illustrate the three algorithms with the example of a simple agent exploring the surface of Mars. A part of the agent's domain is depicted as a hierarchy in Figure 1. The hierarchy's top-level event-goal is to explore a given soil location Y from current location X . This is achieved by plan-rule R_0 , which involves navigating to the location and then doing a soil experiment. Navigation is achieved by rules R_1 and R_2 , which involve moving to the location, possibly after calibrating some of the rover's instruments. Doing a soil experiment involves the two sequential event-goals of getting soil results for Y and

⁷ We do not need to “propagate up” context conditions as we do with plan-bodies' summary information because higher-level context conditions account for lower-level ones due to Definition 2.

transmitting them to the lander. Specifically, the former is refined into actions such as determining moisture content and average soil particle size, and transmitting results involves either establishing a connection with the lander, sending it the results, and then terminating the connection, or if the lander is not within range, navigating to it and uploading the soil results. The table in Figure 1 shows the summary information computed by our algorithms for elements in the figure’s hierarchy. Below, we describe some of the more interesting values in the table.

Plan-body P_7 . Must literals $\neg at(Y)$ and $at(L)$ of P_7 are derived from those of $nav(X, Y)$, after renaming variables X and Y to respectively Y and L in line 1 of Algorithm 2.

Plan-body P_4 . While $hSS(Y)$ is a must literal of P_4 ’s primitive action $pickSoil(Y)$, the literal is must undone by P_4 ’s last primitive action $dropSoil(Y)$. Thus, $hSS(Y)$ is not a must (nor mentioned) literal of P_4 . On the other hand, literal $\neg hSS(Y)$ is indeed a must literal of P_4 , along with literals $hMC(Y)$ and $hPS(Y)$, both of which are derived from the summary information of event-goal $analyseSoil(Y)$.

Plan-body P_0 . While $\neg at(X)$ is a must literal of event-goal $nav(X, Y)$, it is only a mentioned literal of P_0 because it is may undone in event-goal $doSoilExp(Y)$; specifically, its mentioned literal $at(L)$ is such that $\neg at(X)\theta = \neg at(L)\theta$ for $\theta = \{X/L\}$. Similarly, must literal $at(Y)$ of $nav(X, Y)$ is also may undone in $doSoilExp(Y)$.

Event-goal $transmitRes(Y)$. Since literal $rT(Y)$ is a must literal of both of the event-goal’s associated plan-bodies P_6 and P_7 , and Y also occurs in the event-goal, the literal is classified as a must literal of the event-goal. Recall that this means that for any ground instance $transmitRes(Y)\theta$ of the event-goal, literal $rT(Y)\theta$ holds at the end of any successful HTN execution of $transmitRes(Y)\theta$.

Soundness and Completeness

We shall now analyse the properties of the algorithms presented. We show that they are sound, and we then discuss completeness. First, it is not difficult to see that the presented algorithms terminate, and that they run in polynomial time.

Theorem 2 *Algorithm 1 always terminates, and runs in polynomial time on the number of symbols occurring in $\Pi \cup \Lambda$.*

PROOF SKETCH. *Since the algorithms presented are non-recursive, the only non-trivial part of the proof concerns the procedure for computing a unification when determining whether $May\text{-}Undone(l, P, \Delta)$ holds (for a literal l , program P , and a set Δ of summary information). In [13], one such unification procedure is presented that is linear on the number of symbols occurring in the two literals to be unified. \square*

This result is important when the plan-library changes over time, e.g. because the agent learns from past experience, and summary information needs to be recomputed frequently, or when it needs to be computed right at the start of HTN planning, as done in [23].

The next result states that whenever Algorithm 1 (Summ) classifies a literal as a must literal of an event-goal, this is guaranteed to be the case, and that the algorithm correctly computes its precondition and mentioned literals. More specifically, any computed tuple, which includes one event-goal type

e , formula ϕ and must literals L^{mt} , respects Definitions 3 and 4. Moreover, there is exactly one tuple associated with e .

Theorem 3 *Let Π be a plan-library, Λ be an action-library, e be an event-goal type mentioned in Π , and let $\Delta_{out} = \text{Summ}(\Pi, \Lambda)$. There exists one tuple $\langle e, \phi, L^{mt}, L^{mn} \rangle \in \Delta_{out}$, the tuple is the summary information of e , and $L^{mn} \subseteq mnt(e)$ (recall $mnt(e)$ denotes the mentioned literals of e).*

PROOF SKETCH. *We prove this by induction on e ’s rank in Π . First, from our ranking function we obtain a new one \mathcal{R}_Π by making event-goal ranks “contiguous” and start from 0.*

For the base case, take any event-goal e with $\mathcal{R}_\Pi(e) = 0$. According to Definition 1 (Ranking), if $\mathcal{R}_\Pi(e) = 0$, then $children(e, \Pi) = \emptyset$. Thus, for all rules $e' : \psi \leftarrow P \in \Pi$ such that $e = e'\theta$, no event-goals occur in P . Let P_{all} be the set of plan-bodies of all such rules. Then, the two main steps are as follows. First, we show that due to line 1 of procedure $\text{Summ}(\Pi, \Lambda)$ there is exactly one summary tuple $\langle P', \epsilon, L_{P'}^{mt}, L_{P'}^{mn} \rangle \in \Delta$ for each primitive program P' mentioned in each $P \in P_{all}$. Second, since $\text{SummPlan}(P, \Pi, \Lambda, \Delta)$ is called in line 5 for each plan-body $P \in P_{all}$, we show that on the completion of this line, there is exactly one summary tuple $\langle P, \epsilon, L_P^{mt}, L_P^{mn} \rangle \in \Delta$ for each plan-body $P \in P_{all}$. A similar argument applies to line 6.

For the induction hypothesis, we assume that the theorem holds if $\mathcal{R}_\Pi(e) \leq k$, for some $k \in \mathbb{N}_0$.

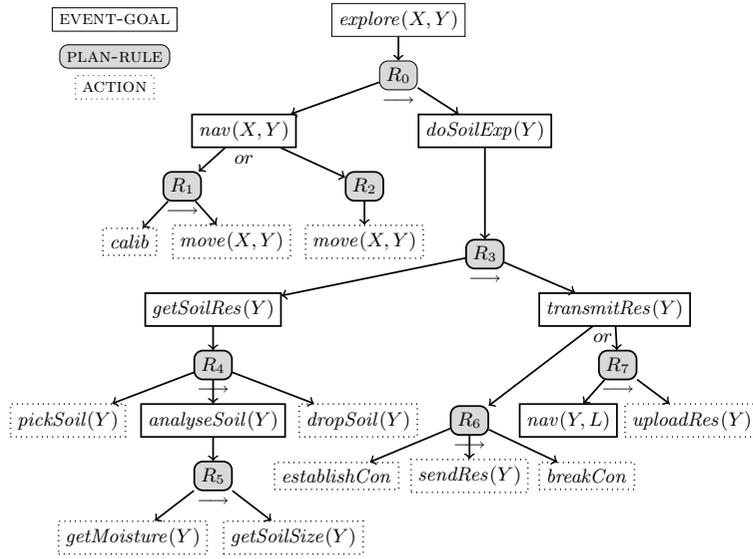
For the inductive step, we show that the theorem holds for $\mathcal{R}_\Pi(e) = k + 1$. The main steps are as follows. Let E_{all} denote the (non-empty) set of event-goal types mentioned in all plan-bodies $P \in P_{all}$, where P_{all} is as before. By Definition 1, we know that for all $e' \in E_{all}$, $\mathcal{R}_\Pi(e') < \mathcal{R}_\Pi(e)$. Then, by the induction hypothesis, it follows that for each $e' \in E_{all}$, there is exactly one tuple $\langle e', \phi_{e'}, L_{e'}^{mt}, L_{e'}^{mn} \rangle \in \Delta$, and this tuple is the summary information of e' . Finally, since e has a higher rank than those of all event-goals $e' \in E_{all}$, $\text{Summ}(\Pi, \Lambda)$ will only call $\text{SummEvent}(e, \dots)$ after the above tuples are added to Δ , resulting in tuple $\langle e, \phi, L^{mt}, L^{mn} \rangle$ also being added to Δ . \square

Next, we discuss completeness. The theorem below states that any precondition computed by Algorithm 3 is complete: i.e., given any state from where there is a successful HTN execution of an event-goal, the precondition extracted for the event-goal will hold in that state. This theorem only concerns Algorithm 3 because we can compute preconditions of event-goals without needing to compute preconditions of plans.

Theorem 4 *Let Π be a plan-library, Λ an action-library, e an event-goal type mentioned in Π , and let $\langle e, \phi, L^{mt}, L^{mn} \rangle \in \text{Summ}(\Pi, \Lambda)$. For all ground instances $!e\theta$ of e and belief bases \mathcal{B} such that there exists a successful HTN execution $C_1 \dots C_n$ of $!e\theta$ with $C_1 |_{\mathcal{B}} = \mathcal{B}$, it is the case that $\mathcal{B} \models \phi\theta$.*

PROOF SKETCH. *We prove that if $!e\theta$ has a successful HTN execution, there is also a plan-rule $e' : \psi \leftarrow P \in \Pi$ associated with e such that $\mathcal{B} \models \psi'\theta$ holds, where ψ' is an appropriate renaming of variables in ψ . We then show that ψ' is a disjunct of ϕ , from which it follows that $\mathcal{B} \models \phi\theta$. \square*

There are, however, situations where the algorithms do not detect all *must literals* of an event-goal. The underlying reason



Program	Must Literals	Mentioned Literals
<i>calib</i>	<i>cal</i>	-
<i>move(X, Y)</i>	$\neg at(X), at(Y)$	-
<i>pickSoil(Y)</i>	<i>hSS(Y)</i>	-
<i>dropSoil(Y)</i>	$\neg hSS(Y)$	-
<i>getMoisture(Y)</i>	<i>hMC(Y)</i>	-
<i>getSoilSize(Y)</i>	<i>hPS(Y)</i>	-
<i>establishCon</i>	<i>cE</i>	-
<i>sendRes(Y)</i>	<i>rT(Y)</i>	-
<i>breakCon</i>	$\neg cE$	-
<i>uploadRes(Y)</i>	<i>rT(Y)</i>	-
P_1	$\neg at(X), at(Y), cal$	-
P_2	$\neg at(X), at(Y)$	-
P_5	<i>hMC(Y), hPS(Y)</i>	-
P_4	<i>hMC(Y), hPS(Y),</i> $\neg hSS(Y)$	-
P_6	<i>rT(Y), \neg cE</i>	-
P_7	$\neg at(Y), at(L),$ <i>rT(Y)</i>	<i>cal</i>
P_3	<i>rT(Y), hMC(Y),</i> <i>hPS(Y), \neg hSS(Y)</i>	$\neg cE, \neg at(Y), at(L),$ <i>cal</i>
P_0	<i>rT(Y), hMC(Y),</i> <i>hPS(Y), \neg hSS(Y)</i>	$\neg cE, at(Y), \neg at(Y),$ <i>at(L), cal, \neg at(X)</i>
<i>nav(X, Y)</i>	$\neg at(X), at(Y)$	<i>cal</i>
<i>analyseSoil(Y)</i>	Same as P_5	-
<i>getSoilRes(Y)</i>	Same as P_4	-
<i>transmitRes(Y)</i>	<i>rT(Y)</i>	$\neg cE, \neg at(Y), at(L),$ <i>cal</i>
<i>doSoilExp(Y)</i>	Same as P_3	Same as P_3
<i>explore(X, Y)</i>	Same as P_0	Same as P_0

Figure 1: Must and mentioned literals (right) of atomic programs and plan-bodies in the hierarchy (left). The rightmost column only shows mentioned literals that are not also must literals. Abbreviations in the table are as follows: *cal* = *calibrated*, *hSS* = *haveSoilSample*, *hMC* = *haveMoistureContent*, *hPS* = *haveParticleSize*, *cE* = *connectionEstablished*, *rT* = *resultsTransmitted*, and variable L = *Lander*. Rule R_7 's context condition binds L to the lander's location. Each plan-body P_i corresponds to rule R_i in the hierarchy.

for this is that we do not reason about (FOL) precondition formulas; specifically, we do not check entailment, because this is semi-decidable in general [9]. In what follows, we use examples to characterise the four cases in which the algorithms are unable to recognise must literals, and show how some of the cases can be averted.

The first case was depicted in our example about going to work on Fridays: by Definition 4, literal $\neg haveCar$ is a must literal of *goToWorkFridaysPlan*, but Algorithm 2 classifies it as only a mentioned literal, as it cannot infer that the context condition of rule *driveHmPlan* is contradicted by literal *intox*, and therefore that *driveHmPlan* can never be applied.

The second case is where a literal is a must literal simply because it is entailed by a context condition. For example, take an event-goal *mov(P, T, L)* that is associated with one plan-rule, whose context condition checks whether package P is in truck T , i.e., *in(P, T)*, and whose plan-body moves the truck to location L . Observe that *in(P, T)* is a must literal of *mov(P, T, L)* by definition, but since *in(P, T)* does not occur in the plan-body, Algorithm 2 does not consider the literal. We do not expect this to be an issue in practice, however, because such literals are accounted for by the event-goal's (extracted) precondition.

The third case is where must literals are “hidden” due to the particular variable/constant symbols chosen by the domain writer when encoding literals. For example, given the following two plan-rules for an event-goal that sends an email from F to T , literal *sent(T)* is only a mentioned literal of *sendMail(F, T)* according to Algorithm 3 (line 7 in particu-

lar), but a must literal of it by definition:

$$\begin{aligned} sendMail(F, T) : (F \neq T) &\leftarrow +addedSignature ; +sent(T), \\ sendMail(F, T) : (F = T) &\leftarrow +sent(F). \end{aligned}$$

Nonetheless, by changing $+sent(F)$ to $+sent(T)$, which then mentions the same variable symbol as the first plan-body, *sent(T)* is identified by the algorithm as a must literal of *sendMail(F, T)*. In general, such “hidden” must literals can be disclosed by choosing terms with appropriate care.

Finally, while Algorithm 2 “conservatively” classifies any must literal that is may undone as a may literal, it could still be a must literal by definition. For example, given an event-goal *move(X, Y)*, suppose that the following plan-rule is the only one relevant for the event-goal:

$$move(X, Y) : at(X) \wedge \neg at(Y) \leftarrow \neg at(X) ; +at(Y).$$

Then, by Definition 4, both $\neg at(X)$ and $at(Y)$ are must literals of the event-goal, but only $at(Y)$ is its must literal according to Algorithm 2, because it cannot infer that the context condition entails $X \neq Y$.⁸ While the algorithm does fail to detect some must literals in such domains, this can sometimes be averted by encoding the domain differently. For example, the above rule can be encoded as an action-rule instead, in which case Algorithm 1 (in line 1) will classify $\neg at(X)$ (and $at(Y)$) as a must literal of *move(X, Y)*, under the assumption that action-rules are coherent.

⁸ Note that if the context condition is just $at(X)$, then, by definition, $at(Y)$ would indeed be the only must literal of the event-goal, because it would then be possible for X and Y to have the same value, and for $at(Y)$ to “undo” $\neg at(X)$.

AN APPLICATION TO PLANNING

One application of the algorithms presented is to create abstract planning operators that may be used together with primitive operators and a classical planner in order to obtain abstract (or “hybrid”) plans. While [6] focuses on algorithms for extracting an “ideal” abstract plan from an abstract plan that is supplied, here we give the details regarding how a first abstract plan may be obtained.

To get abstract operators Λ^a from a plan-library Π and an action-library Λ , we take the set $\Delta = \text{Summ}(\Pi, \Lambda)$ and create an (abstract) operator for every summary information tuple $\langle e, \phi, L^{mt}, L^{mn} \rangle \in \Delta$. To this end, we take the operator’s name as e , appended with its arity and combined with any additional variables occurring in ϕ ; the operator’s precondition as ϕ ; and its postcondition as the set of must literals L^{mt} .

Since mentioned literals of event-goals are not included in their associated abstract operators, it is crucial that we ascertain whether these literals will cause unavoidable conflicts in an abstract plan found. For example, consider the classical planning problem with initial state p and goal state r , and the abstract plan $e_1 \cdot e_2$ consisting of two event-goals (or abstract operators). Suppose e_1 and e_2 have the following plan-rules:

$$e_1 : \text{true} \leftarrow +p; +q \quad e_1 : \text{true} \leftarrow -p; +q \quad e_2 : p \wedge q \leftarrow +r$$

Notice that the postconditions (must literals) of abstract operators e_1 and e_2 are respectively q and r , and that $e_1 \cdot e_2$ is a classical planning solution for the given planning problem. However, when this plan is executed, if e_1 is decomposed using its second plan-rule, this will cause (mentioned literal) $\neg p$ to be brought about, thereby invalidating the context condition of e_2 (which requires p).

To check for such cases, we present the following simple polynomial-time algorithm. Suppose that $P = P_1; \dots; P_n$ is the program corresponding to a classical planning solution $P'_1 \cdot \dots \cdot P'_n$ for some planning problem, where each (ground) P_i is either an action or event-goal. Then, we say that P is *correct* relative to Δ if for any (ground) literal l occurring in the precondition of any P_i , the following condition holds: if l is not must undone and it is may undone (relative to Δ) in the preceding subplan $P_1; \dots; P_{i-1}$ by some mentioned literal l' of a step P_k in the subplan,⁹ then literal l , or its complement, is also must undone (relative to Δ) in the steps $P_{k+1}; \dots; P_{i-1}$. Otherwise, P is said to be *potentially incorrect*. Interestingly, the situation where l is must undone in $P_1; \dots; P_{i-1}$ is not unacceptable because it cannot invalidate the (possibly disjunctive) precondition of P_i , given that $P_1; \dots; P_n$ corresponds to a solution for some classical planning problem. The following theorem states that, as expected, a correct program P will always have at least one successful HTN decomposition.

Theorem 5 *Let Π be a plan-library, Λ an action-library, $\Delta = \text{Summ}(\Pi, \Lambda)$, and P the program corresponding to a solution for the classical planning problem $\langle \mathcal{B}, \mathcal{B}_g, A \cup \Lambda^a \rangle$, where \mathcal{B} and \mathcal{B}_g are belief bases representing respectively initial and goal states. Then, if P is correct (relative to Δ), there*

is a successful HTN execution $C_1 \cdot \dots \cdot C_n$ of P such that $C_1|_{\mathcal{B}} = \mathcal{B}, C_1|_P = P$ and $C_n|_{\mathcal{B}} \models \mathcal{B}_g$.

PROOF SKETCH. *If there is no such successful execution, since $P = P_1; \dots; P_n$ is a classical planning solution, there must be a mentioned literal of some P_i that intuitively “conflicts” with a literal occurring in the precondition of some P_j , with $j > i$. The classical planner will not have taken such conflicts into account, but according to the definition of what it means for P to be correct, such a mentioned literal cannot exist. \square*

If we find that P is potentially incorrect, we then determine whether it is *definitely incorrect*, i.e., whether there are conflicts that are unavoidable. To this end, we look for a successful HTN decomposition of P , failing which the plan is discarded and the process repeated with a new abstract plan.

DISCUSSION & FUTURE WORK

We have presented definitions and sound algorithms for summarising plan hierarchies which, unlike past work, are defined in a typical and well understood BDI agent-oriented programming language. By virtue of its syntax and semantics being inherently tied to HTN planning, our work straightforwardly applies to HTN planners such as SHOP [14]. Our approach is closely related to [5], the main differences being that we support variables in agent programs, and we reason about non-concurrent plans. While these do make a part of our approach incomplete, we have shown how this can sometimes be averted by writing domains with appropriate care. Crucially, we have handled variables “natively”, without grounding them on a finite set of constants. We concluded with one application of our algorithms, showing how they can be used together with a classical planner in order to obtain abstract plans.

We expect that the summaries we compute will be useful in other applications that rely on similar information, such as coordinating the plans of single [21, 22] and multiple [5] agents, and particularly in improving HTN planning efficiency [23]. There is also potential for using such information as guidance when creating agent plans manually [25].

Interestingly, the application we presented mitigates our restriction that plan-libraries cannot be recursive, as the classical planner can, if necessary, repeat an event-goal in an abstract plan. Nonetheless, allowing recursive plan-libraries is still an interesting avenue for future work. Another useful improvement would be to allow partially ordered steps in plan-bodies (i.e., the construct $P \parallel P'$). Given a plan-library Π , one potential approach to that end is to obtain the plan-library Π' consisting of all linear extensions of plan-rules in Π , and then use Π' as the input into Algorithm 1 (Summ). We could use existing, fast algorithms to generate linear extensions [16], or consider simpler plan-rules corresponding to restricted classes of partially ordered sets [4]. Finally, it would be interesting to formally characterise the restricted class of domains in which the presented algorithms are complete.

ACKNOWLEDGEMENTS

This work was supported by Agent Oriented Software and the Australian Research Council (grant LP0882234). We thank Brian Logan for useful discussions relating to the work presented, and the anonymous reviewers for helpful feedback.

⁹ We rely here on a slightly extended version of the definition of may undone from before, to have the exact step (P_k) and literal (l') responsible for the “undoing”. Moreover, observe that literals l and l' are obtained by applying the same substitution that the planner applied to obtain P'_i and P'_k , respectively.

REFERENCES

- [1] R. W. Alford, U. Kuter, and D. Nau. Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1629–1634, 2009.
- [2] J. A. Baier, C. Fritz, and S. A. McIlraith. Exploiting procedural domain control knowledge in state-of-the-art planners. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-07)*, pages 26–33, 2007.
- [3] A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer. Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research (JAIR)*, 24:581–621, 2005.
- [4] V. Bouchitte and M. Habib. The calculation of invariants for ordered sets. In I. Rival, editor, *Algorithms and Order*, volume 255, pages 231–279. Springer Netherlands, 1989.
- [5] B. J. Clement, E. H. Durfee, and A. C. Barrett. Abstract reasoning for planning and coordination. *Journal of Artificial Intelligence Research (JAIR)*, 28:453–515, 2007.
- [6] L. de Silva, S. Sardina, and L. Padgham. First Principles Planning in BDI systems. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-09)*, pages 1105–1112, 2009.
- [7] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-94)*, pages 1123–1128, 1994.
- [8] C. Fritz, J. A. Baier, and S. A. McIlraith. ConGolog, Sin Trans: Compiling ConGolog into Basic Action Theories for planning and beyond. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-08)*, pages 600–610, 2008.
- [9] D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors. *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, 1994.
- [10] M. Ghallab, D. S. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers Inc., 2004.
- [11] S. Kambhampati, A. D. Mali, and B. Srivastava. Hybrid planning for partially hierarchical domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-98)*, pages 882–888, 1998.
- [12] J. W. Lloyd. *Foundations of Logic Programming; (2nd Extended Ed.)*. Springer-Verlag New York, Inc., 1987.
- [13] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.
- [14] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research (JAIR)*, 20:379–404, 2003.
- [15] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, University of Aarhus, Denmark, 1981.
- [16] G. Pruesse and F. Ruskey. Generating linear extensions fast. *SIAM Journal on Computing*, 23(2):373–386, 1994.
- [17] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the European workshop on Modelling Autonomous Agents in a Multi-Agent World : agents breaking away (MAAMAW-96)*, pages 42–55. Springer, 1996.
- [18] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the International Conference on Multiagent Systems (ICMAS-95)*, pages 312–319, 1995.
- [19] S. Sardina, L. de Silva, and L. Padgham. Hierarchical planning in BDI agent programming languages: A formal approach. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-06)*, pages 1001–1008, 2006.
- [20] S. Sardina and L. Padgham. A BDI agent programming language with failure handling, declarative goals, and planning. *Autonomous Agents and Multiagent Systems*, 23(1):18–70, 2011.
- [21] J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and avoiding interference between goals in intelligent agents. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 721–726, 2003.
- [22] J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and exploiting positive goal interaction in intelligent agents. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pages 401–408, 2003.
- [23] R. Tsuneto, J. Hendler, and D. Nau. Analyzing external conditions to improve the efficiency of HTN planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-98)*, pages 913–920, 1998.
- [24] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative and procedural goals in intelligent agent systems. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-02)*, pages 470–481, 2002.
- [25] Y. Yao, L. de Silva, and B. Logan. Reasoning about the executability of goal-plan trees. In *Engineering Multi-Agent Systems Workshop (EMAS-16)*, pages 181–196, 2016.

Knowledge-Based Programs with Defaults in a Modal Situation Calculus

Jens Claßen¹ and Malte Neuss²

Abstract. We consider the realistic case of a GOLOG agent that only possesses incomplete knowledge about the state of its environment and has to resort to sensing in order to gather additional information at runtime, and where the agent is controlled by a knowledge-based program in which test conditions explicitly refer to the agent’s knowledge (or lack thereof). In this paper, we propose a formalization of knowledge-based agents that extends earlier proposals by a form of non-monotonic reasoning that includes Reiter-style defaults. We present a reasoning mechanism that enables us to reduce projection queries about future states of the agent’s knowledge (including nesting of epistemic modalities) to classical Default Logic, and provide a corresponding Representation Theorem. We thus obtain the theoretical foundation for an implementation where reasoning sub-tasks can be handed to an embedded off-the-shelf reasoner for Default Logic, and that supports a (in some respects) more expressive epistemic action language than previous solutions.

1 INTRODUCTION

The GOLOG [19, 4] family of action languages and the underlying Situation Calculus action formalism [25, 32] are a popular means for the high-level control of autonomous agents. Reiter [33] proposed an extension for realistic scenarios where the agent possesses only incomplete information about its surroundings and has to resort to runtime sensing to fulfill its task. In what is called a *knowledge-based program*, test conditions may then explicitly refer to the agent’s epistemic state, thus enabling it to reason about its own knowledge (or lack thereof). Based on Scherl and Levesque’s [36] epistemic extension of the Situation Calculus, he furthermore provided a regression-based reasoning procedure where deciding subjective queries about future situations is reduced to standard theorem proving.

Nevertheless, Reiter’s approach came with some limitations. On the one hand, the notion of *only knowing* [17] – the fact that the agent’s knowledge base represents *all and only* what the agent knows – is only formalized in a meta-theoretic manner, which makes theoretical considerations difficult. More importantly, despite the ability to explicitly refer to the agent’s knowledge, he considers a very limited set of queries that disallows referring to future situations, nesting of modal knowledge operators (needed for introspection) and quantifying-in (necessary to distinguish “knowing that” from “knowing what”). Consider the well-known Wumpus domain [34], where the agent acts in a grid world whose cells (called *rooms*) may be occupied by pits (e.g. squares $\langle 3, 1 \rangle$ and $\langle 4, 4 \rangle$ in Figure 1) or the

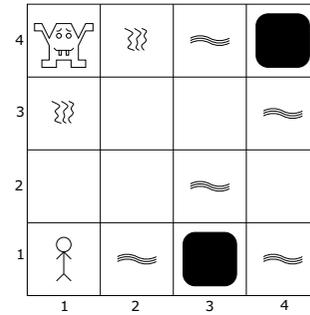


Figure 1. Example Wumpus world

Wumpus (square $\langle 1, 4 \rangle$), each of which are lethal (entering such a room causes the agent to die instantly). Initially, the agent is unaware about the locations of pits and Wumpus, but in adjacent squares it can sense a breeze (in the case of pits) or a stench (in the case of the Wumpus) and use that information to infer which grid cells are safe.

The basic reasoning task during the execution of a knowledge-based program that controls such an agent is to solve projection queries, i.e. to decide whether some formula about the agent’s knowledge after the execution of certain actions holds or not. For example,

$$[move(north)]\mathbf{K}([\textit{sensebreeze}]\exists r\mathbf{K}(Safe(r))) \quad (1)$$

expresses that after moving north, it is known that when a *sensebreeze* action is performed, the agent will come to know a room that is safe. Claßen and Lakemeyer [3] proposed a new formalization of knowledge-based programs based on the modal Situation Calculus variant \mathcal{ES} [11, 10] that includes a modal operator \mathbf{O} for only knowing and that resorts to Levesque and Lakemeyer’s [18] Representation Theorem in order to be able to evaluate conditions such as the above.

So far, this approach requires the agent’s knowledge base to only contain objective statements about the world such as the fact that a room r is only safe if it does not contain a pit or the Wumpus:

$$\neg Pit(r) \wedge \neg Wumpus(r) \supset Safe(r) \quad (2)$$

However, often we may want to exploit more of the expressiveness of \mathcal{ES} and add non-objective formulas that represent defaults, and hence enable non-monotonic reasoning. In the example, we could say that any room that is potentially unsafe should be considered dangerous (and thus avoided):

$$\mathbf{K}(Room(r)) \wedge \mathbf{M}(\neg Safe(r)) \supset Dangerous(r) \quad (3)$$

Whereas $\mathbf{K}(\alpha)$ is to be read as “ α is known”, $\mathbf{M}(\alpha)$ means “ α is consistent with what is known”. *Dangerous*(r) then is a default conclusion that may later be withdrawn in face of new information obtained

¹ Knowledge-Based Systems Group, RWTH Aachen University, Germany, email: classen@ksbg.rwth-aachen.de

² Knowledge-Based Systems Group, RWTH Aachen University, Germany, email: malte.neuss@rwth-aachen.de

through sensing: If the agent neither senses a breeze nor a stench near r , it concludes that that room neither contains a pit nor the Wumpus, and hence does not regard it as dangerous anymore.

The main contribution of this paper is a new variant of Levesque and Lakemeyer’s Representation Theorem for knowledge bases that may contain such non-objective default rules. In the same way that their method for objective knowledge bases reduces reasoning about action and knowledge to standard first-order theorem proving, ours will constitute a similar reduction to a standard non-monotonic reasoning method. In particular, we are interested in compatibility to Reiter’s Default Logic [30], which will allow us to resort to existing off-the-shelf reasoners for this formalism, including solvers for Answer Set Programming (ASP) [6] due to the well-known relationship to normal logic programs under the stable model semantics [7].

For this purpose, we combine \mathcal{ES} with $\mathcal{O}_3\mathcal{L}$ [12, 13], a logic that establishes an exact correspondence between three variants of the only-knowing operator \mathbf{O} (along with the meaning of \mathbf{M}) and different non-monotonic logics. In particular, Levesque’s original definition of only-knowing [18] corresponds to Moore’s Autoepistemic Logic [26], in which case $\mathbf{M}(\alpha)$ is interpreted as $\neg\mathbf{K}(\neg\alpha)$. Moreover, varying the semantics of \mathbf{O} enables to capture Reiter’s Default Logic, where the duality between \mathbf{M} and \mathbf{K} is given up, intuitively because “ungrounded” extensions are treated differently.³

The remainder of this paper is organized as follows. In Section 2, we introduce our new logic called \mathcal{ESD} that can be viewed as amalgamation of Lakemeyer and Levesque’s \mathcal{ES} (which supports action, sensing, but no Reiter-style defaults) with the default part of their logic $\mathcal{O}_3\mathcal{L}$ (which does not include actions and sensing). We then proceed to show that the new logic is a unifying formalism that serves as a coherent foundation for the definition of knowledge-based agents. For this purpose, in Section 3 we extend the standard Situation Calculus and \mathcal{ES} definitions of basic action theories (used for axiomatizing dynamic domains) to the \mathcal{ESD} case, and present a corresponding regression operator (used for reducing reasoning about actions to reasoning about the initial situation). Section 4 then contains our main contribution in form of a new variant of Levesque and Lakemeyer’s Representation Theorem for eliminating knowledge operators, now also accounting for defaults in the agent’s knowledge base. Section 5 then discusses how to combine all above mentioned results in order to implement a knowledge-based GOLOG agent capable of default reasoning. Finally, we review related work and conclude.

2 THE LOGIC \mathcal{ESD}

In this section we define a dynamic logic of only-knowing and default reasoning. It can be viewed as an amalgamation of the Default Logic part of Lakemeyer and Levesque’s static logic $\mathcal{O}_3\mathcal{L}$ [12, 13] with their modal epistemic Situation Calculus variant \mathcal{ES} [11].

2.1 Syntax

The alphabet of \mathcal{ESD} consists of the usual logical connectives and quantifiers, punctuation, parentheses, a countably infinite supply of first-order variables, equality, rigid predicates of any arity, fluent predicates of any arity (including the special predicates $Poss$ and SF for action preconditions and sensing, respectively), a countably

infinite set of standard names (which are syntactically treated as constants), the modal operators $[\cdot]$ and \square as well as the epistemic modal operators \mathbf{K} , \mathbf{M} , \mathbf{O}_R and \mathbf{O}_M . The *terms* are the variables and standard names. Formulas are defined inductively as follows:

- if t_1, \dots, t_k are terms and P is a predicate of arity k , then $P(t_1, \dots, t_k)$ is an (atomic) formula;
- if t_1 and t_2 are terms, then $(t_1 = t_2)$ is a formula;
- if α and β are formulas, x is a variable, and t a term, then $\neg\alpha$, $(\alpha \wedge \beta)$, $\forall x\alpha$, $\mathbf{K}(\alpha)$, $\mathbf{M}(\alpha)$, $\mathbf{O}_M(\alpha)$, $\mathbf{O}_R(\alpha)$, $[t]\alpha$ and $\square\alpha$ are also formulas.

Intuitively, $\mathbf{K}(\alpha)$ is read as “ α is known by the agent”, while $\mathbf{M}(\alpha)$ means “ α is consistent with what the agent knows”. Both $\mathbf{O}_M(\alpha)$ and $\mathbf{O}_R(\alpha)$ are to be read as “ α is all that is known”, with the difference that with \mathbf{O}_M , defaults will be evaluated according to Moore’s Autoepistemic Logic, whereas \mathbf{O}_R corresponds to Reiter’s Default Logic. Finally, $[t]\alpha$ means “ α holds after executing action t ” and $\square\alpha$ stands for “ α holds after any number of actions”. Also note that for simplicity we do not distinguish sorts, but allow any term to be used as an action.

We treat $(\alpha \vee \beta)$, $(\alpha \supset \beta)$, $\exists x\alpha$, truth \top , and falsity \perp as the usual abbreviations. For a finite sequence $z = \langle n_1, \dots, n_k \rangle$ of actions, we let $[z]\alpha$ stand for $[n_1] \dots [n_k]\alpha$. The notion of free and bound variables is defined in the usual way, and α_x^t means α with all free occurrences of x replaced by t . A formula without free variables is called a *sentence*, and an atomic formula $P(n_1, \dots, n_k)$ where all n_i are standard names is called *primitive sentence*. Formulas without any occurrence of epistemic modal operators are called *objective*, those where all predicates appear within the scope of an epistemic modal operator *subjective*, those without \mathbf{O}_R and \mathbf{O}_M *basic*, those without $[\cdot]$ and \square *static*, and those without \square *bounded*. A *fluent* formula is one that is objective, static and neither mentions $Poss$ nor SF .

2.2 Semantics

The semantics is as follows. Let \mathcal{N} denote the set of all standard names and \mathcal{Z} the set of all finite sequences z of standard names, including the empty sequence $\langle \rangle$. A *world* w is given by a mapping from the primitive sentences and \mathcal{Z} to truth values $\{0, 1\}$, respecting *rigidity*, i.e. if R is a rigid predicate, then for all $z, z' \in \mathcal{Z}$, $w[R(\vec{n}), z] = w[R(\vec{n}), z']$. Let \mathcal{W} be the set of all worlds. An *epistemic state* e is given by a set of worlds, i.e. a subset of \mathcal{W} . For two worlds w and w' and a sequence $z \in \mathcal{Z}$, $w \simeq_z w'$ (read: w and w' agree on the sensing for z) is inductively defined as follows:

- $w \simeq_{\langle \rangle} w'$ for every w and w' ;
- $w \simeq_{z \cdot n} w'$
iff $w \simeq_z w'$ and $w[SF(n), z] = w'[SF(n), z]$.

We are now ready to define the truth of sentences. \mathcal{ESD} , similar as $\mathcal{O}_3\mathcal{L}$, uses two epistemic states to interpret formulas, one to interpret formulas with \mathbf{K} , the other to interpret formulas with \mathbf{M} . We need this distinction because the duality $\mathbf{M}(\alpha) \equiv \neg\mathbf{K}(\neg\alpha)$ only holds in case of Moore’s Autoepistemic Logic (i.e. \mathbf{O}_M), but not for Reiter’s Default Logic (i.e. \mathbf{O}_R).

Formally, for any epistemic states e_1, e_2 , world w and $z \in \mathcal{Z}$, a sentence α is true wrt. e_1, e_2, w, z , which we write as $e_1, e_2, w, z \models \alpha$, as follows:

1. $e_1, e_2, w, z \models P(n_1, \dots, n_k)$
iff $w[P(n_1, \dots, n_k), z] = 1$;

³ The third logic is Konolige’s variant of Autoepistemic Logic using moderately grounded extensions [9]. For simplicity we do not consider it in this paper. Adapting our definitions and results accordingly is straightforward.

2. $e_1, e_2, w, z \models (n_1 = n_2)$
iff n_1 and n_2 are identical standard names;
3. $e_1, e_2, w, z \models \neg\alpha$ iff $e_1, e_2, w, z \not\models \alpha$;
4. $e_1, e_2, w, z \models \alpha \wedge \beta$
iff $e_1, e_2, w, z \models \alpha$ and $e_1, e_2, w, z \models \beta$;
5. $e_1, e_2, w, z \models \forall x\alpha$
iff $e_1, e_2, w, z \models \alpha_n^x$ for every standard name n ;

The above rules define the truth of atoms, equalities, and the usual logical connectives in the presence of standard names. The latter can be thought of as a countably infinite set of constants that satisfy the unique names assumption and an infinitary version of domain closure. Thus, first-order quantifiers can be interpreted substitutionally. Next, action modalities are defined similar as for \mathcal{ES} :

6. $e_1, e_2, w, z \models [n]\alpha$
iff $e_1, e_2, w, z \cdot n \models \alpha$;
7. $e_1, e_2, w, z \models \Box\alpha$
iff $e_1, e_2, w, z \cdot z' \models \alpha$ for all z' ;

Here, $z \cdot n$ refers to the result of concatenating standard name n at the end of sequence z . The meaning of belief modalities is given as follows:

8. $e_1, e_2, w, z \models \mathbf{K}(\alpha)$
iff for every $w' \in e_1$ with $w' \simeq_z w$, $e_1, e_2, w', z \models \alpha$;
9. $e_1, e_2, w, z \models \mathbf{M}(\alpha)$
iff for some $w' \in e_2$ with $w' \simeq_z w$, $e_1, e_2, w', z \models \alpha$;
10. $e_1, e_2, w, z \models \mathbf{O}_M(\alpha)$
iff for every w' with $w' \simeq_z w$,
 $e_1, e_2, w', z \models \alpha$ iff $w' \in e_1$;
11. $e_1, e_2, w, z \models \mathbf{O}_R(\alpha)$
iff for all e' with $e_1 \subseteq e'$,
 $e', e_2, w, z \models \mathbf{O}_M(\alpha)$ iff $e' = e_1$.

That is to say a sentence is known iff its true in every world of e_1 , while a sentence is consistent with what is known iff some world of e_2 satisfies it. \mathbf{O}_M coincides with Levesque's only-knowing (essentially the "if" in the case of \mathbf{K} becomes an "iff"), while \mathbf{O}_R is a variant that allows to make the connection to Reiter's Default Logic. In any case, only worlds that agree with the "real" world w on the sensing throughout z are considered, which means additional knowledge is gained by ruling out incompatible possible worlds.

We then define $e, w \models \alpha$ as $e, e, w, \langle \rangle \models \alpha$. A sentence α is *valid* (written as $\models \alpha$) iff $e, w \models \alpha$ for every e and w . A set of sentences Σ *entails* α (written as $\Sigma \models \alpha$) iff for every e, w such that $e, w \models \beta$ for all $\beta \in \Sigma$, also $e, w \models \alpha$.

Note that thus, in the above rules e_1 and e_2 are usually equal, the only exception being \mathbf{O}_R where e' ranges over all possible supersets of e_1 to ensure its minimality (corresponding to a maximal set of \mathbf{K} -beliefs).

For $z = \langle \rangle$, the truth of subjective sentences does not depend on any world, which is why we often omit the w argument and write $e \models \alpha$ in that case. Similarly, we may write $w \models \phi$ for objective sentences ϕ . We will furthermore identify a finite set of sentences (called a *knowledge base*) with the singleton sentence given by the conjunction of all sentences in the set, i.e. we use a loose notation where a finite set can be used anywhere a formula could.

2.3 Properties

We first note the close relation of \mathcal{ESD} to \mathcal{ES} and $\mathcal{O}_3\mathcal{L}$:

Theorem 1. *For every sentence α without \mathbf{O}_R and \mathbf{M} , α is valid in \mathcal{ESD} iff α is valid in \mathcal{ES} .*

Proof. Obvious from the fact that syntax-wise, sentences without \mathbf{O}_R and \mathbf{M} are a subset of \mathcal{ES} and the relevant semantic rules of \mathcal{ESD} coincide with those of \mathcal{ES} . \square

Theorem 2. *For every static sentence α , α is valid in \mathcal{ESD} iff α is valid in $\mathcal{O}_3\mathcal{L}$.*

Proof. Obvious from the fact that syntax-wise, static sentences are a subset of $\mathcal{O}_3\mathcal{L}$ and the semantic rules of \mathcal{ESD} coincide with those of $\mathcal{O}_3\mathcal{L}$ when $z = \langle \rangle$. \square

We may hence directly exploit existing results for these subformalisms. If we move to the propositional case, one particularly interesting result for us here is the following relation to Reiter's Default Logic shown by Lakemeyer and Levesque [12], where a Reiter-style default of the form

$$\alpha : \beta_1, \dots, \beta_k / \gamma \quad (4)$$

is represented by the formula

$$\mathbf{K}(\alpha) \wedge \mathbf{M}(\beta_1) \wedge \dots \wedge \mathbf{M}(\beta_k) \supset \gamma. \quad (5)$$

While Reiter allowed for open defaults that have free variables, which then stand for the set of all their possible ground instances, we here make the restriction to propositional theories, following Lakemeyer and Levesque. A *propositional default theory* $\langle F, D \rangle$ then consists of a finite set F of static, objective, quantifier-free sentences as well as a finite set D of closed defaults of the form (4) where all of $\alpha, \beta_1, \dots, \beta_k, \gamma$ are static, objective, and quantifier-free.

Theorem 3. *Let $\langle F, D \rangle$ be a propositional default theory, ϕ the conjunction of sentences over F , and δ the conjunction of the representations of the defaults D according to (5). Then Γ is a standard Reiter extension of $\langle F, D \rangle$ (as defined in [30]) iff there is an e such that $e \models \mathbf{O}_R(\phi \wedge \delta)$ and Γ is the set of objective beliefs of e , i.e. $\Gamma = \{\psi \mid e \models \mathbf{K}(\psi), \psi \text{ static, objective and quantifier-free}\}$.*

That is to say an epistemic state that Reiter-only-knows a knowledge base of the mentioned form corresponds exactly to an extension according to Reiter's default semantics in the sense that they believe the same objective formulas. As an important corollary, we have:

Corollary 1. $\mathbf{O}_R(\phi \wedge \delta) \models \mathbf{K}(\psi)$ iff ψ is an element of every Reiter extension of $\langle F, D \rangle$.

Note that there is a similar correspondence to Autoepistemic Logic when \mathbf{O}_M is used instead of \mathbf{O}_R . The difference between the two, roughly, is how they deal with "ungrounded" expansions/extensions. While there is no default theory corresponding to only-knowing $\mathbf{K}(p) \supset p$, Autoepistemic Logic admits an expansion where p is believed.

We will also heavily rely on the following theorem of [12]. Note that while there are in general multiple possible e with $e \models \mathbf{O}_R(\Sigma_0)$ for any non-objective knowledge base Σ_0 , the epistemic state that only-knows an objective knowledge base Φ is unique, and is given by the set of all worlds satisfying Φ .

Theorem 4. *Let Σ_0 be a static basic knowledge base without quantifiers. Then there is a finite set of static, objective, quantifier-free sentences $\mathfrak{E}(\Sigma_0) = \{\Phi_1, \dots, \Phi_n\}$ such that*

$$\models \mathbf{O}_R(\Sigma_0) \equiv (\mathbf{O}_M(\Phi_1) \vee \dots \vee \mathbf{O}_M(\Phi_n)).$$

Intuitively, if Σ_0 is of the right form (that corresponds to a default theory), $\mathfrak{E}(\Sigma_0)$ thus represents exactly the possible extensions of Σ_0 .

3 BASIC ACTION THEORIES AND REGRESSION

Similar as in the classical Situation Calculus and \mathcal{ES} , we use basic action theories to define the agent's knowledge about the initial situation, pre- and postconditions of actions as well as sensing results:

Definition 1. A basic action theory (BAT) is a set of sentences of the form $\Sigma = \Sigma_0 \cup \Sigma_{pre} \cup \Sigma_{post} \cup \Sigma_{sense}$, where

1. Σ_0 , the initial theory, is a static basic knowledge base describing the initial state of the world.
2. Σ_{pre} is a precondition axiom of the form $\Box Poss(a) \equiv \pi$, where π is a fluent formula and a its only free variable.⁴
3. Σ_{post} is a finite set of successor state axioms (SSAs) $\Box[a]F(\vec{x}) \equiv \gamma_F$, one for each fluent predicate F relevant to the application domain, where γ_F is a fluent formula whose only free variables are \vec{x} and a . SSAs incorporate Reiter's [31] solution to the frame problem.⁵
4. Σ_{sense} is a sensing axiom of the form $\Box SF(a) \equiv \varphi$, where φ is a fluent formula and a its only free variable.

Note that while usually the initial theory Σ_0 is assumed to be objective, we here allow that it contains \mathbf{K} and \mathbf{M} operators in order to be able to encode defaults.

Example 1. For the Wumpus domain, a BAT may look as follows. For simplicity, we ignore shooting arrows and grabbing the gold. The initial theory Σ_0 first of all contains objective formulas encoding the adjacency relation between rooms as well the initial position of the agent, which is safe:

$$Adj(room_{11}, north, room_{12}), \dots, Adj(room_{43}, south, room_{44}) \\ At(room_{11}) \wedge Safe(room_{11})$$

where safety is defined as in (2). Additionally, there can be default rules such as (3) that lets the agent assume any room to be dangerous about which it is not absolutely sure. Moreover, we can use defaults to make the closed-world assumption about certain parts of the domain:

$$\mathbf{M}(\neg Adj(r, d, r')) \supset \neg Adj(r, d, r')$$

This means whenever it is safe to assume that two rooms are not connected, the agent believes that they indeed are not. Hence, the facts about Adj above list all and only cases where two rooms are adjacent.

Next, we have a precondition axiom Σ_{pre} stating that moving in a direction d is possible if there is an adjacent room, and sensing breezes and stench is always possible:⁶

$$\Box Poss(a) \equiv \exists r, d, r' (At(r) \wedge a = move(d) \wedge Adj(r, d, r')) \vee \\ a = sensebreeze \vee a = sensestench$$

⁴ Free variables in BAT formulas are implicitly assumed to be universally quantified from the outside. Furthermore, \Box has lower syntactic precedence than the logical connectives, so $\Box Poss(a) \equiv \pi$ stands for $\forall a. \Box (Poss(a) \equiv \pi)$.

⁵ The $[a]$ construct has higher precedence than the logical connectives, so $\Box[a]F(\vec{x}) \equiv \gamma_F$ abbreviates $\forall a. \Box (([a]F(\vec{x})) \equiv \gamma_F)$.

⁶ We abuse notation here when using an action function such as $move(d)$ as our formalism does not include function symbols. Note that in the example, there are exactly four directions, so a quantified formula such as $\exists d \phi$ is to be understood as shorthand for the finite disjunction $\phi_n^d \vee \phi_e^d \vee \phi_s^d \vee \phi_w^d$, where we identify $move(n)$, $move(e)$, $move(s)$, $move(w)$ with the standard names $move_n$, $move_e$, $move_s$, $move_w$, respectively. The restriction to standard names only is to keep the formal treatment in this paper as simple as possible and comes without loss of generality (as actions will be eliminated by means of regression); an extension by action functions as used in [3] is straightforward.

For the At fluent, the successor state axiom in Σ_{post} looks like this:

$$\Box[a]At(x) \equiv \exists r, d (At(r) \wedge a = move(d) \wedge Adj(r, d, x)) \vee \\ At(x) \wedge \neg \exists d a = move(d)$$

That is to say the position of the agent after action a will be x if a was moving there from some adjacent room r by going in direction d , or the agent was already at x and did not move.

Finally, the sensing axiom Σ_{sense} expresses that SF is true for sensebreeze iff there is a pit in an adjacent room, and true for sensestench iff the Wumpus is nearby. For non-sensing actions such as $move$, SF will never hold:

$$\Box SF(a) \equiv a = sensebreeze \\ \wedge \exists r, d, r' (At(r) \wedge Adj(r, d, r') \wedge Pit(r')) \vee \\ a = sensestench \\ \wedge \exists r, d, r' (At(r) \wedge Adj(r, d, r') \wedge Wumpus(r'))$$

Using BATs, we can also define a regression operator for eliminating actions from formulas as is done in Reiter's Situation Calculus. For the objective case, we have the following:

Definition 2. For any bounded, objective sentence α and BAT Σ let $\mathcal{R}[\alpha]$, the regression of α wrt Σ , be the fluent formula $\mathcal{R}[\langle \rangle, \alpha]$, where for any sequence of terms σ (not necessarily ground), $\mathcal{R}[\sigma, \alpha]$ is defined inductively on α by:

1. $\mathcal{R}[\sigma, (t_1 = t_2)] = (t_1 = t_2)$;
2. $\mathcal{R}[\sigma, \neg \alpha] = \neg \mathcal{R}[\sigma, \alpha]$;
3. $\mathcal{R}[\sigma, (\alpha \wedge \beta)] = (\mathcal{R}[\sigma, \alpha] \wedge \mathcal{R}[\sigma, \beta])$;
4. $\mathcal{R}[\sigma, \forall x \alpha] = \forall x \mathcal{R}[\sigma, \alpha]$;
5. $\mathcal{R}[\sigma, [t] \alpha] = \mathcal{R}[\sigma \cdot t, \alpha]$;
6. $\mathcal{R}[\sigma, Poss(t)] = \mathcal{R}[\sigma, \pi_t^a]$;
7. $\mathcal{R}[\sigma, SF(t)] = \mathcal{R}[\sigma, \varphi_t^a]$;
8. $\mathcal{R}[\sigma, R(t_1, \dots, t_k)] = R(t_1, \dots, t_k)$ if R is rigid;
9. $\mathcal{R}[\sigma, F(t_1, \dots, t_k)]$ for fluent F is defined inductively on σ by:
 - (a) $\mathcal{R}[\langle \rangle, F(t_1, \dots, t_k)] = F(t_1, \dots, t_k)$;
 - (b) $\mathcal{R}[\sigma \cdot t, F(t_1, \dots, t_k)] = \mathcal{R}[\sigma, (\gamma_F)_{t_1^{x_1} \dots t_k^{x_k}}]$.

Above, π , φ and γ_F are the right-hand sides of the corresponding axioms in Σ_{pre} , Σ_{sense} , and Σ_{post} , respectively.

The general idea here is to subsequently replace formulas of the form $[t]F(\vec{t})$, $Poss(t)$ and $SF(t)$ by equivalent formulas (that do not mention actions) as defined by the BAT. Iterating such steps, a formula involving actions is thus transformed into an equivalent formula that only talks about the initial situation.

When it comes to \mathbf{K} operators, consider the following theorem for \mathcal{ES} which still holds in \mathcal{ESD} :

Theorem 5.

$$\models \Box[a] \mathbf{K}(\alpha) \equiv \\ SF(a) \wedge \mathbf{K}(SF(a) \supset [a]\alpha) \vee \\ \neg SF(a) \wedge \mathbf{K}(\neg SF(a) \supset [a]\alpha).$$

Proof. Similar as for \mathcal{ES} . □

The above is like a successor state axiom for the \mathbf{K} operator in the sense that it relates knowing some formula after an action to the truth of a formula talking about the situation before executing that action. The difference is that this is not an axiom, but a theorem of the logic (i.e. a valid sentence). In addition, we now also have a similar one for \mathbf{M} :

Theorem 6.

$$\begin{aligned} \models \Box[a]\mathbf{M}(\alpha) \equiv \\ SF(a) \supset \mathbf{M}(SF(a) \wedge [a]\alpha) \wedge \\ \neg SF(a) \supset \mathbf{M}(\neg SF(a) \wedge [a]\alpha). \end{aligned}$$

Proof. “ \Rightarrow ”: Let $e_1, e_2, w, z \models [n]\mathbf{M}(\alpha)$. Then for some $w' \in e_2$ with $w' \simeq_{z \cdot n} w$, $e_1, e_2, w', z \cdot n \models \alpha$. If $w, z \models SF(n)$, we thus have some $w' \in e_2$ with $w' \simeq_z w$, $w', z \models SF(n)$ and $e_1, e_2, w', z \models [n]\alpha$, hence $e_1, e_2, w, z \models \mathbf{M}(SF(n) \wedge [n]\alpha)$.

“ \Leftarrow ”: Conversely, let $e_1, e_2, w, z \models SF(n) \supset \mathbf{M}(SF(n) \wedge [n]\alpha) \vee \neg SF(n) \supset \mathbf{M}(\neg SF(n) \wedge [n]\alpha)$ and suppose $w, z \models SF(n)$ (the other case is similar). Then $e_1, e_2, w, z \models \mathbf{M}(SF(n) \supset [n]\alpha)$, hence there is $w' \in e_2$ with $w' \simeq_z w$ such that $w', z \models SF(n)$ and $e_1, e_2, w', z \models [n]\alpha$. Therefore we have $w' \in e_2$ with $w' \simeq_{z \cdot n} w$ and $e_1, e_2, w, z \cdot n \models \alpha$, i.e. $e_1, e_2, w, z \models [n]\mathbf{M}(\alpha)$. \square

Similarly as we use regular SSAs for regressing fluent atoms, we can therefore use the last two theorems to regress formulas involving \mathbf{K} and \mathbf{M} . Formally, we add the following two regression rules to Definition 2:

10. $\mathcal{R}[\sigma, \mathbf{K}(\alpha)]$ is defined inductively on σ by:

- (a) $\mathcal{R}[\langle \rangle, \mathbf{K}(\alpha)] = \mathbf{K}(\mathcal{R}[\langle \rangle, \alpha])$;
- (b) $\mathcal{R}[\sigma \cdot t, \mathbf{K}(\alpha)] = \mathcal{R}[\sigma, \kappa_t^a]$,
where κ is the right-hand side of the equivalence in Theorem 5.

11. $\mathcal{R}[\sigma, \mathbf{M}(\alpha)]$ is defined inductively on σ by:

- (a) $\mathcal{R}[\langle \rangle, \mathbf{M}(\alpha)] = \mathbf{M}(\mathcal{R}[\langle \rangle, \alpha])$;
- (b) $\mathcal{R}[\sigma \cdot t, \mathbf{M}(\alpha)] = \mathcal{R}[\sigma, \mu_t^a]$,
where μ is the right-hand side of the equivalence in Theorem 6.

Example 2. Consider the BAT from Example 1. The reader may verify that the following are true for regressing objective formulas:

$$\begin{aligned} \mathcal{R}[[move(north)]At(x)] &= \exists r (At(r) \wedge Adj(r, north, x)) \\ \mathcal{R}[[a]Safe(x)] &= Safe(x) \\ \mathcal{R}[[sensebreeze]\alpha] &= \alpha \\ \mathcal{R}[SF(sensebreeze)] &= \exists r, d, r' (At(r) \wedge Adj(r, d, r') \wedge Pit(r')) \end{aligned}$$

That is to say after moving north the agent is at x iff x is north of the agent’s previous position x . As *Safe* is a rigid predicate, its truth value does not change by means of any action a . Also, *sensebreeze* is a pure sensing action that does not have any effect on fluents, hence regressing any objective formula α through it leaves α unchanged.

Let *PitNearby* stand for $\exists r, d, r' (At(r) \wedge Adj(r, d, r') \wedge Pit(r'))$ and s for *sensebreeze*. Then we have:

$$\begin{aligned} \mathcal{R}[\langle s \rangle, \mathbf{K}(Safe(x))] \\ &= \mathcal{R}[\langle \rangle, SF(s) \wedge \mathbf{K}(SF(s) \supset [s]Safe(x)) \vee \\ &\quad \neg SF(s) \wedge \mathbf{K}(\neg SF(s) \supset [s]Safe(x))] \\ &= PitNearby \wedge \mathbf{K}(PitNearby \supset Safe(x)) \vee \\ &\quad \neg PitNearby \wedge \mathbf{K}(\neg PitNearby \supset Safe(x)). \end{aligned}$$

In other words the agent comes to know that room x is safe after sensing for breezes at its current location just in case there is a pit nearby and it is known that even when a pit is nearby, x is close (which can only be when x is not adjacent to the pit), or if there is no pit nearby and the agent knows that then x is safe (e.g. when the agent’s position is adjacent to x and it can also rule out that x contains the Wumpus). As a next step, the above regression will then

have to be checked against the agent’s knowledge about the initial situation, as we will discuss in the next section. Whether or not x is actually known to be safe thus also depends on information it has gathered through previous sensing actions at various locations.

Regression is correct as given by the following theorem:

Theorem 7. Let Σ be a BAT and α a bounded, basic sentence. Then $\mathcal{R}[\alpha]$ is static and

- 1. $\mathbf{O}_R(\Sigma) \models \mathbf{K}(\alpha)$ iff $\mathbf{O}_R(\Sigma_0) \models \mathbf{K}(\mathcal{R}[\alpha])$.
- 2. $\mathbf{O}_R(\Sigma) \models \mathbf{M}(\alpha)$ iff $\mathbf{O}_R(\Sigma_0) \models \mathbf{M}(\mathcal{R}[\alpha])$.

Proof. Similar to the proof of Theorem 5 in [11], this follows from the fact that all transformations given by Definitions 2 and Theorems 5 and 6 are equivalence preserving wrt the BAT Σ . \square

Hence we can reduce reasoning about knowledge and action to reasoning about knowledge in the initial situation. In order to also represent the situation where some history of actions z has already been executed, we need the following:

Definition 3. Let $z = \langle n_1 \cdots n_k \rangle \in \mathcal{Z}$. A sensing result for z is a formula of the form

$$\bigwedge_{i=1}^k [n_i] \cdots [n_{i-1}] \pm SF(n_i),$$

where each $\pm SF(n_i)$ is either $SF(n_i)$ or $\neg SF(n_i)$. As a special case, \top is the only sensing result for $\langle \rangle$.

By incorporating sensing results through regression into the knowledge base, coming to know that α holds after executing z is equivalent to knowing initially that after z , α will come to hold:

Theorem 8. Let Σ be a BAT, α a sentence, $z \in \mathcal{Z}$, and Ψ a sensing result for z .

- 1. $\mathbf{O}_R(\Sigma) \wedge \Psi \models [z]\mathbf{K}(\alpha)$ iff $\mathbf{O}_R(\Sigma \wedge \mathcal{R}[\Psi]) \models \mathbf{K}([z]\alpha)$.
- 2. $\mathbf{O}_R(\Sigma) \wedge \Psi \models [z]\mathbf{M}(\alpha)$ iff $\mathbf{O}_R(\Sigma \wedge \mathcal{R}[\Psi]) \models \mathbf{M}([z]\alpha)$.

Proof. Easy to show using the fact that for evaluating both \mathbf{K} and \mathbf{M} operators, the semantics only considers worlds w' in the epistemic states that agree with the real world w on the sensing throughout z . \square

Note that since $\mathcal{R}[\Psi]$ is a fluent formula, $\Sigma \wedge \mathcal{R}[\Psi]$ again conforms to our definition of a BAT as we may view $\mathcal{R}[\Psi]$ as being part of the new BAT’s initial theory, i.e. we set $\Sigma'_0 = \Sigma_0 \cup \{\mathcal{R}[\Psi]\}$.

4 REPRESENTATION THEOREM

To deal with knowledge, we propose a variant of Levesque and Lake-meyer’s [18] Representation Theorem as follows. The general idea is to recursively replace occurrences of $\mathbf{K}(\alpha)$ by objective formulas that represent the known instances of the corresponding α according to the knowledge base Σ_0 .

One important difference of our approach to Levesque and Lake-meyer’s is that they directly evaluate such $\mathbf{K}(\alpha)$ formulas against the (objective) KB, whereas we defer this evaluation to a later point. Instead, we first recursively replace every $\mathbf{K}(\alpha)$ subformula in the query by a corresponding K_α , which is a special, reserved predicate not occurring in the original BAT and query, and additionally augment Σ_0 by the statement $\mathbf{K}(\alpha) \supset K_\alpha$ (and similar for \mathbf{M}). Intuitively, the new predicate serves as a “flag” to indicate when the

corresponding belief subformula holds in an extension, and the new default rule ensures that the flag is assigned the correct value. Formally:

Definition 4. Given a static basic formula α , $\|\alpha\|$ is defined by

1. $\|\alpha\| = \alpha$, when α is a static objective formula;
2. $\|\neg\alpha\| = \neg\|\alpha\|$;
3. $\|\alpha \wedge \beta\| = \|\alpha\| \wedge \|\beta\|$;
4. $\|\exists x\alpha\| = \exists x\|\alpha\|$;
5. $\|\mathbf{K}(\alpha)\| = K_\phi(\vec{x})$,
where $\phi = \|\alpha\|$ and \vec{x} are the free variables in α ;
6. $\|\mathbf{M}(\alpha)\| = M_\phi(\vec{x})$,
where $\phi = \|\alpha\|$ and \vec{x} are the free variables in α .

Moreover, $\Sigma_0 \uparrow \alpha$ is given by:

1. $\Sigma_0 \uparrow \alpha = \Sigma_0$, when α is a static objective formula;
2. $\Sigma_0 \uparrow \neg\alpha = \Sigma_0 \uparrow \alpha$;
3. $\Sigma_0 \uparrow \alpha \wedge \beta = \Sigma_0 \uparrow \alpha \cup \Sigma_0 \uparrow \beta$;
4. $\Sigma_0 \uparrow \exists x\alpha = \Sigma_0 \uparrow \alpha$;
5. $\Sigma_0 \uparrow \mathbf{K}(\alpha) = \Sigma_0 \uparrow \alpha \cup \{\mathbf{K}(\phi) \supset K_\phi(\vec{x})\}$,
where $\phi = \|\alpha\|$ and \vec{x} are the free variables in α ;
6. $\Sigma_0 \uparrow \mathbf{M}(\alpha) = \Sigma_0 \uparrow \alpha \cup \{\mathbf{M}(\phi) \supset M_\phi(\vec{x})\}$,
where $\phi = \|\alpha\|$ and \vec{x} are the free variables in α .

We call $\|\alpha\|$ the reduction of α and $\Sigma_0 \uparrow \alpha$ the augmentation of Σ_0 .

Example 3. If $\alpha = \exists x. \mathbf{K}(\text{Room}(x) \wedge \neg\mathbf{K}(\text{Pit}(x)))$, $\Sigma_0 \uparrow \alpha$ is Σ_0 together with the sentences

$$\begin{aligned} \mathbf{K}(\text{Pit}(x)) &\supset K_{\text{Pit}(x)}(x), \\ \mathbf{K}(\text{Room}(x) \wedge \neg\mathbf{K}(\text{Pit}(x))) &\supset K_\phi(x), \end{aligned}$$

where $\phi \stackrel{\text{def}}{=} \|\text{Room}(x) \wedge \neg\mathbf{K}(\text{Pit}(x))\| = \text{Room}(x) \wedge \neg K_{\text{Pit}(x)}(x)$. $\|\alpha\|$ then is $\exists x K_\phi(x)$.

The above construction is correct as follows. First, consider the case without nesting of \mathbf{K} and \mathbf{M} . If e is an epistemic state such that $e \models \mathbf{O}_R(\Sigma_0)$, let

$$\begin{aligned} e \uparrow \mathbf{K}(\phi) &\stackrel{\text{def}}{=} \{w \in e \mid \text{if } e \models \mathbf{K}(\phi)_{\vec{n}}, \text{ then } w \models K_\phi(\vec{n})\} \\ e \uparrow \mathbf{M}(\phi) &\stackrel{\text{def}}{=} \{w \in e \mid \text{if } e \models \mathbf{M}(\phi)_{\vec{n}}, \text{ then } w \models M_\phi(\vec{n})\} \end{aligned}$$

for objective ϕ where K_ϕ and M_ϕ do not appear in Σ_0 , respectively. Intuitively, $e \uparrow \mathbf{K}(\phi)$ is like e that originally Reiter-only-knows the knowledge base Σ_0 , but where in addition all its worlds also set the right values for the flag predicate associated with $\mathbf{K}(\phi)$. Then we have that $e \uparrow \mathbf{K}(\phi)$ Reiter-only-knows the augmentation (similar for $\mathbf{M}(\phi)$):

Lemma 1. Let Σ_0 be a static basic knowledge base, ϕ an objective formula with free variables \vec{x} , and e, e' and e'' be epistemic states such that $e \models \mathbf{O}_R(\Sigma_0)$, $e' = e \uparrow \mathbf{K}(\phi)$ and $e'' = e \uparrow \mathbf{M}(\phi)$. Then

1. $e \models \mathbf{O}_R(\Sigma_0)$ iff $e' \models \mathbf{O}_R(\Sigma_0 \uparrow \mathbf{K}(\phi))$
2. $e \models \mathbf{O}_R(\Sigma_0)$ iff $e'' \models \mathbf{O}_R(\Sigma_0 \uparrow \mathbf{M}(\phi))$

and for all standard names \vec{n} ,

1. $e \models \mathbf{K}(\phi)_{\vec{n}}$ iff for all $w \in e'$, $w \models K_\phi(\vec{n})$;
2. $e \models \mathbf{M}(\phi)_{\vec{n}}$ iff for all $w \in e''$, $w \models M_\phi(\vec{n})$.

Proof. Since K_ϕ does not appear in Σ_0 , e' behaves exactly like e except for the fact that $K_\phi(\vec{n})$ holds in all its worlds iff $\mathbf{K}(\phi)$ holds in e . Similar for e'' with $M_\phi(\vec{n})$ and $\mathbf{M}(\phi)$. \square

Thus, we have for the recursive evaluation of formulas:

Lemma 2. Let Σ_0 be a static basic knowledge base, α a static basic formula with free variables \vec{x} , and e and e' epistemic states such that $e \models \mathbf{O}_R(\Sigma_0)$ and $e' = e \uparrow \alpha$. Then $\Sigma_0 \uparrow \alpha$ is a static basic knowledge base, $\|\alpha\|$ is a static objective formula, and for all worlds $w \in e'$ and standard names \vec{n} ,

$$e, w \models \alpha_{\vec{n}} \text{ iff } w \models \|\alpha\|_{\vec{n}}.$$

Proof. This can be proven by an induction on the structure of α , using Lemma 1 for the base cases of \mathbf{K} and \mathbf{M} without nesting. \square

We now exploit Theorem 4 guaranteeing that in the propositional case, Reiter-only-knowing Σ_0 boils down to only-knowing one of finitely many objective knowledge bases:

Theorem 9. Let Σ_0 be a static basic knowledge base without quantifiers, $\mathfrak{E}(\dots)$ as in Theorem 4, and α a static basic sentence without quantifiers. Then

1. $\mathbf{O}_R(\Sigma_0) \models \mathbf{K}(\alpha)$ iff for all $\Phi \in \mathfrak{E}(\Sigma_0 \uparrow \alpha)$, $\Phi \models \|\alpha\|$.
2. $\mathbf{O}_R(\Sigma_0) \models \mathbf{M}(\alpha)$ iff for all $\Phi \in \mathfrak{E}(\Sigma_0 \uparrow \alpha)$, $\Phi \not\models \|\neg\alpha\|$.

Proof. This follows from Lemma 2 and Theorem 4. \square

Thus, testing a formula $\mathbf{K}(\alpha)$ or $\mathbf{M}(\alpha)$ (possibly containing nested modalities) against a knowledge base Σ_0 (possibly containing defaults) is reducible to classical propositional entailment: First, construct the augmented knowledge base $\Sigma_0 \uparrow \alpha$ and determine its extensions. Then, for every one of the finitely many, objective extensions Φ check whether the reduced query $\|\neg\alpha\|$ holds in it. Note that in the propositional case, the formulas by which we augment Σ_0 can be viewed as default rules of the form (5) with either α or the β_j being empty, i.e. TRUE.

Regarding complexity, suffice it to say (without giving a formal analysis) that our Representation Theorem will not be harder than classical default reasoning (i.e., Σ_2^P -complete). The reason is that augmentation adds as many additional defaults as the query contains belief operators, but their only purpose is to set the right “flags” (values for the K_α and M_α predicates) to indicate which belief subformulas hold. The number and the internal structure of extensions remains unchanged.

5 COMPUTING EXTENSIONS

The last two sections gave us the main ingredients for implementing a knowledge-based agent whose main reasoning task during the execution of a knowledge-based program is to decide queries of the forms

$$\begin{aligned} \mathbf{O}_R(\Sigma) \wedge \Psi &\models [z]\mathbf{K}(\alpha) \\ \mathbf{O}_R(\Sigma) \wedge \Psi &\models [z]\mathbf{M}(\alpha) \end{aligned}$$

where Ψ is the sensing result obtained for z . Theorems 8 and 7 tell us that regression can be used to incorporate sensing results into the BAT and reduce the problem to reasoning about knowledge in the initial situation only. In the propositional case that we consider here,

we can then apply Theorem 9 to further reduce the problem to propositional entailment checks. The only missing piece of the puzzle now is how to determine the set $\mathfrak{E}(\dots)$ from Theorem 4.

There are multiple options. The direct approach would be to make use of Reiter's [30] theorem that characterizes extensions by means of a fixpoint construction. Let $\langle F, D \rangle$ be the default theory corresponding to our knowledge base Σ_0 , where F are the objective formulas (facts), and D the defaults corresponding to formulas of the form (5). Procedure 1 depicts the algorithm to compute extensions in pseudo code. After initializing the result to the empty set, we first determine all objective subformulas Φ in the theory (line 2). We then consider all subsets of Φ as candidates for extensions (line 3). To check whether some candidate E actually represents an extension, we start with the set of facts (line 4) and incrementally add conclusions γ for defaults whose prerequisite α holds in the previous set and whose negated justifications $\neg\beta_j$ are not in E (line 8). Since there are only finitely many defaults and we consider propositional logic, this process will eventually converge to a fixpoint (line 9). If the resulting E_i is equivalent to E (line 10), we have found an extension and include E in the result.

Procedure 1 Calculating Default Extensions

Input: a propositional default theory $\langle F, D \rangle$

Output: its set of extensions $\mathfrak{E}(\langle F, D \rangle)$

```

1:  $\mathfrak{E}(\langle F, D \rangle) := \emptyset;$ 
2:  $\Phi := F \cup \{\alpha, \beta_1, \dots, \beta_k, \gamma \mid \alpha : \beta_1, \dots, \beta_k / \gamma \in D\};$ 
3: for all  $E \subseteq \Phi$  do
4:    $E_0 := F;$ 
5:    $i := 0;$ 
6:   repeat
7:      $i := i + 1;$ 
8:      $E_i := E_{i-1} \cup \{\gamma \mid \alpha : \beta_1, \dots, \beta_k / \gamma \in D,$ 
        $E_{i-1} \models \alpha, \neg\beta_1, \dots, \neg\beta_k \notin E_i\};$ 
9:   until  $E_i = E_{i-1};$ 
10:  if for all  $\phi \in \Phi$ ,  $E \models \phi$  iff  $E_i \models \phi$  then
11:     $\mathfrak{E}(\langle F, D \rangle) := \mathfrak{E}(\langle F, D \rangle) \cup \{E\};$ 
12:  end if
13: end for
14: return  $\mathfrak{E}(\langle F, D \rangle)$ 

```

Note that similar to our Representation Theorem, we here again reduce the overall reasoning task to a finite number of entailment tests in classical propositional logic. Hence, in principle, all we need is a propositional reasoner (SAT solver).

However, instead of coming up with a complete re-implementation of a default reasoner, it is also possible to make use of existing off-the-shelf systems, i.e. resort to a default reasoner such as XRay [27] or DeReS [2]. Another interesting option is to employ state-of-the-art ASP solvers implementing the stable model semantics, for example `clasp`⁷. Gelfond and Lifschitz [7] showed the relation of stable models of a normal logic program with classical negation to a fragment of Reiter's Default Logic. The key idea is to identify a default

$$B_1 \wedge \dots \wedge B_k : \overline{B_{k+1}}, \dots, \overline{B_{k+m}} / A \quad (6)$$

with the rule

$$A \leftarrow B_1, \dots, B_k, \text{not}(B_{k+1}), \dots, \text{not}(B_{k+m}) \quad (7)$$

and a fact

$$B_1 \wedge \dots \wedge B_k \supset A \quad (8)$$

with a rule

$$A \leftarrow B_1, \dots, B_k \quad (9)$$

not containing negation as failure. Here, A and the B_i are literals and \overline{L} denotes the complement of L . The correspondence then is as follows:

Theorem 10. *Let P be a program consisting of rules of the form (7) and (9), D_P the corresponding defaults (6) and F_P the corresponding facts (8). Then M is a stable model of P iff M is a maximal set of atoms such that $\text{Th}(F_P \cup M)$ is an extension of $\langle F_P, D_P \rangle$.*

Therefore, if we require Σ_0 to only contain formulas of the form (7) and (9), we can use an ASP solver to compute the extensions $\mathfrak{E}(\Sigma_0)$. Note that in general however, subformulas in facts and defaults of our knowledge base are not necessarily literals, but can be of arbitrary form, in particular when they originate from regressed sensing results $\mathcal{R}[\Psi]$ or $\mathbf{K}(\phi)$ and $\mathbf{M}(\phi)$ subformulas introduced by augmentation. One possible direction for future work is identifying syntactical restrictions on the knowledge base and queries such that the required form can be enforced. Alternatively, we can look for a way to exploit results relating SAT to ASP: Every Boolean formula ϕ can be mapped to a normal logic program P_ϕ such that the stable models of P_ϕ correspond precisely to the classical, Boolean models of ϕ , e.g. for formulas in clausal form [28] or even of arbitrary form [37].

6 RELATED WORK

The Situation Calculus and default reasoning have been combined before, albeit differently. Lee and Palla [16] present a reformulation of Situation Calculus within the stable model semantics, which allows to solve the Frame and Qualification Problems for Lin's [21] causal theories, but they do not consider sensing and knowledge. Pagnucco et al. [29] describe an account of belief change in the Situation Calculus based on Default Logic. Strass and Thielscher [38] formalize \mathcal{D} , a language for default reasoning about action and change and descendent of Gelfond and Lifschitz' [8] language \mathcal{A} , and implement it using ASP. Ryan moreover [35] presents an ASP-based interpreter for a fragment of the GOLOG programming language, but also only for the objective case.

Finally, much more work than what we have discussed here has been done on relating different non-monotonic formalisms to one another. For example, Marek and Truszczyński [24] show the correspondence between ASP and the modal logic K45. The close relation of only-knowing to classical modal systems of belief is further studied by Lakemeyer and Levesque [15]. Earlier works of integrating modal epistemic logics with default reasoning are by Lin and Shoham [23], Lifschitz [20], Amati et al. [1], and Denecker et al. [5], but either require Autoepistemic Logic to be treated differently from Default Logic or rely on a very complex semantics with fixed-point constructions.

7 CONCLUSION

We presented a formalization of knowledge-based agents with defaults based on variants of Reiter's regression and Levesque and Lakemeyer's Representation Theorem, thus laying the foundation for an implementation where backend reasoning tasks can be outsourced to off-the-shelf default reasoners.

Apart from coming up with an actual implementation and empirical evaluation thereof, there are many other possible lines for future

⁷ <http://www.cs.uni-potsdam.de/clasp>

work. One particularly promising would be to integrate our results with [14] where yet another variant of only-knowing is presented that correctly captures what it would mean to *progress* [22] a knowledge base with defaults (though only Moore-style) through an action in the presence of sensing. The formalization of [12] we have used here is problematic in that respect as it may cause inconsistencies between the default conclusions holding after an action and the facts to be forgotten in the process of progression, which is why we needed to reduce everything to the static case by means of regression. The proposed solution is to introduce additional modalities that allow to distinguish default conclusion from “hard” facts, and it would be interesting to combine our results with theirs to come up with a progression-based variant of knowledge-based agents with defaults.

ACKNOWLEDGEMENTS

This work was supported by the German Research Foundation (DFG) research unit FOR 1513 on Hybrid Reasoning for Intelligent Systems, project A1.

REFERENCES

- [1] Gianni Amati, Luigia Carlucci Aiello, and Fiora Pirri, ‘Definability and commonsense reasoning’, *Artificial Intelligence*, **93**, 169–199, (1997).
- [2] Pawel Cholewinski, V. Wiktor Marek, Mirosław Truszczyński, and Artur Mikitiuk, ‘Computing with default logic’, *Artificial Intelligence*, **112**(1-2), 105–146, (1999).
- [3] Jens Claßen and Gerhard Lakemeyer, ‘Foundations for knowledge-based programs using \mathcal{ES} ’, in *Proceedings of the Tenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, eds., Patrick Doherty, John Mylopoulos, and Christopher A. Welty, pp. 318–328. AAAI Press, (2006).
- [4] Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque, ‘ConGolog, a concurrent programming language based on the situation calculus’, *Artificial Intelligence*, **121**(1–2), 109–169, (2000).
- [5] Marc Denecker, V. Wiktor Marek, and Mirosław Truszczyński, ‘Uniform semantic treatment of default and autoepistemic logics’, *Artificial Intelligence*, **143**(1), 79–122, (2003).
- [6] Michael Gelfond, ‘Answer sets’, *Foundations of Artificial Intelligence*, **3**, 285–316, (2008).
- [7] Michael Gelfond and Vladimir Lifschitz, ‘Classical negation in logic programs and disjunctive databases’, *New Generation Computing*, **9**(3/4), 365–386, (1991).
- [8] Michael Gelfond and Vladimir Lifschitz, ‘Representing action and change by logic programs’, *Journal of Logic Programming*, **17**(2), 301–321, (1993).
- [9] Kurt Konolige, ‘On the relation between default and autoepistemic logic’, *Artificial Intelligence*, **35**(3), 343–382, (1988).
- [10] Gerhard Lakemeyer, ‘The situation calculus: A case for modal logic’, *Journal of Logic, Language and Information*, **19**(4), 431–450, (2010).
- [11] Gerhard Lakemeyer and Hector J. Levesque, ‘Situations, si! situation terms, no!’, in *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004)*, eds., Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, pp. 516–526. AAAI Press, (2004).
- [12] Gerhard Lakemeyer and Hector J. Levesque, ‘Only-knowing: Taking it beyond autoepistemic reasoning’, in *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, eds., Manuela M. Veloso and Subbarao Kambhampati, pp. 633–638. AAAI Press, (2005).
- [13] Gerhard Lakemeyer and Hector J. Levesque, ‘Towards an axiom system for default logic’, in *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 2006)*, eds., Yolanda Gil and Raymond J. Mooney, pp. 263–268. AAAI Press, (2006).
- [14] Gerhard Lakemeyer and Hector J. Levesque, ‘A semantical account of progression in the presence of defaults’, in *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, eds., Alexander Borgida, Vinay K. Chaudhri, Paolo Giorgini, and Eric S. K. Yu, volume 5600 of *Lecture Notes in Computer Science*, pp. 82–98. Springer-Verlag, (2009).
- [15] Gerhard Lakemeyer and Hector J. Levesque, ‘Only-knowing meets nonmonotonic modal logic’, in *Proceedings of the Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2012)*, eds., Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith. AAAI Press, (2012).
- [16] Joohyung Lee and Ravi Palla, ‘Reformulating the situation calculus and the event calculus in the general theory of stable models and in answer set programming’, *Journal of Artificial Intelligence Research*, **43**, 571–620, (2012).
- [17] Hector J. Levesque, ‘All I know: a study in autoepistemic logic’, *Artificial Intelligence*, **42**(2), 263–309, (1990).
- [18] Hector J. Levesque and Gerhard Lakemeyer, *The Logic of Knowledge Bases*, MIT Press, 2001.
- [19] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl, ‘GOLOG: A logic programming language for dynamic domains’, *Journal of Logic Programming*, **31**(1–3), 59–83, (1997).
- [20] Vladimir Lifschitz, ‘Minimal belief and negation as failure’, *Artificial Intelligence*, **70**(1-2), 53–72, (1994).
- [21] Fangzhen Lin, ‘Embracing causality in specifying the indirect effects of actions’, in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1995)*, ed., Chris S. Mellish, pp. 1985–1993. Morgan Kaufmann Publishers Inc., (1995).
- [22] Fangzhen Lin and Raymond Reiter, ‘How to progress a database’, *Artificial Intelligence*, **92**(1–2), 131–167, (1997).
- [23] Fangzhen Lin and Yoav Shoham, ‘Epistemic semantics for fixed-points nonmonotonic logics’, in *Proceedings of the Third Conference on Theoretical Aspects on Reasoning about Knowledge (TARK 1990)*, ed., Rohit Parikh, pp. 111–120. Morgan Kaufmann Publishers Inc., (1990).
- [24] Wiktor W. Marek and Mirosław Truszczyński, *Nonmonotonic logic: context-dependent reasoning*, Springer-Verlag, 1993.
- [25] John McCarthy and Patrick Hayes, ‘Some philosophical problems from the standpoint of artificial intelligence’, in *Machine Intelligence 4*, eds., B. Meltzer and D. Michie, 463–502, American Elsevier, New York, (1969).
- [26] Robert C. Moore, ‘Semantical considerations on nonmonotonic logic’, *Artificial Intelligence*, **25**(1), 75–94, (1985).
- [27] Pascal Nicolas and Torsten Schaub, ‘The XRay system: An implementation platform for local query-answering in default logics’, in *Applications of Uncertainty Formalisms*, eds., Anthony Hunter and Simon Parsons, volume 1455 of *Lecture Notes in Computer Science*, pp. 354–378. Springer-Verlag, (1998).
- [28] Ilkka Niemelä, ‘Logic programs with stable model semantics as a constraint programming paradigm’, *Annals of Mathematics and Artificial Intelligence*, **25**(3–4), 241–273, (1999).
- [29] Maurice Pagnucco, David Rajaratnam, Hannes Strass, and Michael Thielscher, ‘Implementing belief change in the situation calculus and an application’, in *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013)*, eds., Pedro Cabalar and Tran Cao Son, volume 8148 of *Lecture Notes in Computer Science*, pp. 439–451. Springer-Verlag, (2013).
- [30] Raymond Reiter, ‘A logic for default reasoning’, *Artificial Intelligence*, **13**(1–2), 81–132, (1980).
- [31] Raymond Reiter, ‘The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression’, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, 359–380, (1991).
- [32] Raymond Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*, MIT Press, 2001.
- [33] Raymond Reiter, ‘On knowledge-based programming with sensing in the situation calculus’, *ACM Transactions on Computational Logic*, **2**(4), 433–457, (2001).
- [34] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 3rd edn., 2009.
- [35] Malcolm Ryan, ‘Efficiently implementing GOLOG with answer set programming’, in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, eds., Carla E. Brodley and Peter Stone, pp. 2352–2357. AAAI Press, (2014).
- [36] Richard B. Scherl and Hector J. Levesque, ‘Knowledge, action, and the frame problem’, *Artificial Intelligence*, **144**(1–2), 1–39, (2003).
- [37] Igor Stéphan, Benoit Da Mota, and Pascal Nicolas, ‘From (quantified) boolean formulae to answer set programming’, *Journal of Logic and Computation*, **19**(4), 565–590, (2009).
- [38] Hannes Strass and Michael Thielscher, ‘A language for default reason-

ing about actions', in *Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz*, eds., Esra Erdem, Joohyung Lee, Yuliya Lierler, and David Pearce, volume 7265 of *Lecture Notes in Computer Science*, pp. 527–542. Springer-Verlag, (2012).

Solving Multi-Agent Knapsack Problems Using Incremental Approval Voting

Nawal Benabbou and Patrice Perny¹

Abstract. In this paper, we study approval voting for multi-agent knapsack problems under incomplete preference information. The agents consider the same set of feasible knapsacks, implicitly defined by a budget constraint, but they possibly diverge in the utilities they attach to items. Individual utilities being difficult to assess precisely and to compare, we collect approval statements on knapsacks from the agents with the aim of determining the optimal solutions by approval voting. We first propose a search procedure based on mixed-integer programming to explore the space of utilities compatible with the known part of preferences in order to determine or approximate the set of possible approval winners. Then, we propose an incremental procedure combining preference elicitation and search in order to determine the set of approval winners without requiring the full elicitation of the agents' preferences. Finally, the practical efficiency of these procedures is illustrated by various numerical tests.

1 INTRODUCTION

Collective decision making on a combinatorial domain appears in various contexts such as investment planning, resource allocation or group configuration. Due to strategic aspects often surrounding group decision-making and the possible divergences in individual values, developing formal methods and tools for modeling preferences and solving multi-agent combinatorial optimization problems is a critical issue. This has motivated a lot of work in the recent years, in the field of computational social choice [9]. We focus here on the multi-agent knapsack problem which consists of determining, given a finite set of items, a subset of maximal utility under a budget constraint. This is a standard example of combinatorial problem with many potential applications such as project selection, portfolio management or committee election, see e.g. [22, 16, 27, 31] for examples of recent contributions in AI.

In combinatorial optimization problems, the agents cannot be expected to provide extensive preference models. Compact representations are needed to handle individual and collective preferences. Usually, in knapsack problems, preference over subsets of items are represented by additive utility functions. More precisely, the utility of a subset of items for an agent is defined as the sum of the utilities of its elements. Individual utilities are difficult to assess, especially on a combinatorial domain. Although the elicitation task is simplified when utilities are decomposable, elicitation methods based on systematic pairwise comparisons are practically unfeasible due to the large amount of feasible subsets and their implicit definition. Hence we are interested in designing incremental elicitation procedures, in

which preference queries are selected iteratively, to be as informative as possible at every step, so as to progressively reduce the set of admissible utility profiles until the set of optimal knapsacks can be determined. This approach has been successfully used in AI for additive utility elicitation on explicit sets [12, 33, 7], but also on combinatorial solution spaces [17, 3, 4].

Remark that, even if numerical representations of individual preferences are accessible under the form of utility functions, they are generally constructed independently for each agent. Hence, it is unlikely that such representations allow the welfare of individuals to be compared. In this context, the definition of a social utility as the sum of individual utilities (utilitarianism), for example, would be meaningless. Assuming that utilities of items are expressed on the same scale or that utilities are normalized would not be sufficient to overcome the problem, as shown by the following example:

Example 1. Consider a multi-agent knapsack problem involving 3 items and 2 agents with utilities: $u^1 = u_1^1x_1 + u_2^1x_2 + u_3^1x_3$ and $u^2 = u_1^2x_1 + u_2^2x_2 + u_3^2x_3$ to be maximized under the constraint $x_1 + x_2 + x_3 \leq 2$, where $x_i \in \{0, 1\}$, $i = 1, 2, 3$, are the decision variables and u_j^i represents the utility of item j for agent i . This problem could appear to elect a committee of size 2, given 3 candidates and 2 voters. Assume that individual preference orders over committees have been elicited, and are equal to $\{2, 3\} \succ_1 \{1, 3\} \succ_1 \{1, 2\}$ and $\{1, 2\} \succ_2 \{1, 3\} \succ_2 \{2, 3\}$ for agents 1 and 2 respectively. One possible numerical representation of these preferences (using the same utility scale for the two agents) is given by: $(u_1^1, u_2^1, u_3^1) = (1, 2, 4)$ and $(u_1^2, u_2^2, u_3^2) = (4, 2, 1)$ which leads to the following utilities for solutions of size 2:

	{1, 2}	{2, 3}	{1, 3}
u^1	3	6	5
u^2	6	3	5

Note that these individual values are consistent with preference orders \succ_1 and \succ_2 . Now, if we are utilitarian, we could be tempted to deduce that $\{1, 3\}$ is the optimal knapsack because it maximizes the total utility ($5+5 = 10$). However, such a conclusion would be meaningless, it is only due to the particular numerical representation chosen for individual utilities. Let us change the initial numerical scale by replacing numbers $(1, 2, 4)$ by $(0, 3, 4)$. In this case we obtain two new utility functions characterized by $(u_1^1, u_2^1, u_3^1) = (0, 3, 4)$ and $(u_1^2, u_2^2, u_3^2) = (4, 3, 0)$ which leads to the following utilities for solutions of size 2:

	{1, 2}	{2, 3}	{1, 3}
u^1	3	7	4
u^2	7	3	4

¹ Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 Place Jussieu, 75005 Paris, France, email:name.surname@lip6.fr

Note that these new individual values are still consistent with preference orders \succ_1 and \succ_2 . Yet, with the same utilitarian principle, we should admit now that $\{1, 3\}$ is the least preferred knapsack. Therefore, choosing the solution maximizing the sum of individual utilities would not be a good procedure. It would merely be a consequence of arbitrary choices of numerical representations of preferences rather than a robust conclusion derived from the observed preference profile. Note that the problem persists if we normalize u^1 and u^2 utilities to obtain value 1 for set $\{1, 2, 3\}$. Other examples could be found for other aggregators (e.g. the minimum for an egalitarian aggregation).

In order to be able to compare the solutions of a knapsack problem when individual utility scales are not commensurate and/or not rich enough to allow the construction of a social utility, it seems natural to resort to a voting rule. The main advantage of a voting rule is indeed to perform an ordinal aggregation procedure. There is no need to know how the welfare of individuals should be compared, we only need to elicit individual preference orders. Some recent contributions consider voting rules in a very different perspective, for example, by investigating their ability to approximate the winner with respect to the utilitarian criterion [29, 11, 6].

Preference elicitation can be performed incrementally so as to determine the winner with a reduced amount of preference queries. Various incremental elicitation procedures have been proposed and studied in the context of single-winner elections with incomplete preferences [18, 23, 14]. In this setting, several contributions study the determination of possible and necessary winners from a partial preference profile, e.g., [20, 34, 21, 15], when the set of candidates is defined explicitly.

In knapsack problems however, solutions are numerous and defined implicitly, which is an additional challenge for the winner determination. This explains the current interest for incremental voting procedures on combinatorial domains and the purpose of this paper. Our aim here is to propose an incremental voting rule in which individual preferences are progressively revealed until a collective decision can be made, and to apply this procedure on the multi-agent knapsack problem, taking advantage of the fact that individual preferences are representable by additive utilities.

It is important to note that implementing a voting rule is not in contradiction with the representation of individual values by utilities. Individual utility functions are indeed seen as convenient representations of individual preference orders, and their use will significantly contribute to relieve the preference elicitation burden. In the standard knapsack problem, due to the linearity of preferences, the set of all preference orders compatible with a given partial order can be characterized by a convex polyhedron in the utility space. This makes it possible to resort to mathematical programming to explore all possible completions of any partially known preference profile, but also to look for possible winners, and to develop an efficient incremental elicitation procedure for the determination of all winners, as it will be seen later in the paper.

Implementing a voting rule for the knapsack problem with a partially specified preference profile could be related to multi-winner voting rules studied in the field of computational social choice. Most approaches recently proposed for multi-winner elections assume that individual preferences over items are sufficient to explain preference over subsets because they derive satisfaction from their most preferred candidate (see e.g., [13, 26, 28, 30, 25, 22, 5, 16, 32], and see [24] for incremental elicitation of voter preferences). This assumption is well-suited to the election of representatives. However, for any agent, it may happen that the selection of the most preferred candi-

date is not sufficient to counterbalance the presence of multiple least preferred candidates in the elected committee. This limitation also applies to the selection of items in multi-agent knapsack problems.

In this paper, we focus on approval voting because this is a simple rule that can be decisive even if only a part of the preference profile is known. Approval voting is the voting method which allows each agent to approve of (vote for) as many solutions as she wishes, and the solution with the most approval votes is the winner of the election [8]. In approval voting, we only need to learn, for every agent, which are the approved or disapproved subsets, so as to elect a solution receiving the maximal support. We therefore investigate incremental procedures for approval voting and their application to the knapsack problem. This work differs from multi-winner approval voting [19, 2, 1] which only collects approval statements over items instead of feasible subsets of items.

The paper is organized as follows: we introduce the multi-agent knapsack problem in Section 2 and study computational issues for this problem. In Section 3, we propose a search procedure for the determination of the possible winners. Then, an incremental voting procedure to determine the set of approval winners is proposed in Section 4. Finally, numerical tests are provided in Section 5.

2 THE GENERAL FRAMEWORK

We consider a collective decision problem where a set of agents $N = \{1, \dots, n\}$ has to jointly select a set of items (e.g., candidates, projects, objects) in a set $P = \{1, \dots, p\}$. Any subset of items can be represented by a solution vector $x = (x_1, \dots, x_p) \in \{0, 1\}^P$ where $x_j = 1$ if item j is in the subset and $x_j = 0$ otherwise. Some linear constraints on variables $x_j, j \in P$, are imposed to define the admissible solution vectors. For instance, one may want to impose cardinality constraints to control the size of the subset and/or to ensure gender parity in the elected committee; there may also exist budget constraints (e.g., when the decision is subject to a maximum total cost) or capacity constraints as in knapsack problems, making some subsets of items unfeasible. For the simplicity of the presentation, we will only consider the standard knapsack constraint of the form $\sum_{j \in P} w_j x_j \leq W$ where w_j is the (positive) weight of item j and W is a positive value representing the maximum total weight; the set of feasible solutions will be denoted by \mathcal{X} in the sequel. Our purpose and the algorithms proposed in the paper also apply when additional (linear) feasibility constraints are considered.

We assume that the preferences of agent $i, i \in N$, can be represented by a function $u^i : \{0, 1\}^P \rightarrow \mathbb{R}$ measuring the overall utility of any solution. Hence, given two solutions x, y representing two subsets of items, x is at least as good as y for agent $i \in N$ whenever $u^i(x) \geq u^i(y)$. Here $u^i(x) = \sum_{j \in P} u_j^i x_j$, where $u_j^i \in \mathbb{R}$ represents the utility of item j for agent i . The profile (u^1, \dots, u^n) of utility functions will be denoted by u . Note also that u^i , as a numerical representation of a preference order, is generally not unique and any transform preserving inequalities of type $u^i(x) \geq u^i(y)$ for all solutions x, y could be considered as well.

Nevertheless, numerical representations of individual preferences by utility functions $u^i, i \in N$, can be used in approval voting. Under the assumption that individual preferences are represented by utility functions $u^i, i \in N$, a solution x is approved by agent i if and only if $u^i(x) \geq \delta^i$ where $\delta^i \in \mathbb{R}$ is an approval (or utility) threshold that separates approved and non-approved solutions. The profile of thresholds $(\delta^1, \dots, \delta^n)$ will be denoted by δ in the sequel. The pair (u, δ) characterizes approved and non-approved solutions for all agents and enables the computation of approval

scores for any feasible solution x . This approval score is given by $f(x, u, \delta) = |\{i \in N, u^i(x) \geq \delta^i\}|$, and the winner of the election is a feasible solution maximizing this score (various tie-breaking rules can be considered). In this paper, we consider the approval multi-agent knapsack problem defined as follows:

APPROVAL MULTI-AGENT KNAPSACK PROBLEM (AMKP)

Input: A finite set P of items; a positive integer W ; for each $j \in P$, a weight w_j ; a finite set N of agents; a positive integer K ; for each $i \in N$, an approval threshold δ^i , for each $j \in P$, a utility value u_j^i .

Question: Is there a subset $X \subseteq P$ such that $\sum_{j \in X} w_j \leq W$ and $|\{i \in N, \sum_{j \in X} u_j^i \geq \delta^i\}| \geq K$?

Proposition 1. *AMKP is NP-complete.*

Proof. The proof is quite straightforward due to a simple reduction from the knapsack decision problem. Testing the existence of an admissible knapsack having a utility greater or equal to a given value K' is indeed equivalent to solving an instance of the AMKP problem involving a single agent with the same utilities over items, an approval threshold equal to K' and with $K = 1$. \square

There are obvious tractable cases when W is a constant and considering integer weights, for example, committee election problems ($w_j = 1$ for all $j \in P$) such that the committee size is a constant (W) specified explicitly. Indeed, we have:

Proposition 2. *When W is constant and $w_j \in \mathbb{N} \setminus \{0\}$ for all $j \in P$, AMKP is in \mathcal{P} .*

Proof. Since w_j is a non-zero positive integer for all $j \in P$, then we know that all feasible knapsacks necessarily include at most W items. The number of knapsacks of size at most W is equal to:

$$\sum_{k=0}^W \binom{p}{k} = \sum_{k=0}^W \frac{p!}{k!(p-k)!}$$

This number is obviously polynomial in p when W is a constant. Therefore, the result can be easily obtained by considering the following naive procedure: for all sets $X \subseteq P$ of size at most W , we test whether $\sum_{j \in X} w_j \leq W$ (feasibility condition), and if the test succeeds, we compute $|\{i \in N, \sum_{j \in X} u_j^i \geq \delta^i\}|$ (approval score). This procedure is polynomial in p and n when W is a constant. \square

Proposition 1 shows that finding the knapsack maximizing the approval score is NP-hard in the general case. Moreover, well-known pseudo-polynomial solution methods proposed for the standard knapsack problem (based on dynamic programming) are not easily transposable to the approval winner determination problem, as shown in Example 2.

Example 2. *Consider a collective decision problem where 2 agents have to choose 2 representatives from a pool of 3 candidates, i.e. $N = \{1, 2\}$ and $P = \{1, 2, 3\}$. Assume that utilities and approval thresholds are the following:*

u_1^1	u_2^1	u_3^1	δ^1	u_1^2	u_2^2	u_3^2	δ^2
0.5	0.3	0.2	0.5	0.1	0.5	0.3	0.7

In this case, we have $f(\{1\}, u, \delta) = 1 > 0 = f(\{2\}, u, \delta)$ but $f(\{1, 3\}, u, \delta) = 1 < 2 = f(\{2, 3\}, u, \delta)$. We observe a preference reversal because $\{1\}$ is preferred to $\{2\}$ whereas $\{2, 3\}$ is preferred to $\{1, 3\}$. Thus, preferences induced by the approval score are not additive with respect to union with disjoint items. This precludes to construct the optimal knapsack from optimal subsets of items.

Nevertheless, the winner can be obtained by solving the following mixed integer program (MIP₁):

$$\begin{aligned} \max \quad & \sum_{i \in N} a^i \\ \text{s.t.} \quad & \begin{cases} \sum_{j \in P} u_j^i x_j - \delta^i \geq M(a^i - 1), \forall i \in N \\ \sum_{j \in P} w_j x_j \leq W \\ x_j \in \{0, 1\}, a^i \in \{0, 1\}, \forall i \in N, \forall j \in P \end{cases} \end{aligned}$$

In this program, M is a constant greater than $\max\{\delta^i - u^i(x), i \in N\}$ that allows the introduction of boolean variables a^i which will be equal to 1 if and only if agent i approves solution x . Moreover, the second Equation is the knapsack constraint.

However, in practice, the full elicitation of individual utilities and approval thresholds is too expensive. Usually, we can observe simple preference statements of type “I prefer solution x to solution y ”, and in the case of approval voting, “I approve solution x ”, or “I don’t approve solution y ”. These preference statements enable to restrict the sets of possible utility functions but generally do not allow to derive a precise utility function for each agent (see Example 1). Instead, uncertainty sets representing all possible utility functions compatible with the preference information obtained so far must be considered. The same observation applies to approval thresholds. Under utility uncertainty, we study now the determination of possible winners.

3 POSSIBLE WINNERS DETERMINATION

In this section, we propose an algorithm that enables to compute the set of possible approval winners given some partial knowledge of the agents’ preferences. More precisely, the input of the algorithm consists of three sets of preference information for each $i \in N$: a set A^i (resp. \bar{A}^i) of solutions that are known to be approved (resp. not approved) by agent i , and a set \mathcal{P}^i of pairs (y, z) such that solution y is known to be preferred to solution z by agent i . The elements of \mathcal{P}^i are not explicit approval statements, but they can be used to derive new positive or negative approval statements from those included in A^i and \bar{A}^i .

For each agent $i \in N$, let U^i (resp. Δ^i) denotes the set of utility functions (resp. approval thresholds) compatible with the available preference statements A^i, \bar{A}^i and \mathcal{P}^i . Formally, (U^i, Δ^i) is the set of all pairs (u^i, δ^i) such that:

$$\forall y \in A^i, u^i(y) \geq \delta^i; \forall y \in \bar{A}^i, u^i(y) < \delta^i; \forall (y, z) \in \mathcal{P}^i, u^i(y) \geq u^i(z)$$

where u^i is of the form $u^i(x) = \sum_{j \in P} u_j^i x_j$ and $\delta^i \in \mathbb{R}$. Let U (resp. Δ) be the cartesian product $U^1 \times \dots \times U^n$ (resp. $\Delta^1 \times \dots \times \Delta^n$). Given such uncertainty sets, the set of possible approval winners is defined as follows:

Definition 1. *The set $PW(\mathcal{X}, U, \Delta)$ of possible approval winners is the set of all solutions $x \in \mathcal{X}$ that maximize the approval score for some utility profile $u \in U$ and some approval threshold vector $\delta \in \Delta$. More formally: $PW(\mathcal{X}, U, \Delta) = \bigcup_{u \in U, \delta \in \Delta} \arg \max_{x \in \mathcal{X}} f(x, u, \delta)$.*

Recall that Example 2 shows that standard dynamic programming procedures cannot be used to determine the approval winners when utilities and approval thresholds are known. This difficulty remains when utilities and/or approval thresholds are partially known.

The branch and bound approach is the most commonly used tool for solving NP-hard optimization problems. We propose here a branch and bound procedure to compute the set $\text{PW}(\mathcal{X}, U, \Delta)$, where nodes of the search tree represent partial instances of the decision variable vector $x = (x_1, \dots, x_p)$. More precisely, each node η of the tree is characterized by a pair (P_η^0, P_η^1) where $P_\eta^k = \{j \in P, x_j = k\}, k = 0, 1$. Let $P_\eta = P \setminus (P_\eta^0 \cup P_\eta^1)$ denote the set of all undecided variables at node η . Thus, each node η is associated with a region of the solution space as follows: solution $x = (x_1, \dots, x_p)$ is attached to node η if and only if $x_j = k$ for all $j \in P_\eta^k, k = 0, 1$. The set of feasible solutions attached to node η is denoted by S_η hereafter. The main features of our search procedure are the following:

Initialization. Using a heuristic, a branch and bound procedure determines some feasible solutions before performing the search so as to define an initial bound on candidate solutions.

In order to obtain such a bounding set for the knapsack problem, denoted by S_0 hereafter, we propose to initially ask the agents to rank all the items by preference order. Let $r_i(j)$ denote the rank of item j in the preference order provided by agent i . We can define the score $\alpha^i(j) = \frac{p-r_i(j)}{w_j}$ for each agent $i \in N$ and each item $j \in P$ representing the tradeoff achieved between preference and weight. Hence, for each agent i , a ‘‘good’’ solution to the knapsack problem can be obtained by a greedy algorithm selecting items one by one, by decreasing order with respect to scoring function α^i , skipping elements whose weight is greater than the residual weight capacity. The resulting solution is inserted in S_0 for initialization because it represents a good solution from the point of view of agent i . This process is repeated for all agents $i \in N$.

Moreover, a similar procedure is used with the average scoring function defined by $\alpha(j) = 1/n \sum_{i=1}^n \alpha^i(j)$, to complete S_0 with a solution which is likely to be more consensual. This solution will be denoted by \bar{s} in the sequel.

Evaluation and pruning. Let S be the set of solutions found so far (initially $S = S_0$) and O be the current set of nodes to be explored. Our pruning rule is based on the notion of setwise max regret defined as follows: the setwise max regret $SR(A, B, U, \Delta)$ of a set $A \subseteq \mathcal{X}$ with respect to a set $B \subseteq \mathcal{X}$ is the maximal feasible approval score difference between the best solution in B and the best solution in A . More formally:

$$SR(A, B, U, \Delta) = \max_{u \in U, \delta \in \Delta} \left\{ \max_{b \in B} f(b, u, \delta) - \max_{a \in A} f(a, u, \delta) \right\}$$

If $SR(A, B, U, \Delta) < 0$, then we know that B does not contain any possible approval winner; it indeed induces that, for all solutions $b \in B$, for all $u \in U$ and for all $\delta \in \Delta$, there exists $a \in A$ such that $f(b, u, \delta) < f(a, u, \delta)$. Therefore, we propose to prune a node $\eta \in O$ if the setwise max regret $SR(S, S_\eta, U, \Delta)$ of set S with respect to set S_η is strictly negative. Note that $SR(S, S_\eta, U, \Delta) = \max_{x \in S_\eta} \max_{u \in U, \delta \in \Delta} \min_{s \in S} \{f(x, u, \delta) - f(s, u, \delta)\}$. This alternative formulation of setwise max regrets enables to compute $SR(S, S_\eta, U, \Delta)$ as the optimal value of the mixed-integer quadratic program (denoted by MIQP_η) given in Figure 1. In this program, $\xi > 0$ is an arbitrary small value enabling to model strict inequalities. Equations (2e-2g) enable to restrict utility functions and approval thresholds to those compatible with the available preference information. Then, since the preferences of agent i , for any $i \in N$, are invariant by positive affine transformations jointly applied to function u^i and threshold δ^i , we can assume without loss of generality

$$\begin{aligned} \max \quad & t \\ \text{s.t.} \quad & \left\{ \begin{array}{l} t \leq \sum_{i \in N} a^i - \sum_{i \in N} a_s^i, \forall s \in S \quad (2a) \\ \sum_{j \in P_\eta^1} u_j^i + \sum_{j \in P_\eta} u_j^i x_j - \delta^i \geq M(a^i - 1), \forall i \in N \quad (2b) \\ \sum_{j \in P} u_j^i s_j - \delta^i + \xi \leq M a_s^i, \forall s \in S, \forall i \in N \quad (2c) \\ \sum_{j \in P_\eta} w_j x_j + \sum_{j \in P_\eta^1} w_j \leq W \\ \sum_{j \in P} u_j^i = M, \forall i \in N \quad (2d) \\ \sum_{j \in P} u_j^i y_j \geq \delta^i, \forall i \in N, \forall y \in A^i \quad (2e) \\ \sum_{j \in P} u_j^i y_j \leq \delta^i - \xi, \forall i \in N, \forall y \in \bar{A}^i \quad (2f) \\ \sum_{j \in P} u_j^i y_j \geq \sum_{j \in P} u_j^i z_j, \forall i \in N, \forall (y, z) \in \mathcal{P}^i \quad (2g) \\ x_j \in \{0, 1\}, \forall i \in N, \forall j \in P_\eta \\ a^i \in \{0, 1\}, a_s^i \in \{0, 1\}, \forall i \in N, \forall s \in S \\ u_j^i \geq 0, \delta^i \geq 0, \forall i \in N, \forall j \in P \end{array} \right. \end{aligned}$$

Figure 1. MIQP_η

that utilities are positive and bounded above by a constant $M > 0$ (see Equation (2d)). Moreover, a^i is a boolean variable that will be equal to 1 iff agent i approves solution x , and a_s^i is a boolean variable that will be equal to 1 iff agent i approves solution $s, s \in S$. Finally, Equation (2a) introduces variable $t \in \mathbb{R}$ representing the smallest approval score difference between solution x and a solution $s, s \in S$.

Note that constraints given in Equation (2b) include quadratic terms of type $u_j^i x_j, j \in P_\eta$, since u_j^i are also variables of the optimization problem. In order to linearize these constraints, we introduce positive variables $v_j^i, i \in N, j \in P_\eta$, representing the product $u_j^i x_j$ and Equation (2b) is replaced by the following constraints:

$$\left\{ \begin{array}{l} \sum_{j \in P_\eta^1} u_j^i + \sum_{j \in P_\eta} v_j^i - \delta^i \geq M(a^i - 1), \forall i \in N \\ v_j^i \leq u_j^i, \forall i \in N, \forall j \in P_\eta \\ v_j^i \leq M x_j, \forall i \in N, \forall j \in P_\eta \\ v_j^i - u_j^i \geq M(x_j - 1), \forall i \in N, \forall j \in P_\eta \end{array} \right.$$

The resulting mixed-integer linear program will be denoted by MIP_η .

Branching. Setwise max regrets $SR(S, S_\eta, U, \Delta)$ available for all nodes $\eta \in O$ are also used to select the next node to be explored. More precisely, we select here a node $\eta \in O$ which maximizes $SR(S, S_\eta, U, \Delta)$. This branching strategy aims to maximally improve the current solution set S . The optimal solution of MIP_η indeed maximizes the gap $f(x, u, \delta) - \max_{s \in S} f(s, u, \delta)$ over all $u \in U$, all $\delta \in \Delta$ and all $x \in \bigcup_{\eta' \in O} S_{\eta'}$. Then, S_η is split in two by considering possible instantiations of a variable $x_j, j \in P_\eta$ chosen among the variables equal to 1 in the optimal solution of MIP_η .

Filtering. As we will see in Proposition 3, the proposed Branch and Bound outputs, in general, a superset of the set of possible approval

winners. To remove undesirable elements, we use a final filtering process, named FILTER hereafter, which iteratively deletes all solutions $s' \in S$ such that $SR(S \setminus \{s'\}, \{s'\}, U, \Delta) < 0$, using a simplified version of MIP_η .

The algorithm implementing these principles is referred to AS (Approval-based Search) in the sequel and it is summarized by Algorithm 1.

Algorithm 1: Approval-based Search

Input: S_0 : initial solutions; U, Δ : uncertainty sets
Output: $PW(\mathcal{X}, U, \Delta)$: the set of possible approval winners

```

1  $S \leftarrow S_0$ 
2  $\eta \leftarrow [\emptyset, \emptyset]$ 
3  $O \leftarrow \{\eta\}$ 
4 while  $O \neq \emptyset$  do
5   Select a node  $\eta$  in  $\arg \max_{\eta' \in O} SR(S, S_{\eta'}, U, \Delta)$ 
6   if  $P_\eta = \emptyset$  then
7      $S \leftarrow S \cup S_\eta$ 
8   else
9     Select  $j \in P_\eta$  s.t.  $x_j = 1$  in the optimal solution of  $MIP_\eta$ 
10    Generate  $\eta^0 = [P_\eta^0 \cup \{j\}, P_\eta^1]$  and  $\eta^1 = [P_\eta^0, P_\eta^1 \cup \{j\}]$ 
11    forall  $\eta' \in \{\eta^0, \eta^1\}$  do
12      if  $S_{\eta'} \neq \emptyset$  and  $SR(S, S_{\eta'}, U, \Delta) \geq 0$  then
13         $O \leftarrow O \cup \{\eta'\}$ 
14      end
15    end
16  end
17   $O \leftarrow O \setminus \{\eta\}$ 
18 end
19  $S \leftarrow FILTER(S)$ 
20 return  $S$ 

```

This algorithm is justified by the following proposition:

Proposition 3. *AS returns the set $PW(\mathcal{X}, U, \Delta)$.*

Proof. Since, the pruning rule only prunes nodes including no possible approval winner (by definition of setwise max regrets), we know that S is a superset of $PW(\mathcal{X}, U, \Delta)$ at the end of the while loop. Let x be an element inserted in S such that $x \notin PW(\mathcal{X}, U, \Delta)$, if it exists. Since x is not a possible winner, we know that, for all $u \in U$ and for all $\delta \in \Delta$, there exists $x' \in PW(\mathcal{X}, U, \Delta) \subseteq S$ such that $f(x, u, \delta) < f(x', u, \delta)$. Therefore, x is necessarily removed from S during the filtering (by definition of the filtering process). \square

Depending on the uncertainty sets (U and Δ) implicitly defined by the available preference information, possible approval winners might be too numerous to be enumerated efficiently. In order to save time, one may be interested in approximating the set of possible approval winners with performance guarantees. We propose below a variant of Algorithm AS for approximating possible winners with some guarantees on the quality of the output.

Approximation. Given a constant $\varepsilon > 0$, a set $X \subseteq \mathcal{X}$ is an $(1 + \varepsilon)$ -approximation of the set of possible winners if, for all $x \in \mathcal{X}$, all $u \in U$ and all $\delta \in \Delta$, there exists $x' \in X$ such that $f(x, u, \delta) \leq (1 + \varepsilon)f(x', u, \delta)$. In order to compute an $(1 + \varepsilon)$ -approximation of the set of possible winners, we introduce an approximate version of the setwise max regret. The setwise max ε -regret $SR_\varepsilon(A, B, U, \Delta)$ of a set $A \subseteq \mathcal{X}$ with respect to a set $B \subseteq \mathcal{X}$ is defined as follows:

$$SR_\varepsilon(A, B, U, \Delta) = \max_{u \in U, \delta \in \Delta} \left\{ \max_{b \in B} f(b, u, \delta) - \max_{a \in A} (1 + \varepsilon)f(a, u, \delta) \right\}$$

If $SR_\varepsilon(A, B, U, \Delta) \leq 0$, then we know that, for all $b \in B$, all $u \in U$ and all $\delta \in \Delta$, there exists $a \in A$ such that $f(b, u, \delta) \leq (1 + \varepsilon)f(a, u, \delta)$. Therefore, we propose a variant of AS where a node $\eta \in O$ is pruned if $SR_\varepsilon(S, S_\eta, U, \Delta) \leq 0$. This pruning rule is sharper than the previous one and is more likely to prune nodes in the search tree. The implementation is simple within Algorithm AS: computations of SR values must simply be replaced by computations of SR_ε values. Moreover, the value SR_ε is obtained using MIP_η in which Equation (2a) is simply replaced by:

$$t \leq \sum_{i \in N} a^i - (1 + \varepsilon) \sum_{i \in N} a_s^i, \quad \forall s \in S$$

The resulting algorithm is denoted by AS_ε in the sequel. Then, the following proposition holds.

Proposition 4. *AS_ε returns an $(1 + \varepsilon)$ -approximation of the set $PW(\mathcal{X}, U, \Delta)$.*

Proof. Let x be a solution that does not belong to S at the end of the search procedure. Let $u \in U$ and $\delta \in \Delta$. We want to prove that $f(x, u, \delta) \leq (1 + \varepsilon)f(s, u, \delta)$ for some $s \in S$.

If x was inserted in S at some step, then x has necessarily been deleted during the filtering of S . In that case, by definition of the filtering, we know that there exists $s \in S$ such that $f(x, u, \delta) \leq f(s, u, \delta)$; hence, $f(x, u, \delta) \leq f(s, u, \delta) \leq (1 + \varepsilon)f(s, u, \delta)$.

Assume now that x was pruned at some step without ever belonging to S . In that case, we know that, at this step, there exists $s \in S$ such that $f(x, u, \delta) \leq (1 + \varepsilon)f(s, u, \delta)$. If solution s belongs to S at the end of the procedure, then we can directly infer the result. If solution s is deleted during the filtering, then we know that there exists $s' \in S$ such that $f(s, u, \delta) \leq f(s', u, \delta)$. Therefore, $f(x, u, \delta) \leq (1 + \varepsilon)f(s, u, \delta) \leq (1 + \varepsilon)f(s', u, \delta)$ which completes the proof (note that there is no chaining of errors). \square

4 ELICITATION FOR WINNER DETERMINATION

In Section 3, we have introduced a procedure for the determination of possible approval winners. It can be used in situations where a significant part of approval judgements is available. However, with incomplete preference information, the number of possible winners remains often very large, due to the combinatorial nature of the knapsack problem. Actually, the notion of possible winner (even its approximate version) is not sufficiently discriminating to support efficiently a collective decision making process. A more discriminating concept is the notion of necessary winner defined as follows:

Definition 2. *The set $NW(\mathcal{X}, U, \Delta)$ of necessary approval winners is the set of all solutions $x \in \mathcal{X}$ that maximize the approval score for all utility profiles $u \in U$ and all approval threshold vectors $\delta \in \Delta$. More formally: $NW(\mathcal{X}, U, \Delta) = \bigcap_{u \in U, \delta \in \Delta} \arg \max_{x \in \mathcal{X}} f(x, u, \delta)$.*

However, given the uncertainty sets U and Δ , it might be the case that no necessary winner exists. In this situation, we need to collect more preference information so as to be able to make a collective decision. For this reason, we focus now on preference elicitation for the approval winners determination.

One may consider a two stage procedure that consists first in eliciting the entire set of approval judgements and then in applying the

approval voting method. However, the exhaustive elicitation, prior to preference aggregation, is unfeasible due to the combinatorial nature of the problem. Collecting all approval statements would indeed require $O(n2^p)$ queries, for n agents and p items. We propose instead to combine preference elicitation and search so as to quickly focus the search on the relevant subsets of items and to concentrate the elicitation burden on the useful part of preferences. Our proposal is to collect approval statements from individuals during the search so as to progressively reduce the uncertainty attached to approval scores until determining the actual winners, i.e. the solutions maximizing the approval score.

Note that the number of possible winners reduces with the uncertainty about the agents' preferences. More precisely, for any $U' \subseteq U$ and any $\Delta' \subseteq \Delta$, we have $\text{PW}(\mathcal{X}, U', \Delta') \subseteq \text{PW}(\mathcal{X}, U, \Delta)$. Moreover, if U reduces to a single utility profile and Δ to a single approval threshold vector, then the possible winners become the actual winners of the election. More generally, if we consider an incremental elicitation procedure that iteratively collects preference information so as to progressively reduce sets U and Δ , we will necessarily reach a point where all possible winners are necessary winners. This will generally happen long before reducing U and Δ to singletons. At that point, we know that the possible winners are the actual winners.

Therefore, we propose now an incremental elicitation procedure progressively reducing uncertainty sets to U' and Δ' such that $\text{PW}(\mathcal{X}, U', \Delta') = \text{NW}(\mathcal{X}, U', \Delta')$. Our incremental elicitation algorithm consists in inserting preference queries in Algorithm AS (see the previous section) so as to discriminate between the current best solutions (those that were stored in S). In practice, S is now restricted to the most approved solutions found so far. Within this set, we can arbitrarily select a representant, named the incumbent. Initially, the incumbent may be any feasible solution to the knapsack problem (e.g., solution \bar{s} , see the initialization in Section 3). Each time a new solution (or a challenger) is found (line 6, Algorithm 1), it is now compared to the incumbent in terms of approval scores.

Note that, given the current uncertainty sets U and Δ , a solution $x \in \mathcal{X}$ is necessarily approved by an agent $i, i \in N$, if and only if $u^i(x_j) \geq \delta^i$ holds for all $u \in U$ and all $\delta \in \Delta$. This can be checked by testing whether the optimum of the following linear program is positive: $\min \sum_{j \in P} u_j^i x_j - \delta^i$ subject to Equations (2d-2g) and $\delta^i \geq 0, u_j^i \geq 0 \forall j \in P$. Similarly, testing whether a solution $x \in \mathcal{X}$ is necessarily disapproved by an agent or not can be performed using linear programming. Thus, each time a new solution is found, we can efficiently determine the agents - if they exist - that necessarily approve/disapprove it. Then, a natural query generation strategy, denoted by σ_0 hereafter, consists in questioning all the other agents to know whether they approve this solution or not. The challenger will be inserted in S if it has the same approval score than the incumbent. If the challenger is strictly better than the incumbent, then S is reinitialized to include only the challenger. This strategy provides an incremental search algorithm, named Algorithm IAS (Incremental Approval-based Search) in the sequel, that determines the set of approval winners, as shown by the following proposition:

Proposition 5. *IAS returns the exact set of approval winners.*

Proof. To prove this result, it is sufficient to prove the following: for any initial uncertainty sets U and Δ , Algorithm IAS terminates with uncertainty sets $U' \subseteq U$ and $\Delta' \subseteq \Delta$ such that $\text{PW}(\mathcal{X}, U', \Delta') = \text{NW}(\mathcal{X}, U', \Delta')$. Let $s \in S$ be any solution returned by IAS. For all $x \in \mathcal{X}$, we want to prove that we have $f(s, u, \delta) \geq f(x, u, \delta)$ for all $u \in U'$ and all $\delta \in \Delta'$ at the end of the search procedure. First, whenever a node η is pruned at some step k of IAS using the

current incumbent s_k , we know that, for all $x \in S_\eta$, $f(x, u, \delta) \leq f(s_k, u, \delta)$ for all $u \in U_k$ and all $\delta \in \Delta_k$ where U_k and Δ_k are the current uncertainty sets at step k . Since $U' \subseteq U_k$ and $\Delta' \subseteq \Delta_k$, we have $f(x, u, \delta) \leq f(s_k, u, \delta)$ for all $u \in U'$ and all $\delta \in \Delta'$. Then, according to strategy σ_0 , solution s_k is such that $f(s_k, u, \delta) \leq f(s, u, \delta)$ for all $u \in U'$ and all $\delta \in \Delta'$. The result is obtained by transitivity. \square

Algorithm IAS can be interrupted at any time to obtain the best solution found so far (the incumbent). Assume that IAS is interrupted at some step k , and let s_k denote the incumbent at the end of this step. It is worth noting that the quality of s_k can be measured by considering the following notion of maximum regret:

$$MR(s_k, \mathcal{X}, U_k, \Delta_k) = \max_{x \in \mathcal{X}} \max_{u \in U_k} \max_{\delta \in \Delta_k} \{f(x, u, \delta) - f(s_k, u, \delta)\}$$

where U_k, Δ_k and O_k respectively denote U, Δ and O at the end of step k . This maximum regret is indeed an upper bound on the gap to optimality: it represents the worst-case regret of choosing the incumbent instead of any other solution in terms of approval scores. This regret can be easily obtained using the available setwise max regrets $SR(\{s_k\}, S_\eta, U_k, \Delta_k)$ computed to evaluate nodes $\eta \in O_k$ during the search. More precisely, the following proposition holds:

Proposition 6. *At any iteration step k of IAS, we have:*

$$MR(s_k, \mathcal{X}, U_k, \Delta_k) = \max_{\eta \in O_k} SR(\{s_k\}, S_\eta, U_k, \Delta_k)$$

Proof. We want to prove that $\max_{x \in \mathcal{X}} R(s_k, x, U_k, \Delta_k)$ is equal to $\max_{\eta \in O_k} SR(\{s_k\}, S_\eta, U_k, \Delta_k)$, where $R(s_k, x, U_k, \Delta_k) = \max_{u \in U_k} \max_{\delta \in \Delta_k} \{f(x, u, \delta) - f(s_k, u, \delta)\}$. Let Y be the set of solutions defined by $Y = \{x \in S_\eta, \eta \in O_k\}$. Note that Y is not empty since Algorithm IAS has been interrupted. Then, by definition of setwise max regrets, we have:

$$\begin{aligned} SR(\{s_k\}, S_\eta, U_k, \Delta_k) &= \max_{u \in U_k} \max_{\delta \in \Delta_k} \max_{x \in S_\eta} \{f(x, u, \delta) - f(s_k, u, \delta)\} \\ &= \max_{x \in S_\eta} \max_{u \in U_k} \max_{\delta \in \Delta_k} \{f(x, u, \delta) - f(s_k, u, \delta)\} \\ &= \max_{x \in S_\eta} R(s_k, x, U_k, \Delta_k) \end{aligned}$$

for any $\eta \in O_k$. Therefore, $\max_{\eta \in O_k} SR(\{s_k\}, S_\eta, U_k, \Delta_k) = \max_{x \in Y} R(s_k, x, U_k, \Delta_k)$. Moreover, by definition of the pruning rule, we know that $SR(\{s_k\}, S_\eta, U_k, \Delta_k) \geq 0$ for any $\eta \in O_k$, and so $\max_{x \in Y} R(s_k, x, U_k, \Delta_k) \geq 0$. As a consequence, to establish the result, it is sufficient to prove that $R(s_k, x, U_k, \Delta_k) \leq 0$ for any solution $x \in \mathcal{X} \setminus Y$. Three cases may occur:

- For $x = s_k$, we can easily infer $R(s_k, x, U_k, \Delta_k) = 0$.
- For $x \in S$, since s_k is the incumbent, we know that $f(x, u, \delta) \leq f(s_k, u, \delta)$ for all $u \in U_k$ and $\delta \in \Delta_k$ (by definition of strategy σ_0). Therefore, we can infer $R(s_k, x, U_k, \Delta_k) \leq 0$.
- For $x \in S_\eta$, where η is a node that has been pruned during the search, we know that $SR(\{s_k\}, S_\eta, U_k, \Delta_k) < 0$ (by definition of the pruning rule). Then, since we have just proved that $SR(\{s_k\}, S_\eta, U_k, \Delta_k) = \max_{x' \in S_\eta} R(s_k, x', U_k, \Delta_k)$, we necessarily have $R(s_k, x, U_k, \Delta_k) < 0$.

Hence $\max_{x \in \mathcal{X}} R(s_k, x, U_k, \Delta_k) = \max_{x \in Y} R(s_k, x, U_k, \Delta_k)$. This establishes the result. \square

5 NUMERICAL TESTS

Since finding the knapsack that maximizes the approval score is NP-hard, even when all functions u_i and thresholds δ^i are known (see Proposition 1), a Branch and Bound for this problem may obviously have bad running time (in the worst case, all solutions may be enumerated). Usually, the efficiency of a Branch and Bound method is evaluated in an empirical way, analysing instances with a large number of solutions. It is also evaluated by the quality of the returned solutions. We show below the practical efficiency of our algorithms.

5.1 Approval-based Search (AS)

In this subsection, we report numerical tests aiming at evaluating the computation times (given in seconds) of possible winners calculation using AS and AS_ε (linear optimizations are performed using the Gurobi solver). In these experiments, instances of the multi-agent knapsack problem with $p = 12$ are generated as follows: weights $w_j, j \in P$, are uniformly drawn in $\{1, \dots, 100\}$. Then, capacity W is set to $d \times \sum_{j \in P} w_j$ where $d = 0.3, 0.4$ and 0.5 so as to vary the number of solutions: the number of maximal (w.r.t. set inclusion) feasible subsets is approximately equal to 500, 1000 and 2000 respectively. Moreover, to evaluate the impact of the uncertainty sets, we randomly generate $q = 10, 20$ preference statements per agent before running the algorithms. The computation times obtained by averaging over 50 runs for instances with $n = 20$ are reported in Table 1. We also report $\#S$ the average number of possible winners.

method	q	$d = 0.3$		$d = 0.4$		$d = 0.5$	
		time	$\#S$	time	$\#S$	time	$\#S$
AS	10	26.6	19.8	71.8	29.8	614.4	111.4
AS	20	25.2	17.4	55.7	23.9	442.3	90.3
$AS_{0.1}$	10	1.1	1.9	2.7	3.8	13.9	8.3
$AS_{0.1}$	20	1.0	1.6	2.3	2.7	9.4	5.0

Table 1. Computations of possible winners.

As expected, we can see that computation times increase with the size of the problem, and decrease as the number of available preference statements increase. Moreover, we observe that $AS_{0.1}$ is drastically faster than AS. More precisely, $AS_{0.1}$ determines an 1.1-approximation of possible winners in a few seconds while AS needs a few minutes on average.

5.2 Incremental Approval-based Search (IAS)

In this subsection, we report various numerical experiments aiming at evaluating the performance of IAS (presented in Section 4). In these experiments, only the preference ranking over single items is initially available for each agent. Then, answers to approval queries are simulated using approval thresholds and utility functions randomly generated with negative correlations so as to obtain difficult instances.

The first series of tests aims to evaluate the performance of its pruning rule in terms of average number of explored nodes (which corresponds to the average size of the search tree). For a basic comparison, we also report the number of nodes explored by exhaustive enumeration (this basic procedure is named S0 hereafter). Due to the possibility of collecting preference information during the search, algorithm IAS is significantly faster than AS and can solve much

larger instances. For example, we report here the results obtained for $p = 12, 15, 18$ and 20 (with $d = 0.4$), which approximatively represents 1000, 8500, 60000 and 250000 maximal (w.r.t. set inclusion) feasible solutions respectively. Results obtained by averaging over 30 runs are reported in Table 2 for problems involving 30 agents.

p	S0	IAS
12	3024	109
15	21468	238
18	152415	289
20	647834	440

Table 2. Number of explored nodes.

Table 2 shows that the average number of nodes explored by IAS remains quite low when increasing the number of solutions, whereas it drastically increases with an exhaustive enumeration. For example, our regret-based pruning rule reduces the average size of the search tree by a factor of 30 for instances with $p = 12$ (1000 feasible solutions), while reducing this number by a factor of 1600 for $p = 20$ (250000 feasible solutions).

The second series of experiments aims to evaluate the performance of IAS in terms of computation times (given in seconds) and average number of queries per agent (denoted by $\#Q$ hereafter). Results obtained by averaging over 50 runs are reported in Table 3 for instances involving 10, 20 and 30 agents.

n	$p = 15$		$p = 18$		$p = 20$	
	time	$\#Q$	time	$\#Q$	time	$\#Q$
10	9.2	10.5	29.8	12.2	48.7	14.3
20	17.7	9.8	42.2	10.1	162.1	14.9
30	29.7	10.9	154.0	11.8	225.7	13.1

Table 3. Computations of necessary winners.

In Table 3, we can see that IAS is very efficient both in terms of number of queries per agent and computation times; for instance, for problems with 10 agents and 20 items (250000 feasible solutions), IAS determines the set of optimal knapsacks in less than 50 seconds with less than 15 queries per agent on average, whereas the full elicitation of approval statements would require 250000 queries per agent. Moreover, although the number of feasible solutions increases exponentially with p the number of items, we can see that the average number of queries increases very slowly. This shows that the compact numerical representation of individual preferences is well exploited by IAS during the resolution. Finally, when comparing Tables 1 and 3, we observe that combining elicitation and search is more efficient than asking preference queries prior to the search; for example, with 20 agents, IAS determines the approval winners with no more than 15 queries per agent for problems with 250000 solutions, whereas asking 20 queries per agent prior to the search leaves us with 90 possible winners for smaller problems (2000 solutions) while doubling computation times. This shows that elicitation driven by resolution enables us to save both computation times and preference queries needed to determine the optimum.

We focus now on the evaluation of the branching strategy of IAS. The branching strategy is of crucial importance for the efficiency of

the query generation strategy proposed in Section 4. This elicitation strategy indeed consists in asking agents whether they approve or not some solutions found during the search; these solutions are obviously dependent on the branching strategy. For comparison, we also consider the interactive branch and bound procedure (named Random hereafter) that differs from IAS only on the branching strategy as follows: the next node to be explored and the next variable to be instantiated are both selected at random. For both branching strategies, we compute the maximum regret (MR) attached to the incumbent (using Proposition 6) each time an agent answers an approval query during the resolution. Recall that this regret gives an upper bound on the regret of choosing the incumbent instead of any other solution in terms of approval scores. Whenever this value equals zero, the incumbent is necessarily an approval winner. Regrets are here expressed on a normalized scale assigning value 1 to the initial maximum regret (computed before collecting any preference information) and value 0 when the maximum regret is 0. Figure 2 shows the results obtained by averaging over 30 runs.

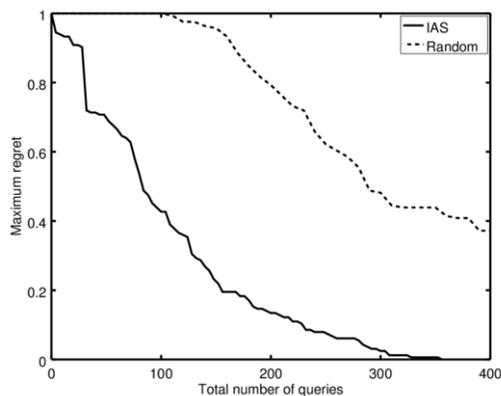


Figure 2. Maximum regret attached to the incumbent with respect to the total number of queries ($p = 12$, $d = 0.4$, $n = 30$).

In Figure 2, we can see that the maximum regret reduces much more quickly with IAS than with Random. For instance, after 240 queries (i.e. 8 queries per agent) on average, the regret of choosing the incumbent instead of any true approval winner is under 10% of the initial regret, whereas it remains above 65% with the random branching strategy. Hence, the elicitation strategy seems to be much more informative using our regret-based branching strategy instead of the random branching strategy.

Finally, we estimate the performance of IAS as an anytime algorithm by computing the maximum regret attached to the incumbent at each iteration step of IAS. This regret indeed provides a guarantee on the gap to optimality at any step of the branch and bound procedure. For comparison, we consider here also the random branch and bound procedure (named Random). The results reported in Figure 3 are obtained by averaging over 30 runs. We observe that the maximum regret decreases much more quickly with IAS than with Random, as the number of iteration steps increases. For instance, after 200 iteration steps on average, the maximum regret is under 10% with our procedure while remaining above 70% for the random strategy. Moreover, we observe that this regret drops drastically from the very beginning of IAS. This shows that IAS provides (very quickly) a good solution before the end of the search.

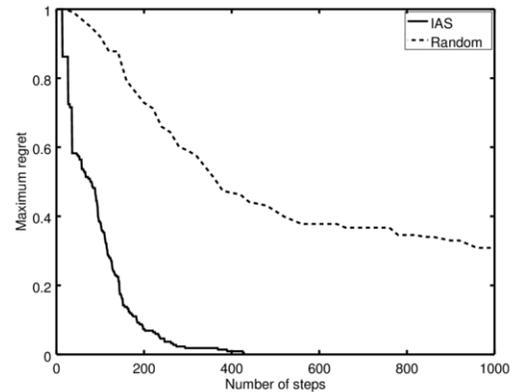


Figure 3. Maximum regret attached to the incumbent at each step of the branch and bound procedures ($p = 15$, $d = 0.5$, $n = 30$).

6 CONCLUSION

We have presented a new approach for incremental approval voting on combinatorial domains, illustrated on the knapsack problem. The first specificity of this approach is to exploit compact numerical representations of individual preferences (additive utilities) to propose more efficient elicitation sequences. Thus, learning that agent i approves or not solution x is no longer an isolated preference information; it induces a constraint on the utility space possibly reducing the set of weak-orders consistent with the observed preferences. This contributes to derive implicitly other approval judgements on other knapsacks, which saves many preference queries.

The second specificity of our approach is to interleave preference elicitation and search. This makes it possible to elicit preferences on a combinatorial set implicitly defined with a twofold benefit in view of winner determination: on the one hand, working on partial instances of feasible solutions in the search tree facilitates the identification of relevant queries and relieves the elicitation burden. On the other hand, the search is earlier focused on the relevant part of the solution space due to the integration of new preference information at decisive steps of the search algorithm, which saves a significant part of the computational effort. This enables to solve large instances for which the systematic elicitation of all approval statements is not feasible.

We see several directions to extend this work. The first one is to relax the additivity of individual utilities so as to be able to model interactions between items. In this line, it seems natural to use generalized additive utilities functions (GAI) that could also be elicited incrementally [10]. Another direction would be to consider alternative voting rules using possibly more information than approval statements. For example, positional scoring rules may be worth investigating under the assumption that individual preferences are representable by additive utilities. This is a challenging issue because, when preferences are only partially known, the ranges of possible ranks in individual rankings is generally too large to be decisive.

ACKNOWLEDGEMENTS

We wish to thank Jérôme Lang for stimulating discussions during the preparation of this paper. This work is supported by the ANR project 14-CE24-0007-01- Cocorico-CoDec.

REFERENCES

- [1] Haris Aziz, Markus Brill, Vincent Conitzer, Edith Elkind, Rupert Freeman, and Toby Walsh, 'Justified representation in approval-based committee voting', in *Proceedings of AAAI'15*, pp. 784–790, (2015).
- [2] Haris Aziz, Serge Gaspers, Joachim Gudmundsson, Simon Mackenzie, Nicholas Mattei, and Toby Walsh, 'Computational aspects of multi-winner approval voting', in *Proceedings of AAMAS'15*, pp. 107–115, (2015).
- [3] Nawal Benabbou and Patrice Perny, 'Incremental Weight Elicitation for Multiobjective State Space Search', in *Proceedings of AAAI'15*, pp. 1093–1099, (2015).
- [4] Nawal Benabbou and Patrice Perny, 'On possibly optimal tradeoffs in multicriteria spanning tree problems', in *Proceedings of ADT'15*, pp. 322–337, (2015).
- [5] Nadja Betzler, Arkadii Slinko, and Johannes Uhlmann, 'On the computation of fully proportional representation', *Journal of Artificial Intelligence Research*, 475–519, (2013).
- [6] Craig Boutilier, Ioannis Caragiannis, Simi Haber, Tyler Lu, Ariel D Procaccia, and Or Sheffet, 'Optimal social choice functions: A utilitarian view', *Artificial Intelligence*, 227, 190–213, (2015).
- [7] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans, 'Constraint-based Optimization and Utility Elicitation using the Minimax Decision Criterion', *Artificial Intelligence*, 170(8–9), 686–713, (2006).
- [8] Steven J. Brams and Peter C. Fishburn, 'Approval voting', *The American Political Science Review*, 72(3), 831–847, (1978).
- [9] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, Cambridge University Press, 2016. In press.
- [10] Darius Braziunas and Craig Boutilier, 'Minimax regret based elicitation of generalized additive utilities', in *Proceedings of UAI'07*, pp. 25–32, (2007).
- [11] Ioannis Caragiannis and Ariel D Procaccia, 'Voting almost maximizes social welfare despite limited communication', *Artificial Intelligence*, 175(9), 1655–1671, (2011).
- [12] Urszula Chajewska, Daphne Koller, and Ronald Parr, 'Making rational decisions using adaptive utility elicitation', in *Proceedings of AAAI'00*, pp. 363–369, (2000).
- [13] John R Chamberlin and Paul N Courant, 'Representative deliberations and representative decisions: Proportional representation and the Borda rule', *American Political Science Review*, 77(03), 718–733, (1983).
- [14] Lihi Naamani Dery, Meir Kalech, Lior Rokach, and Bracha Shapira, 'Reaching a joint decision with minimal elicitation of voter preferences', *Information Sciences*, 278, 466–487, (2014).
- [15] Ning Ding and Fangzhen Lin, 'Voting with partial information: what questions to ask?', in *Proceedings of AAMAS'13*, pp. 1237–1238, (2013).
- [16] Edith Elkind, Piotr Faliszewski, Piotr Skowron, and Arkadii Slinko, 'Properties of multiwinner voting rules', in *Proceedings of AAMAS'14*, pp. 53–60, (2014).
- [17] M. Gelain, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh, 'Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies', *Artificial Intelligence Journal*, 174(3–4), 270–294, (2010).
- [18] Meir Kalech, Sarit Kraus, and Gal A. Kaminka, 'Practical voting rules with partial information', *Autonomous Agents and Multi-Agent Systems*, 22(1), 151–182, (2010).
- [19] D Marc Kilgour, 'Approval balloting for multi-winner elections', in *Handbook on approval voting*, 105–124, Springer, (2010).
- [20] Kathrin Konczak and Jérôme Lang, 'Voting procedures with incomplete preferences', in *Proceedings of IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, volume 20, (2005).
- [21] Jérôme Lang, Maria Silvia Pini, Francesca Rossi, Domenico Salvagnin, Kristen Brent Venable, and Toby Walsh, 'Winner determination in voting trees with incomplete preferences and weighted votes', *Autonomous Agents and Multi-Agent Systems*, 25(1), 130–157, (2012).
- [22] Tyler Lu and Craig Boutilier, 'Budgeted social choice: From consensus to personalized decision making', in *Proceedings of IJCAI'11*, volume 11, pp. 280–286, (2011).
- [23] Tyler Lu and Craig Boutilier, 'Robust approximation and incremental elicitation in voting protocols', in *Proceedings of IJCAI'11*, pp. 287–293, (2011).
- [24] Tyler Lu and Craig Boutilier, 'Multi-winner social choice with incomplete preferences', in *Proceedings of IJCAI'13*, pp. 263–270, (2013).
- [25] Reshef Meir, Ariel D Procaccia, Jeffrey S Rosenschein, and Aviv Zohar, 'Complexity of strategic behavior in multi-winner elections.', *Journal of Artificial Intelligence Research (JAIR)*, 33, 149–178, (2008).
- [26] Burt L Monroe, 'Fully proportional representation', *American Political Science Review*, 89(04), 925–940, (1995).
- [27] Joel Oren and Brendan Lucier, 'Online (budgeted) social choice', *Proceedings of AAAI'14*, 1456–1462, (2014).
- [28] Richard F Potthoff and Steven J Brams, 'Proportional representation broadening the options', *Journal of Theoretical Politics*, 10(2), 147–178, (1998).
- [29] Ariel D Procaccia and Jeffrey S Rosenschein, 'The distortion of cardinal preferences in voting', in *Proceedings of the Tenth International Workshop on Cooperative Information Agent*, 317–331, Springer, (2006).
- [30] Ariel D Procaccia, Jeffrey S Rosenschein, and Aviv Zohar, 'On the complexity of achieving proportional representation', *Social Choice and Welfare*, 30(3), 353–362, (2008).
- [31] Piotr Skowron, Piotr Faliszewski, and Jérôme Lang, 'Finding a collective set of items: From proportional multirepresentation to group recommendation', in *Proceedings of AAAI'15*, pp. 2131–2137, (2015).
- [32] Piotr Skowron, Lan Yu, Piotr Faliszewski, and Edith Elkind, 'The complexity of fully proportional representation for single-crossing electorates', *Theoretical Computer Science*, 569, 43–57, (2015).
- [33] T. Wang and C. Boutilier, 'Incremental Utility Elicitation with the Minimax Regret Decision Criterion', in *Proceedings of IJCAI-03*, pp. 309–316, (2003).
- [34] Lirong Xia and Vincent Conitzer, 'Determining possible and necessary winners given partial orders', *Journal of Artificial Intelligence Research (JAIR)*, 41, 25–67, (2011).

Propositional Abduction with Implicit Hitting Sets

Alexey Ignatiev^{1, 2} and Antonio Morgado¹ and Joao Marques-Silva¹

Abstract. Logic-based abduction finds important applications in artificial intelligence and related areas. One application example is in finding explanations for observed phenomena. Propositional abduction is a restriction of abduction to the propositional domain, and complexity-wise is in the second level of the polynomial hierarchy. Recent work has shown that exploiting implicit hitting sets and propositional satisfiability (SAT) solvers provides an efficient approach for propositional abduction. This paper investigates this earlier work and proposes a number of algorithmic improvements. These improvements are shown to yield exponential reductions in the number of SAT solver calls. More importantly, the experimental results show significant performance improvements compared to the the best approaches for propositional abduction.

1 Introduction

Logic-based abduction finds relevant applications in artificial intelligence and related areas [9, 12, 18–20, 25–27, 33, 49–52, 54, 58–60]. Given a background theory, a set of manifestations and a set of hypotheses, abduction seeks to identify a cost-minimum set of hypotheses which explain the manifestations and are consistent given the background theory. Propositional abduction is hard for the second level of the polynomial hierarchy, but finds a growing number of applications [58, 60]. Noticeable examples of where propositional abduction algorithms can be applied include abductive inference [15–17, 55], logic programming [34, 36, 41], knowledge bases updates [35, 59], security protocols verification [1], and constraint optimization [21, 22], among many others.

Given the complexity class of propositional abduction, it is conceptually simple to solve the problem with a linear (or logarithmic) number of calls to a Σ_2^P -oracle, e.g. a oracle for quantified Boolean formulas (QBF) with one quantifier alternation. Unfortunately, in practice QBF solvers are not as efficient as SAT solvers, and do not scale as well. As a result, recent work [58] on solving propositional abduction focused on using calls to SAT oracles instead of QBF oracles, following a trend also observed for solving QBF [28, 29, 31, 32]. This recent work on solving propositional abduction is motivated by the practical success of implicit hitting set algorithms in a number of different settings [4, 10, 11, 13, 23, 24, 28, 29, 31, 32, 37–39, 45, 53, 58, 61].

The contributions of this paper can be summarized as follows. The paper revisits QBF models for solving propositional abduction and proposes a Quantified MaxSAT (QMaxSAT) [23, 24] model for propositional abduction. Moreover, the paper notes that the MaxHS [13] approach for MaxSAT can be readily applied to QMaxSAT by replacing the oracle used. The paper then investigates the application of implicit hitting sets to solving propositional abduction

[58] and identifies a number of algorithmic improvements. This leads to a new algorithm, Hyper, for solving propositional abduction. This new algorithm is shown to significantly outperform the current state of the art, solving a large number of instances that could not be solved with existing solutions. More importantly, the paper shows that the algorithmic improvements proposed can save an exponential number of iterations when compared with the current state of the art [58].

The paper is organized as follows. Section 2 introduces the definitions used throughout the paper, and also overviews related work. Section 3 revisits a QBF model for abduction, which is then used for developing a number of alternative approaches for solving propositional abduction. Among these, the paper proposes improvements to recent work [58], which are shown to yield exponential reductions on the number of SAT oracle calls. Section 4 analyzes the experimental results, running existing and the proposed algorithms on existing problem instances [58]. Section 4 also provides experimental evidence that the proposed algorithms for propositional abduction can save an exponential number of oracle calls in different settings. Section 5 concludes the paper, and identifies possible research directions.

2 Preliminaries

This section introduces the notation and definitions used throughout the paper.

2.1 Satisfiability

Standard propositional logic definitions apply (e.g. [7]). CNF formulas are defined over a set of propositional variables. A CNF formula (or theory) T is a propositional formula represented as a conjunction of clauses, also interpreted as a set of clauses. A clause is a disjunction of literals, also interpreted as a set of literals. A literal is a variable or its complement. The set of variables of a theory T is denoted $X \triangleq \text{var}(T)$. The dependency of T on X can be made explicit by writing $T(X)$. Where convenient, a formula can be rewritten with a fresh set of variables, e.g. we can replace X by Y , writing $T(Y)$. Conflict-driven clause learning (CDCL) SAT solvers are summarized in [7]. Throughout the paper, SAT solvers are viewed as oracles. Given a CNF formula F , a SAT oracle decides whether F is satisfiable, and returns a satisfying assignment μ if F is satisfiable. A SAT oracle can also return a subset of the clauses (i.e. an unsatisfiable core $U \subseteq F$) if F is unsatisfiable.

CNF formulas are often used to model overconstrained problems. In general, clauses in a CNF formula are characterized as hard, meaning that these must be satisfied, or soft, meaning that these are to be satisfied, if at all possible. A weight can be associated with each soft clause, and the goal of maximum satisfiability (MaxSAT) is to find an assignment to the propositional variables such that the hard clauses are satisfied, and the sum of the satisfied soft clauses is maximized.

¹ LaSIGE, Faculty of Science, University of Lisbon, Portugal, email: {aignatiev,ajmorgado,jpms}@ciencias.ulisboa.pt

² ISDCT SB RAS, Irkutsk, Russia

Branch-and-bound algorithms for MaxSAT are overviewed in [7]. Recent work on MaxSAT investigated core-guided algorithms [3,46] and also the use of implicit hitting sets [13].

In the analysis of unsatisfiable CNF formulas, a number of definitions are used. Given an unsatisfiable CNF formula F , a minimal unsatisfiable subset (MUS) $M \subseteq F$ is both unsatisfiable and irreducible. Given an unsatisfiable CNF formula F , a minimal correction subset (MCS) $C \subseteq F$ is both irreducible and its complement is satisfiable. Given an unsatisfiable CNF formula F , a maximal satisfiable subset (MSS) S is the complement of some MCS of F . A largest MSS is a solution to the MaxSAT problem.

Additionally, it is well-known [40,56] that MUSes and MCSes are connected by a *hitting set duality*. Given a collection Γ of sets from a universe \mathbb{U} , a hitting set h for Γ is a set such that $\forall S \in \Gamma, h \cap S \neq \emptyset$. A hitting set h is *minimal* if none of its subsets is a hitting set. It is straightforward to extend the previous definitions to the case where there are hard clauses.

Quantified Boolean formulas (QBFs) are an extension of propositional logic with *existential* and *universal* quantifiers (\forall, \exists) [7]. A QBF can be in *prenex closed* form $Q_1x_1 \dots Q_nx_n. \varphi$, where $Q_i \in \{\forall, \exists\}$, x_i are distinct Boolean variables, and φ is a Boolean formula over the variables x_i and the constants 0 (false), 1 (true). The sequence of quantifiers in a QBF is called the *prefix* and the Boolean formula the *matrix*. The semantics of QBF is defined recursively. A QBF $\exists x_1 Q_2x_2 \dots Q_nx_n. \varphi$ is true iff $Q_2x_2 \dots Q_nx_n. \varphi|_{x_1=1}$ or $Q_2x_2 \dots Q_nx_n. \varphi|_{x_1=0}$ is true. A QBF $\forall x_1 Q_2x_2 \dots Q_nx_n. \varphi$ is true iff both $Q_2x_2 \dots Q_nx_n. \varphi|_{x_1=1}$ and $Q_2x_2 \dots Q_nx_n. \varphi|_{x_1=0}$ are true. To decide whether a given QBF is true or not, is known to be PSPACE-complete [7].

2.2 Propositional Abduction

A propositional abduction problem (PAP) is a 5-tuple $P = (V, H, M, T, c)$. V is a finite set of variables. H, M and T are CNF formulas representing, respectively, the set of hypotheses, the set of manifestations, and the background theory. c is a cost function associating a cost with each clause of $H, c : H \rightarrow \mathbb{R}^+$.

Given a background theory T , a set $S \subseteq H$ of hypotheses is an explanation (for the manifestations) if: (i) S entails the manifestations M (given T); and (ii) S is consistent (given T). The propositional abduction problem consists in computing a minimum size explanation for the manifestations subject to the background theory.

Definition 1 (Explanations for P [58]) Let $P = (V, H, M, T, c)$ be a PAP. The set of explanations of P is given by the set $\text{Expl}(P) = \{S \subseteq H \mid T \wedge S \not\models \perp, T \wedge S \models M\}$. The minimum-cost solutions of P are given by $\text{Expl}_c(P) = \text{argmin}_{E \in \text{Expl}(P)} (c(E))$.

The complexity of logic-based abduction has been investigated in a number of works [9,18], and is surveyed in [58]. Checking whether $S \subseteq H$ is an explanation for a PAP is D^P -complete. Deciding the existence of some explanation is Σ_2^P -complete. Finding a minimum-size explanation can be achieved with a linear number of calls to a Σ_2^P oracle or, if the costs are polynomially bounded, with a logarithmic number of calls to a Σ_2^P oracle.

Example 1 (Example abduction instance.) Consider the propositional abduction problem instance $P = (V, H, M, T, c)$ with the set of variables V , the set of hypotheses H , the manifestations M , and

Input: F WCNF formula

Output: $(\mu, \text{Cost}(\mu))$ MaxSAT assignment and cost

```

1 begin
2    $K \leftarrow \emptyset$ 
3   while true do
4      $h \leftarrow \text{MinimumHS}(K)$ 
5      $(st, \mu) \leftarrow \text{SAT}(F \setminus h)$ 
6     // If  $st$ , then  $\mu$  is an assignment
7     // Otherwise,  $\mu$  is a core
8     if  $st$  then return  $(\mu, \text{Cost}(\mu))$ 
9      $K \leftarrow K \cup \{\mu\}$ 
10 end

```

Algorithm 1: The MaxHS algorithm [13]

the background theory T given by,

$$\begin{aligned}
 V &= \{x_1, x_2, x_3, x_4\} \\
 H &= \{(x_1), (x_2), (x_3)\} \\
 M &= \{(x_4)\} \\
 T &= \{(\neg x_1 \vee x_4), (\neg x_2 \vee \neg x_3 \vee x_4)\}
 \end{aligned} \tag{1}$$

The (propositional) abduction problem for this example is to find a minimum cost subset S of H , such that (i) S is consistent with T (i.e. $T \wedge S \not\models \perp$); and (ii) S and T entail M (i.e. $T \wedge S \models M$). For this instance of propositional abduction, the minimum cost explanation is then $S = \{(x_1)\}$.

2.3 Related Work

This paper builds on recent work on algorithms for propositional abduction [58], which builds on earlier work on solving maximum satisfiability with implicit hitting set algorithms [13]. This work is also tightly related to the body of work on handling hitting sets implicitly [10,11,13,37,45,58], which is also tightly related with implicit hitting set dualization [4,38,39,53,61], but also with abstraction refinement in QBF solving and optimization [23,24,28,29,31,32].

The use of implicit hitting sets for solving MaxSAT is embodied by MaxHS [13], which is summarized in Algorithm 1. The algorithm computes minimum hitting sets of a set of sets, each of which represents an unsatisfiable subformula of the target formula. This essentially exploits Reiter's [56] well-known hitting set relationship between MCSes and MUSes [6,8,40], where an MCS is a minimal hitting set of the MUSes and vice-versa. Moreover, since a minimum hitting set is being computed, we are in search of the smallest MCS, i.e. the MaxSAT solution. In case there are hard clauses, MaxHS needs to take this into consideration, including checking the consistency of the hard clauses. Besides MaxHS, recent work on MaxSAT solving is based on iterative unsatisfiable core identification [3,46].

Recent work on propositional abduction builds on MaxHS and proposes a novel algorithm, AbHS/AbHS+. AbHS mimics MaxHS in that the algorithm iteratively computes minimum cost hitting sets, which identify a subset $S \subseteq H$. This set S is then used for checking whether it represents an explanation of the propositional abduction problem. Since it is a minimum hitting set then, if the conditions hold, it is a minimum-cost explanation. AbHS is summarized in Algorithm 2. As in MaxHS, the algorithm iteratively computes minimum hitting sets using an ILP solver (line 13). The outcome is a subset S of H . The abduction conditions are checked with two distinct SAT oracle calls. One oracle call checks whether $T \wedge S \wedge (\neg M)$ is inconsistent, i.e. whether $T \wedge S \models M$. If the formula is satisfiable, then the set of sets to hit (K) is updated with another set, of the clauses in H falsified by the computed satisfying assignment. If $T \wedge S \wedge (\neg M)$

Input: PAP $P = (V, H, M, T, c)$

Output: Minimum cost explanation S

```

1 begin
2    $K \leftarrow \emptyset$ 
3    $S \leftarrow \emptyset$ 
4    $(st, \mu) \leftarrow \text{SAT}(T \wedge H \wedge (\neg M))$ 
5   if  $st$  then return  $\emptyset$ 
6   while  $S \neq H$  do
7      $(st, \mu) \leftarrow \text{SAT}(T \wedge S \wedge (\neg M))$ 
8     if  $st$  then  $K \leftarrow K \cup \{h \in H \mid \mu(h) = 0\}$ 
9     else
10       $(st, \mu) \leftarrow \text{SAT}(T \wedge S)$ 
11      if not  $st$  then  $K \leftarrow K \cup \{H \setminus S\}$ 
12      else return  $S$ 
13     $S \leftarrow \text{MinimumHS}(K)$ 
14  return  $\emptyset$ 
15 end

```

Algorithm 2: The AbHS/AbHS+ abduction algorithm [58]

is inconsistent, then a second oracle call checks whether $T \wedge S$ is consistent. If it is, then a minimum-cost explanation has been identified. Otherwise, AbHS creates a hitting set by requiring that some non-selected clause of H be selected in subsequently computed minimum hitting sets. For the AbHS+ variant [58], the set added can be viewed as the complements of the literals selecting each clause in S , i.e. at least some clause in S must not be picked.

There is a vast body of work on exploiting implicit hitting sets. The concept of exploiting implicit hitting sets is intended to mean that, instead of starting from an explicit representation of the complete set of hitting sets, hitting sets are computed on demand, as deemed necessary by the problem being solved. An earlier example of exploiting implicit hitting sets is the work of Bailey and Stuckey [6], in the concrete application to hitting set dualization. The concept was re-introduced more recently [10, 13, 37], and then applied in a number of different settings. Among this vast body of work, as will become clear throughout the paper, our work can be related with abstraction refinement ideas used in recent expansion-based QBF solvers, namely RReQS [24, 28, 31] and quantified optimization extensions [23, 29], but now in the context of handling implicit hitting sets.

3 Algorithms for Propositional Abduction

This section overviews different algorithms for solving propositional abduction, all of which are based on reducing the problem to QBF.

3.1 QBF Model for Abduction

Given a PAP $P = (V, H, M, T, c)$, the problem of deciding whether some set S is an explanation can be reduced to QBF. $S \subseteq H$ is an explanation of P iff:

$$\exists_X T(X) \wedge S(X) \wedge \forall_Y \neg(T(Y) \wedge S(Y) \wedge \neg M(Y)) \quad (2)$$

is true. (Observe X and Y denote sets of variables, thus highlighting that different sets of variables are used.) (2) can be rewritten as follows:

$$\exists_X \phi(X) \wedge \forall_Y \psi(Y), \quad (3)$$

where $\phi = T \wedge S$ and $\psi = \neg(T \wedge S \wedge \neg M)$.

As indicated in Section 2, the goal of propositional abduction is to find a minimum cost explanation, i.e. to pick a minimum cost set $S \subseteq H$ that is an explanation of P .

The problem of finding a minimum cost explanation of P can be reduced to quantified maximum satisfiability [24] (QMaxSAT). Associate a variable r_i with each clause $C_i \in H$, and create a set H' where each clause $C_i \in H$ is replaced by $(r_i \vee C_i)$, to enable relaxing the clause. Let R denote the set of the r_i (relaxation) variables, with $|R| = |H|$. H' serves to create a modified QBF:

$$\exists_R \exists_X T(X) \wedge H'(R, X) \wedge \forall_Y \neg(T(Y) \wedge H'(R, Y) \wedge \neg M(Y)) \quad (4)$$

As before, (4) can be rewritten as follows:

$$\exists_R \exists_X \phi(X, R) \wedge \forall_Y \psi(Y, R) \quad (5)$$

The above QBF can be transformed into prenex normal form, and represents the hard part of the QMaxSAT problem. Moreover, the fact that the goal is to compute a minimum cost explanation of P is modeled by adding a soft clause $(\neg r_i)$, with cost $c(C_i)$, for each $r_i \in R$. Each soft clause denotes a preference not to include the associated clause in H in the computed explanation.

Example 2 With respect to the PAP from example 1, the QBF associated with the hard part of the QMaxSAT problem is:

$$\begin{aligned} & \exists_{r_1, r_2, r_3} \exists_{x_1, x_2, x_3, x_4} \\ & (\neg x_1 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge \\ & (r_1 \vee x_1) \wedge (r_2 \vee x_2) \wedge (r_3 \vee x_3) \\ & \forall_{y_1, y_2, y_3, y_4} \\ & \neg[(\neg y_1 \vee y_4) \wedge (\neg y_2 \vee \neg y_3 \vee y_4) \wedge \\ & (r_1 \vee y_1) \wedge (r_2 \vee y_2) \wedge (r_3 \vee y_3) \wedge (\neg y_4)] \end{aligned} \quad (6)$$

with the soft clauses being $\{(\neg r_1), (\neg r_2), (\neg r_3)\}$.

The QMaxSAT formulation can be used to develop a number of alternative approaches for solving PAP. These approaches are detailed in the next sections.

Observe that, if the propositional abduction problem is not trivial to solve, then the QBF (2) is false for $S = \emptyset$ and $S = H$.

3.2 Abduction with QMaxSAT

Similarly to MaxSAT, a number of algorithms can be envisioned for solving QMaxSAT. These are analyzed in the subsections below, taking into account the specific structure of the reduction of propositional abduction to QMaxSAT.

3.2.1 Iterative QBF Solving

A standard approach for solving MaxSAT is iterative SAT solving [46]. Similarly, we can use iterative QBF solving for QMaxSAT. At each step, and given cost k , the following pseudo-Boolean constraint is used:

$$\text{PB}(R, k) \triangleq \left(\sum_{C_i \in H} c(C_i) r_i \leq k \right) \quad (7)$$

For some positive k , the QBF (5) can be used for iteratively QBF solving as follows:

$$\exists_R \text{PB}(R, k) \wedge \exists_X \phi(X, R) \wedge \forall_Y \psi(Y, R) \quad (8)$$

Clearly, binary search can be used to ensure a linear (or logarithmic, depending on whether the costs are bounded) number of Σ_2^P oracle calls [18]. In practice, most QBF solvers expect clausal representations. Classification introduces one additional level of quantification. Typically, each quantification level makes a QBF formula harder to decide. And thus in practice, QBF solvers scale worse than SAT solvers, and so this approach is unlikely to scale for large propositional abduction problem instances.

3.2.2 Core-Guided QBF Solving

Core-guided algorithms [3, 46] represent another approach for solving QMaxSAT. Many variants of core-guided MaxSAT algorithms have been proposed in recent years [3, 46].

Given the reduction of propositional abduction to QMaxSAT, any core-guided MaxSAT algorithm can be used, provided a core-producing QBF solver is used [24, 42].

Nevertheless, for the abduction problem the use of alternative MaxSAT solving approaches, based on MaxHS [13] is amenable to efficient optimizations, which solely use SAT solvers [58].

3.2.3 Exploiting MaxHS

A recent approach for MaxSAT solving is MaxHS [13], which exploits integer linear programming (ILP) solving, resulting in simpler SAT oracle calls, at the cost of a possibly exponentially larger number of oracle calls. Recall that the MaxHS approach for solving MaxSAT is outlined in Algorithm 1.

A straightforward solution for solving QMaxSAT is to replace the SAT oracle by a QBF oracle in MaxHS. This approach is referred to as QMaxHS, and it was implemented on top of DepQBF, the known QBF solver which is capable of reporting unsatisfiable cores if the input QBF is false [42]. It should be noted (and it is also mentioned in Section 4) that the implementation of QMaxHS performs quite bad (it cannot solve any benchmark instances considered in Section 4). A possible explanation of this is that the QBF formulas, which are iteratively solved by the QBF solver, are too hard even though the original idea of MaxHS-like algorithms is to get (many) simple calls to the oracle. This suggests that implementing core-guided QMaxSAT algorithms would not pay off as well since the QBF formulas in core-guided QMaxSAT are much harder to deal with. Recent work on propositional abduction [58] proposed a MaxHS-like approach, but the QBF oracle call was replaced by two SAT solver calls, which is expected to outperform QMaxHS.

3.3 Exploiting Implicit Hitting Sets

The use of implicit hitting sets for abduction was proposed in recent work [58]. This work can be viewed as extending the MaxHS algorithm for MaxSAT [13], which is based on implicit enumeration of hitting sets. In contrast to MaxHS, instead of one SAT oracle call, AbHS [58] uses two SAT oracle calls, one to check entailment of M by $T \wedge S$ and another to check the consistency of $T \wedge S$. A variant of AbHS, AbHS+, differs on which sets are added to the hitting set representation. Whereas the connection of MaxHS and AbHS with implicit hitting sets is clear, the approach used in AbHS+ can be viewed as adding both only positive clauses and only negative clauses to hit, and so the connection with hitting sets is less evident. An alternative way of explaining AbHS/AbHS+ is to consider (5). The ILP solver is used for computing some minimum cost hitting set, which represents a set of clauses $S \subseteq H$. Then, one SAT oracle call checks $\exists_X \phi(X)$, given S , and another SAT oracle call checks $\forall_Y \psi(Y)$, also given S .

(Observe that this second formula corresponds to checking unsatisfiability.) This explanation of how AbHS/AbHS+ works is investigated in greater detail below.

In the following, an alternative approach for propositional abduction is developed which, similarly to AbHS/AbHS+, is also based on handling implicit hitting sets, but which is shown to yield exponential reductions on the number of oracle calls in the worst case. The new algorithm, Hyper, shares similarities with MaxHS and AbHS/AbHS+ in that minimum hitting sets are also computed, and an implicit representation of the hitting sets is maintained. However, and in contrast with AbHS/AbHS+, Hyper analyzes the structure of the problem formulation, and develops a number of optimizations that exploit that formulation.

The propositional abduction problem formulation can be presented in a slightly modified form, i.e. to find a smallest cost set $S \subseteq H$, consistent with T (i.e. a T -consistent set S), which, together with T , entails M . Consider a T -consistent candidate set $S \subseteq H$. If $T \wedge S \not\models M$, then the formula $T \wedge S \wedge (\neg M)$ is satisfiable and the satisfying assignment returned by the SAT oracle is a *counterexample* explaining why the selected set S is such that $T \wedge S \not\models M$. Moreover, this satisfying assignment can be used for revealing a (possibly subset-minimal) set of clauses in $H \setminus S$ which are falsified. Clearly, one of these falsified clauses must be included (i.e. *hit*) in any T -consistent set $S \subseteq H$ which, together with T , will entail M . Thus, from each T -consistent candidate set $S \subseteq H$ which, together with T , does not entail M , we can identify a set of clauses, from which at least one must be picked, in order to pick another T -consistent set $S \subseteq H$, such that eventually $T \wedge S \models M$.

The approach outlined in the paragraph above, although apparently similar to the description of AbHS/AbHS+, reveals a number of significant insights. First, checking the consistency of S with T can be carried out *concurrently* with the selection of S itself. This is also apparent from the QBF formulation (2), in that all existential quantifiers can be aggregated and handled simultaneously. Thus, each minimum hitting set S is computed while guaranteeing that $T \wedge S$ holds. More importantly, after each set S is picked, it is only necessary to check whether $T \wedge S \models M$, and this can be done with a *single* SAT oracle call.

Concretely, the next minimum cost hitting set, given the already identified sets to hit, is computed guaranteeing that the existential part of (4) is satisfied:

$$\exists_R \exists_X T(X) \wedge H'(R, X) \quad (9)$$

The selected set S , identified by the assignment to the R variables, is then used for checking the satisfiability of the second component of QBF (2):

$$\forall_Y \neg(T(Y) \wedge S(Y) \wedge \neg M(Y)) \quad (10)$$

Observe that this can be decided with a SAT oracle call.

Thus, a careful analysis of the problem formulation enables improving upon AbHS/AbHS+, specifically by eliminating one SAT oracle call per iteration. However, as shown in the next section, the new algorithm can save an exponentially large number of SAT oracle calls when compared with AbHS/AbHS+.

Besides the aggregation of the existential quantifiers, additional optimizations can be envisioned. Propositional abduction seeks a minimum cost set $S \subseteq H$ such that $T \wedge S \models M$. Ideally, one would prefer not to select a set $S \subseteq H$ such that $T \wedge S \not\models M$. Observe that $T \wedge S \models M$ implies that $T \wedge S \wedge M$ holds, but the converse is in general not true. Thus, M can be added to (9), resulting in requiring that $T \wedge S \wedge M$ be consistent when selecting the set S . The inclusion of M when picking a minimum hitting set can also reduce the num-

Input: PAP $P = (V, H, M, T, c)$

Output: Minimum cost explanation S

```

1 begin
2    $(H', R) \leftarrow \text{RelaxCls}(H)$ 
3    $B \leftarrow T \wedge M \wedge H'$ 
4    $A \leftarrow \emptyset$ 
5   while true do
6      $(st, h) \leftarrow \text{MinimumHS}(A, B)$ 
7     if not  $st$  then return  $\emptyset$ 
8      $S \leftarrow \{C_i \in H' \mid r_i \in h\}$ 
9      $(st, \mu) \leftarrow \text{SAT}(T \wedge S \wedge (\neg M))$ 
10    if not  $st$  then return  $S$ 
11     $W \leftarrow \text{PickFalseCls}(H \setminus S, \mu)$ 
12     $Y \leftarrow \text{GetRelaxationVars}(W)$ 
13     $A \leftarrow A \cup Y$ 
14 end

```

Algorithm 3: Organization of Hyper

ber of oracle calls exponentially. This is also investigated in the next section.

The new Hyper algorithm for propositional abduction is shown as Algorithm 3. Clauses in H are relaxed, to allow each clause $C_i \in H$ to be picked when the associated relaxation variable r_i is assigned value 1. The minimum hitting sets are computed for the set of sets A , subject to a background theory B , which conjoins T , M and the relaxed clauses of H . In Hyper minimum hitting sets are computed with a MaxSAT solver [47], since the hard part (containing T , M and H') plays a significant role in deciding consistency. If all hitting sets have been (implicitly) tried unsuccessfully, then the algorithm terminates and returns \emptyset . If not, and if $T \wedge S \models M$, then the algorithm terminates and returns the computed set S . Otherwise, a subset of the clauses in $H \setminus S$, which are falsified by the computed satisfying assignment μ , is identified, and the associated relaxation variables are used to create another set to hit, i.e. one of those clauses must be included in any selected set S .

3.3.1 Additional Optimizations

A few additional optimizations are possible, which can be expected to have some impact on the performance of Hyper. These are discussed next. The first two optimizations are implemented in a variant of Hyper, Hyper*. The other optimizations are analyzed to explain why performance improvements are not expected to be significant.

One optimization is to perform *partial reduction* of the counterexamples computed in lines 11–12 of Algorithm 3. Recall that counterexamples, i.e. sets that need to be hit next time, comprise clauses of $H \setminus S$ that are falsified by a model μ of $T \wedge S \wedge (\neg M)$. Thus, the counterexamples can be seen as correction subsets for the partial CNF formula $T \wedge H \wedge (\neg M)$, where $T \wedge (\neg M)$ is the hard part and H is the soft part. Observe that instead of computing *any* correction subset, one may want to reduce it to get a subset-minimal correction subset (an MCS), i.e. to try to minimize the number of falsified clauses in $H \setminus S$. In Hyper* this is done using the standard linear search algorithm [44], which iterates through the falsified clauses and tries to satisfy them. In order not to spend too much time on doing the reduction, the version of Hyper* presented here iterates only over $0.2 \times m$ falsified clauses of the initial counterexample starting from the clauses of the smallest weight, where m is the size of the initial counterexample.

A second optimization is to start by computing a fixed number of minimum hitting sets. Given that $T \wedge H \wedge (\neg M)$ is inconsistent, one

can enumerate MCSes of this formula, which must be hit, so that one can eventually prove that there exists some set S with $T \wedge S \models M$. In Hyper, 100 MCSes of $T \wedge H \wedge (\neg M)$ are computed before starting the process of generating candidate sets S . These MCSes are computed by size, using MaxSAT-based MCS enumeration [40, 48].

A third optimization respects the clauses in M . Any $C_j \in M$, such that $T \models C_j$, can be removed from M . In a preprocessing step, each clause in M is checked for entailment with respect to T . Any entailed clause is removed. This technique reduces the practical hardness of the formulas checked for unsatisfiability. Since most of the running time of Hyper is spent on computing minimum hitting sets, the impact of the technique is expected to be marginal.

A fourth, and final optimization respects the clauses in H . Any $C_j \in H$ that $T \models C_j$, can also be removed from H , as it will not be included in any minimum cost hitting set. It should be noted that the gains of this technique should be also marginal. Since by construction $T \wedge S$ is consistent, and the only computed counterexamples satisfy $T \wedge S \wedge (\neg M)$, then any clause $C_j \in H$ with $T \models C_j$ will also be satisfied. Since, the counterexamples only consider falsified clauses, then any clause $C_j \in H$ entailed by T will *never* be included in a set to be hit.

3.3.2 Exponential Reductions in Oracle Calls

This section argues that the new Hyper algorithm for solving propositional abduction can save an exponentially larger number of iterations when compared with the AbHS/AbHS+ algorithm proposed in earlier work [58].

Consider a PAP $P_1 = (V, H, M, T, c)$, with:

$$\begin{aligned}
 V &= \{t_1, x_1, y_1, m_1, \dots, t_n, x_n, y_n, m_n\} \\
 H &= \{(\neg x_1), (x_1 \vee t_1), (\neg y_1), (y_1 \vee t_1), \dots, \\
 &\quad (\neg x_n), (x_n \vee t_n), (\neg y_n), (y_n \vee t_n)\} \\
 M &= \{(m_1), (m_2), \dots, (m_n)\} \\
 T &= \{(\neg t_1 \vee \neg t_2 \vee \dots \vee \neg t_n), \\
 &\quad (\neg t_1 \vee m_1), \dots, (\neg t_n \vee m_n)\}
 \end{aligned} \tag{11}$$

and $c(C_i) = 1$ for $C_i \in H$. Clearly, P_1 has no solution. For M to be entailed, S must imply all variables t_i to 1; but this causes $T \wedge S$ to become inconsistent. Moreover, there are exponentially many sets S , which are not consistent with T . AbHS+ will have to enumerate all of these sets S and, for each such set S , it will use one additional SAT oracle call to conclude that $T \wedge S$ is inconsistent. Since AbHS+ (or AbHS) selects all falsified clauses when blocking counterexamples of $T \wedge S \wedge (\neg M)$, all subsets of S inconsistent with T will be eventually enumerated. In contrast, since Hyper ensures consistency between S and T when selecting a minimum hitting set, this exponentially large number of oracle calls is not observed. (These differences between AbHS/AbHS+ and Hyper are experimentally validated in Section 4.)

It should be clear that the exponentially large reduction in the number of oracle calls obtained with Hyper are hidden in the minimum hitting set extractor. However, in Hyper the minimum hitting set extractor is based on MaxSAT (concretely core-guided MaxSAT), and so this hidden complexity is handled (most often efficiently) by the SAT solver.

The inclusion of M to find each set S can also potentially save exponentially many iterations. Consider the following PAP $P_2 =$

(V, H, M, T, c) :

$$\begin{aligned} V &= \{m, t_1, x_1, \dots, t_n, x_n\} \\ H &= \{(m \vee \neg x_1), (m \vee x_1 \vee t_1), \\ &\quad (m \vee \neg x_n), (m \vee x_n \vee t_n)\} \\ M &= \{(m)\} \\ T &= \{(\neg t_1 \vee \dots \vee \neg t_n)\} \end{aligned} \quad (12)$$

In contrast with the previous example, P_2 has a solution, i.e. there exists a subset S of H (with $S = H$) such that $T \wedge S \models M$. Until the solution is found, all computed models of $T \wedge S \wedge (\neg M)$ will also falsify $T \wedge S \wedge M$. Any of these models might be filtered out if any candidate set S is such that $T \wedge S \wedge M$ is consistent. It should be noted that in this case there is no formal guarantee that the number of SAT oracle calls must be exponential. This depends on the solutions provided by the minimum hitting set algorithm used. Essentially, taking M into account when selecting S guarantees that the picked set S will not be such that $T \wedge S \models \neg M$. As the results in the next section confirm, in practice AbHS/AbHS+ can generate exponentially many candidates S for which $T \wedge S \models \neg M$.

4 Experimental Results

4.1 Experimental Setup

All the conducted experiments were performed in Ubuntu Linux on an Intel Xeon E5-2630 2.60GHz processor with 64GByte of memory. The time limit was set to 1800s and the memory limit to 10GByte for each process to run. A prototype of the Hyper algorithm proposed above was implemented in C++ and consists of two interacting parts. One of them computes minimum size hitting sets of the set of counterexamples, also satisfying $T \wedge S \wedge M$. This is achieved with the use of an incremental implementation of the algorithm based on soft cardinality constraints [2, 47], which is a state-of-the-art MaxSAT algorithm that won several categories in the MaxSAT Evaluation 2015³. The other part of the prototype checks satisfiability of $T \wedge S \wedge (\neg M)$, where S is a candidate hitting set reported by the hitting set solver. Note that both parts of the solver were implemented on top of the well-known SAT solver Glucose 3.0⁴ [5]. Besides the basic version of Hyper, we also implemented an improved version, which is below referred to as Hyper* and contains the first two improvements described in Section 3.3.1. Namely, the first improvement does partial reduction of counterexamples by traversing and trying to satisfy $0.2 \times m$ clauses of each counterexample, where m is the size of the counterexample. The second improvement used in Hyper* consists in bootstrapping the hitting set solver with 100 MCSes of MaxSAT formula $T \wedge H \wedge (\neg M)$. It should be noted that bootstrapping the algorithm is not necessary but in some cases it can boost the performance of the main algorithm. Also note that the MaxSAT solver in both Hyper and Hyper* trims unsatisfiable cores [47] detected during the solving process at most 5 times.

Hyper and Hyper* were compared to recent state-of-the-art algorithms AbHS and AbHS+⁵ [58]. Additionally, we also implemented the QMaxHS approach described in Section 3.3. The implementation was done on top of DepQBF⁶, the known QBF solver which is capable of reporting unsatisfiable cores [42]. However, the performance of QMaxHS is poor (i.e. in our evaluation it did not solve any instance from the considered benchmark suite) and we decided to exclude it from consideration.

4.2 Abduction Problem Suite

In order to assess the efficiency of the new approach to propositional abduction, the following benchmark suite was used, which was proposed and also considered in [58]. According to [58], the benchmark instances were generated based on crafted and industrial instances from MaxSAT Evaluation 2014 with the use of the MaxSAT solver LMHS⁷ [57] and the backbone solver minibones⁸ [30]. The reader is referred to [58] for details. The resulting benchmark suite contains 6 benchmarks sets: *pms-5*, *pms-10*, *pms-15*, *wpms-5*, *wpms-10*, and *wpms-15*, where the number indicates the number of manifestations. In the conducted experimental evaluation, benchmark sets were aggregated based on their type (weighted or unweighted) and resulted in two benchmark sets: PMS and WPMS, having 847 and 795 instances, respectively. The total number of instances is 1642.

The cactus plots reporting the performance of the considered algorithms measured for the considered problem instances is shown in Figure 1. As one can see in Figure 1a, for PMS benchmark instances, Hyper and Hyper* perform significantly better than both AbHS and AbHS+. More precisely, Hyper* solves 321 instances (out of 847), which is 76 more (> 31%) than the number of instances solved by AbHS+ (245). The second best competitor is Hyper, which solves 318 instances being 3 instances behind Hyper*. Finally, the worst performance is shown by AbHS, which solves 174 instances. In contrast to the unweighted problem instances, the performance of the new algorithm for weighted instances is penalized by the use of MaxSAT for computing minimal hitting sets. The reason is that the MaxSAT solver used in Hyper does not exploit any modern and widely used heuristics to efficiently deal with weights, e.g. Boolean lexicographic optimization [43] or stratification [3].⁹ This can explain a similar performance shown by Hyper and Hyper* compared to AbHS+. More precisely, Hyper* is able to solve 398 instances (out of 795). AbHS+ comes second solving 389 instances while Hyper is 2 instances behind AbHS+ (387 solved). The worst performance is shown again by AbHS, which solves 252 instances.

Regarding the performance of the *virtual best solver* (VBS), the data for both sets of benchmarks can be seen in Figure 1 and it is the following. For PMS, the VBS aggregating Hyper, Hyper* as well as AbHS+¹⁰ is able to solve 328 instances, which is 7 more instances than what Hyper* can solve alone. In contrast, for the WPMS instances the picture is different: the VBS solves 425 instances, which is 27 and 36 more instances than what Hyper* and AbHS+ can solve separately. This indicates that Hyper* and AbHS+ complement each other in this case, which suggests building a portfolio of the solvers for weighted instances. The performance comparison between AbHS+ and Hyper* is detailed in the scatter plots shown in Figure 2 and also confirms this conclusion. Although the experimental results for the abduction problem suite show clear performance gain of the proposed Hyper algorithm over the state-of-the-art in propositional abduction (AbHS+), it is important to mention that the benchmark suite was generated (see [58]) from the MaxSAT instances by filtering out those of them that are hard for MaxHS-like MaxSAT solvers, i.e. MaxHS [13, 14] and LMHS [57]. This fact suggests that applying similar ideas for generating problem instances by targeting MaxSAT formulas that are easier for another family of

⁷ <http://www.cs.helsinki.fi/group/coreo/lmhs/>

⁸ <http://sat.inesc-id.pt/~mikolas/sw/minibones/>

⁹ This conjecture is also suggested by the average numbers of iterations done by Hyper* and AbHS+ for the WPMS benchmarks, which are 69 and 229, respectively. Since Hyper* does significantly fewer iterations (on average) and solves around the same number of instances as AbHS+ does, we assume that the calls to the MaxSAT oracle are harder (on average).

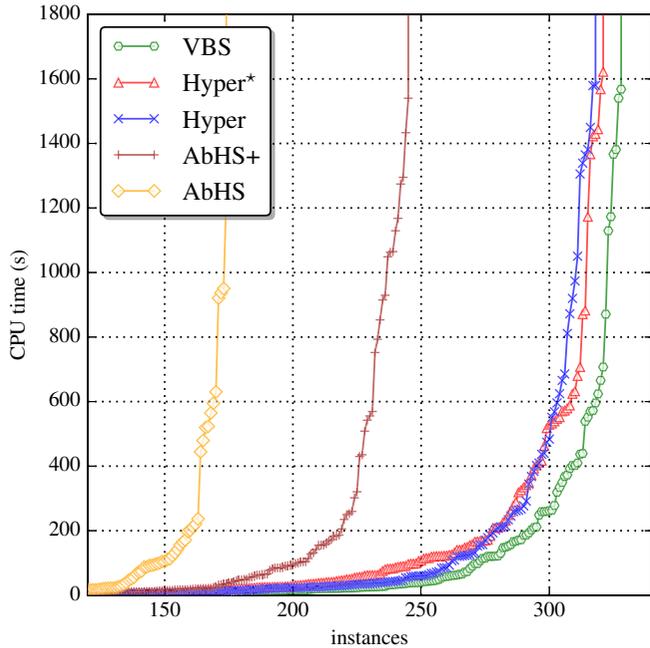
¹⁰ AbHS is excluded from the VBS since it does not contribute to its performance.

³ See results for MSCG15b at <http://www.maxsat.udl.cat/15/>

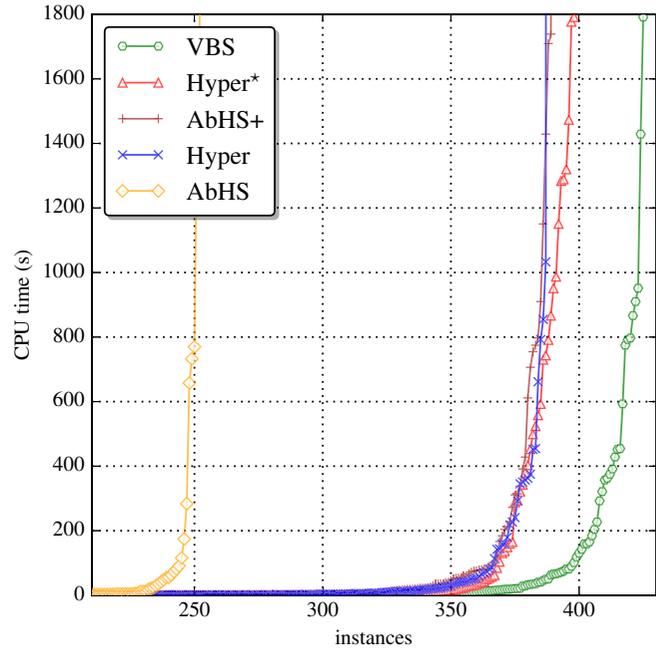
⁴ <http://www.labri.fr/perso/lisimon/glucose>

⁵ <http://cs.helsinki.fi/group/coreo/abhs/>

⁶ <http://lonsing.github.io/depqbf/>

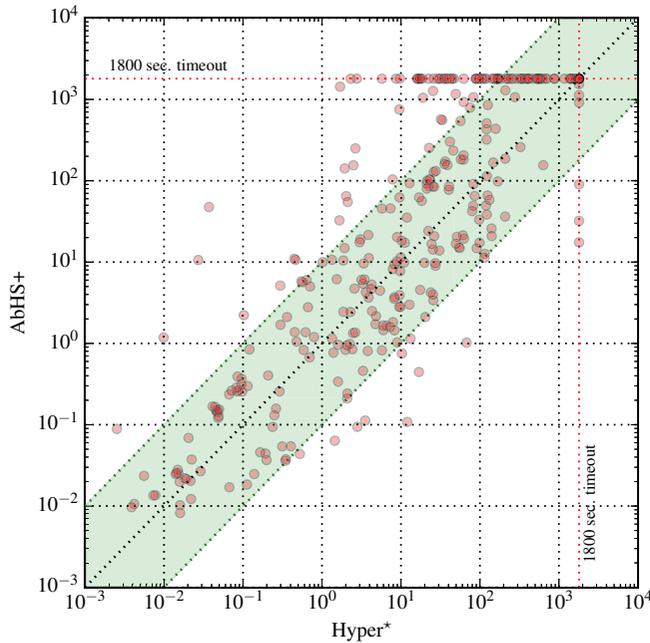


(a) PMS instances

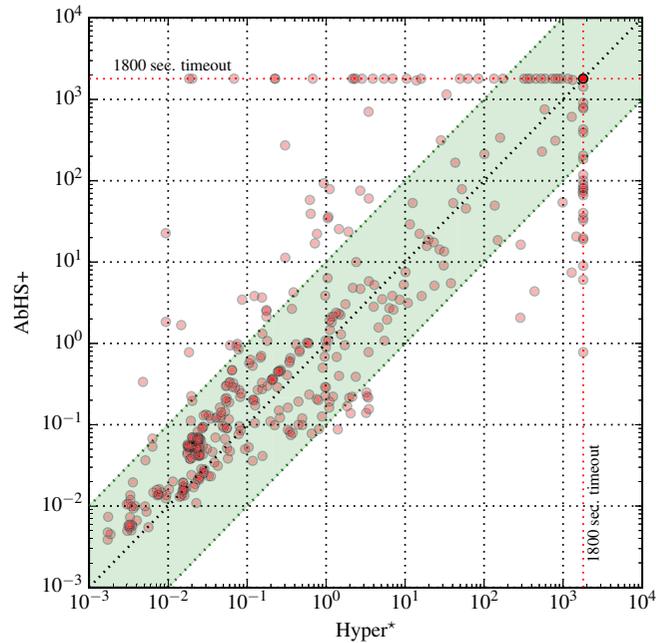


(b) WPMS instances

Figure 1: Performance of Hyper, Hyper*, AbHS, and AbHS+ for the Abduction Problem Suite benchmarks.



(a) PMS instances



(b) WPMS instances

Figure 2: Hyper* vs AbHS+.

MaxSAT algorithms, e.g. the core-guided algorithms based on soft cardinality constraints [47] (recall that the MaxSAT solver in Hyper is one of them), would result in even better performance of Hyper. Moreover, the results for the weighted benchmarks emphasize the importance of applying modern techniques (e.g. Boolean lexicographic optimization and stratification). With such improvements, we expect Hyper to perform significantly better for problem instances

with weights.

4.3 Oracle Calls in AbHS+

This section studies the number of iterations for the considered approaches for the families of examples described in Section 3.3.2. For both examples (11) and (12) we generated 10 instances varying size

Table 1: The number of iterations and running time per solver for example family (11). Additionally, for AbHS/AbHS+ the number of iterations of type 2 is also shown (in parentheses). Value n varies from 1 to 10.

	1	2	3	4	5	6	7	8	9	10
AbHS	11 (7)	59 (49)	363 (343)	2401 (829)	—	—	—	—	—	—
	0.0s	0.1s	3.4s	166.2s	>1800s	>1800s	>1800s	>1800s	>1800s	>1800s
AbHS+	6 (2)	14 (4)	28 (8)	51 (16)	125 (32)	388 (64)	978 (128)	2242 (256)	—	—
	0.0s	0.0s	0.0s	0.0s	0.3s	2.8s	24.3s	180.3s	>1800s	>1800s
Hyper	6	14	17	19	27	32	32	35	39	48
	0.0s	0.0s	0.0s	0.0s	0.0s	0.0s	0.0s	0.0s	0.0s	0.0s

Table 2: The number of iterations and running time per solver for example family (12). Value n varies from 1 to 10.

	1	2	3	4	5	6	7	8	9	10
AbHS /	5	11	21	54	133	350	878	1995	—	—
AbHS+	0.0s	0.1s	0.1s	0.2s	0.3s	2.0s	15.5s	168.8s	>1800s	>1800s
Hyper	6	16	17	14	20	29	18	27	30	37
	0.0s	0.0s	0.0s	0.0s						

n of the instance from 1 to 10 in order to show how the number of iterations grows with the growth of size n for each approach.¹¹

Recall that example (11) aims at showing the importance of adding the theory clauses into the hitting set solver by saving an exponential number of iterations related to candidates that are not consistent with the theory. In AbHS/AbHS+ the consistency check is done through the second SAT call and results in a counterexample blocking the candidate (AbHS+ also blocks its supersets). The idea of proposing example family (11) is, thus, to show that the number of iterations of this type (let us call them iterations of type 2 because they are related with the 2nd SAT call) in AbHS and AbHS+ can be exponentially larger than the number of iterations done by Hyper¹². Indeed, Table 1 confirms this conjecture indicating that the number of iterations of type 2 in AbHS+ grows exponentially with the growth of n , i.e. it is exactly 2^n (see the values in parentheses), while the number of iterations done by Hyper is negligible. The situation gets even more dramatic for AbHS. (As one can see, the total number of iterations performed by AbHS and AbHS+ grows even faster.) The running time spent by AbHS and AbHS+ also grows significantly with the growth of n . As a result, AbHS and AbHS+ cannot solve any instances for $n > 4$ and $n > 8$, respectively, within 1800 seconds. Observe that Hyper reports the result for each n immediately.

Regarding example (12), it shows the importance of adding M into the hitting set solver. Table 2 confirms that it can save an exponential number of iterations. As one can observe, the number of iterations

grows exponentially with growing value n for AbHS and AbHS+. Note that in this case AbHS and AbHS+ behave similarly to each other, which is why Table 2 does not have a separate row for AbHS+. Analogously to the previous case, the performance of the solver (i.e. its running time) is severely affected by the number of iterations. Analogously to the previous example and in contrast to AbHS and AbHS+, the basic version of Hyper does significantly fewer iterations and spends almost no time for each of the considered instances.

5 Conclusions

Abduction finds many applications in Artificial Intelligence, with a large body of work over the years. Recent work investigated propositional abduction, and proposed the use of a variant of the implicit hitting set algorithm MaxHS for solving the problem [58].

This paper identifies several sources of inefficiency with earlier work, and proposes a novel, implicit hitting set inspired, algorithm for propositional abduction. The novel algorithm, Hyper, is shown to outperform the recently proposed algorithms AbHS and AbHS+ on existing problem instances. In addition, the paper demonstrates that the proposed improvements can result in exponential savings on the number of SAT oracle calls, which helps explain the observed performance improvements.

In a broader context, this paper contributes to the recent body of work on implicit hitting set algorithms, and identifies algorithmic optimizations that can be significant in other contexts.

A number of research directions can be envisioned. These include improvements to the MaxSAT solver used for computing minimum hitting sets, as this represents the main bottleneck of the Hyper algorithm. Additional work will involve applying Hyper to a larger range of problem instances.

Acknowledgments

This work was partially supported by FCT from MCTES Portugal through the scholarship with reference SFRH/BPD/103609/2014.

¹¹ It should be noted that the pseudo-code in Algorithm 2 (taken from [58]), as well as the actual source code, needs to be modified for AbHS+ to produce correct results when a PAP does not have a solution, as is the case with example (11). When blocking previously computed hitting sets, AbHS+ can generate clauses both with positive literals and clauses with negative literals and, for a PAP without solution, it will eventually compute an empty hitting set, denoting that there is no solution to the constraints added as sets to hit. As a result, the pseudo-code (and the source) needs to test for the case when the minimum hitting set returned is empty, in which case it must return ‘no solution’. This fix was added to AbHS+ to get the results presented in this section.

¹² To test the number of iterations, the basic version Hyper was considered.

References

- [1] Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni, 'Security protocols verification in abductive logic program.: A case study', in *ESAW*, pp. 106–124, (2005).
- [2] Benjamin Andres, Benjamin Kaufmann, Oliver Matheis, and Torsten Schaub, 'Unsatisfiability-based optimization in clasp', in *ICLP*, pp. 211–221, (2012).
- [3] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy, 'SAT-based MaxSAT algorithms', *Artif. Intell.*, **196**, 77–105, (2013).
- [4] M. Fareed Arif, Carlos Mencía, and Joao Marques-Silva, 'Efficient MUS enumeration of horn formulae with applications to axiom pinpointing', in *IJCAI*, pp. 324–342, (2015).
- [5] Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon, 'Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction', in *SAT*, pp. 309–317, (2013).
- [6] James Bailey and Peter J. Stuckey, 'Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization', in *PADL*, pp. 174–186, (2005).
- [7] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, eds. *Handbook of Satisfiability*. IOS Press, 2009.
- [8] Elazar Birnbaum and Eliezer L. Lozinskii, 'Consistent subsets of inconsistent systems: structure and behaviour', *J. Exp. Theor. Artif. Intell.*, **15**(1), 25–46, (2003).
- [9] Tom Bylander, Dean Allemang, Michael C. Tanner, and John R. Josephson, 'The computational complexity of abduction', *Artif. Intell.*, **49**(1-3), 25–60, (1991).
- [10] Karthekeyan Chandrasekaran, Richard M. Karp, Erick Moreno-Centeno, and Santosh Vempala, 'Algorithms for implicit hitting set problems', in *SODA*, pp. 614–629, (2011).
- [11] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani, 'A modular approach to MaxSAT modulo theories', in *SAT*, pp. 150–165, (2013).
- [12] Nadia Creignou and Bruno Zanuttini, 'A complete classification of the complexity of propositional abduction', *SIAM J. Comput.*, **36**(1), 207–229, (2006).
- [13] Jessica Davies and Fahiem Bacchus, 'Solving MAXSAT by solving a sequence of simpler SAT instances', in *CP*, pp. 225–239, (2011).
- [14] Jessica Davies and Fahiem Bacchus, 'Postponing optimization to speed up MAXSAT solving', in *CP*, pp. 247–262, (2013).
- [15] Isil Dillig and Thomas Dillig, 'Explain: A tool for performing abductive inference', in *CAV*, pp. 684–689, (2013).
- [16] Isil Dillig, Thomas Dillig, and Alex Aiken, 'Automated error diagnosis using abductive inference', in *PLDI*, pp. 181–192, (2012).
- [17] Isil Dillig, Thomas Dillig, Boyang Li, and Kenneth L. McMillan, 'Inductive invariant generation via abductive inference', in *OOPSLA*, pp. 443–456, (2013).
- [18] Thomas Eiter and Georg Gottlob, 'The complexity of logic-based abduction', *J. ACM*, **42**(1), 3–42, (1995).
- [19] Thomas Eiter and Kazuhisa Makino, 'Abduction and the dualization problem', in *Discovery Science*, pp. 1–20, (2003).
- [20] Béchir el Ayeb, Pierre Marquis, and Michaël Rusinowitch, 'Preferring diagnoses by abduction', *IEEE Trans. Systems, Man, and Cybernetics*, **23**(3), 792–808, (1993).
- [21] Marco Gavanelli, Marco Alberti, and Evelina Lamma, 'Integrating abduction and constraint optimization in constraint handling rules', in *ECAI*, pp. 903–904, (2008).
- [22] Marco Gavanelli, Marco Alberti, and Evelina Lamma, 'Integrat. of abductive reason. and const. opt. in SCIFF', in *ICLP*, pp. 387–401, (2009).
- [23] Alexey Ignatiev, Mikolás Janota, and Joao Marques-Silva, 'Quantified MaxSAT: A core-guided approach', in *SAT*, pp. 250–266, (2013).
- [24] Alexey Ignatiev, Mikolás Janota, and Joao Marques-Silva, 'Quantified MaxSAT', *Constraints*, **21**(2), 277–302, (2016).
- [25] Katsumi Inoue, 'Automated abduction', in *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, pp. 311–341, (2002).
- [26] Katsumi Inoue, Andrei Doncescu, and Hidetomo Nabeshima, 'Completing causal networks by meta-level abduction', *Machine Learning*, **91**(2), 239–277, (2013).
- [27] Katsumi Inoue, Taisuke Sato, Masakazu Ishihata, Yoshitaka Kameya, and Hidetomo Nabeshima, 'Evaluating abductive hypotheses using an EM algorithm on BDDs', in *IJCAI*, pp. 810–815, (2009).
- [28] Mikolás Janota, William Klieber, Joao Marques-Silva, and Edmund M. Clarke, 'Solving QBF with counterexample guided refinement', in *SAT*, pp. 114–128, (2012).
- [29] Mikolás Janota, William Klieber, Joao Marques-Silva, and Edmund M. Clarke, 'Solving QBF with counterexample guided refinement', *Artif. Intell.*, **234**, 1–25, (2016).
- [30] Mikolás Janota, Inês Lynce, and Joao Marques-Silva, 'Algorithms for computing backbones of propositional formulae', *AI Commun.*, **28**(2), 161–177, (2015).
- [31] Mikolás Janota and Joao Marques-Silva, 'Abstraction-based algorithm for 2QBF', in *SAT*, pp. 230–244, (2011).
- [32] Mikolás Janota and Joao Marques-Silva, 'Solving QBF by clause selection', in *IJCAI*, pp. 325–331, (2015).
- [33] Eugene Santos Jr., 'A linear constraint satisfaction approach to cost-based abduction', *Artif. Intell.*, **65**(1), 1–28, (1994).
- [34] Antonis C. Kakas, Robert A. Kowalski, and Francesca Toni, 'Abductive logic programming', *J. Log. Comput.*, **2**(6), 719–770, (1992).
- [35] Antonis C. Kakas and Paolo Mancarella, 'Database updates through abduction', in *VLDB*, pp. 650–661, (1990).
- [36] Antonis C. Kakas and Fabrizio Riguzzi, 'Learning with abduction', in *ILP*, pp. 181–188, (1997).
- [37] Richard M. Karp, 'Implicit hitting set problems and multi-genome alignment', in *CPM*, p. 151, (2010).
- [38] Mark H. Liffiton and Ammar Malik, 'Enumerating infeasibility: Finding multiple MUSes quickly', in *CPAIOR*, pp. 160–175, (2013).
- [39] Mark H. Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva, 'Fast, flexible MUS enumeration', *Constraints*, **21**(2), 223–250, (2016).
- [40] Mark H. Liffiton and Karem A. Sakallah, 'Algorithms for computing minimal unsatisfiable subsets of constraints', *J. Autom. Reasoning*, **40**(1), 1–33, (2008).
- [41] Fangzhen Lin and Jia-Huai You, 'Abduction in logic programming: A new definition and an abductive procedure based on rewriting', *Artif. Intell.*, **140**(1/2), 175–205, (2002).
- [42] Florian Lonsing and Uwe Egly, 'Increment. comput. min. unsat. cores of QBFs via a clause group solver API', in *SAT*, pp. 191–198, (2015).
- [43] Joao Marques-Silva, Josep Argelich, A. Sofia Graca, and Inês Lynce, 'Boolean lexicographic optimization: algorithms & applications', *AMAI*, **62**(3-4), 317–343, (2011).
- [44] Joao Marques-Silva, Federico Heras, Mikolás Janota, Alessandro Previti, and Anton Belov, 'On computing min. cs', in *IJCAI*, (2013).
- [45] Erick Moreno-Centeno and Richard M. Karp, 'The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment', *OR*, **61**(2), 453–468, (2013).
- [46] António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and Joao Marques-Silva, 'Iterative and core-guided maxsat solving: A survey and assessment', *Constraints*, **18**(4), 478–534, (2013).
- [47] Antonio Morgado, Alexey Ignatiev, and Joao Marques-Silva, 'MSCG: Robust core-guided MaxSAT solving', *JSAT*, **9**, 129–134, (2015).
- [48] António Morgado, Mark H. Liffiton, and Joao Marques-Silva, 'MaxSAT-based MCS enumeration', in *HVC*, pp. 86–101, (2012).
- [49] Gustav Nordh and Bruno Zanuttini, 'What makes propositional abduction tractable', *Artif. Intell.*, **172**(10), 1245–1284, (2008).
- [50] Andreas Pfandler, Reinhard Pichler, and Stefan Woltran, 'The complexity of handling minimal solutions in logic-based abduction', *J. Log. Comput.*, **25**(3), 805–825, (2015).
- [51] Reinhard Pichler and Stefan Woltran, 'The complexity of handling minimal solutions in logic-based abduction', in *ECAI*, pp. 895–900, (2010).
- [52] David Poole, 'Probabilistic horn abduction and bayesian networks', *Artif. Intell.*, **64**(1), 81–129, (1993).
- [53] Alessandro Previti and Joao Marques-Silva, 'Partial MUS enumeration', in *AAAI*, (2013).
- [54] Oliver Ray and Katsumi Inoue, 'A consequence finding approach for full clausal abduction', in *Discovery Science*, pp. 173–184, (2007).
- [55] James A Reggia, Barry T Perricone, Dana S Nau, and Yun Peng, 'Answer justification in diagnostic expert systems-Part I: Abductive inference and its justification', *IEEE T. Bio. Eng.*, (4), 263–267, (1985).
- [56] Raymond Reiter, 'A theory of diagnosis from first principles', *Artif. Intell.*, **32**(1), 57–95, (1987).
- [57] Paul Saikko, Jeremias Berg, and Matti Järvisalo, 'LMHS: A SAT-IP hybrid MaxSAT solver', in *SAT*, (2016).
- [58] Paul Saikko, Johannes P. Wallner, and Matti Järvisalo, 'Implicit hitting set algorithms for reasoning beyond NP', in *KR*, (2016).
- [59] Chiaki Sakama and Katsumi Inoue, 'An abductive framework for computing knowledge base updates', *TPLP*, **3**(6), 671–713, (2003).
- [60] Ken Satoh and Takeaki Uno, 'Enumerating minimal explanations by minimal hitting set computation', in *KSEM*, pp. 354–365, (2006).
- [61] Roni Tzvi Stern, Meir Kalech, Alexander Feldman, and Gregory M. Provan, 'Exploring the duality in conflict-directed model-based diagnosis', in *AAAI*, (2012).

Improved Multi-Label Classification Using Inter-Dependence Structure via a Generative Mixture Model

Ramanuja Simha¹ and Hagit Shatkay¹

Abstract. Single-label classification associates each instance with a single label, while multi-label classification (MLC), assigns *multiple* labels to instances. Simple MLC systems assume that labels are independent of one another, while more complex approaches capture inter-dependencies among labels. Experiments comparing performance of MLC systems demonstrate that there is much room for improvement.

Notably, when an instance is associated with multiple labels, a feature-value of the instance may depend only on a subset of these labels and thus be *conditionally independent* of the others given the label-subset. Current systems do not account for such conditional independence. Moreover, dependence of a feature-value on a label is likely to imply its dependence on other inter-dependent labels. Our hypothesis is that by explicitly modeling the dependence between feature values and specific subsets of *inter-dependent* labels, the assignment of multi-labels to instances can be done more accurately.

We present a probabilistic generative model that captures dependencies among labels as well as between features and labels, by means of a Bayesian network. We introduce the concept of *label dependency sets* as a basis for a new mixture model that represents conditional independencies between features and labels given subsets of inter-dependent labels. Experimental results show that the performance of the system we have developed based on our model for MLC significantly improves upon results obtained by current MLC systems that are based on probabilistic models.

1 Introduction

Multi-label classification (MLC) associates instances with possibly multiple labels, in contrast to *single-label classification*, where each instance is associated with a single label. Simple approaches for multi-label classification transform the task into one or more single-label classification task(s). For instance, under the Ranking by Pairwise Comparison method [22], a classifier is trained to distinguish between each possible pair of labels (one-vs-one). A computationally efficient alternative is the Binary Relevance method [22], where each classification task corresponds to distinguishing a single label from the rest (one-vs-all).

More advanced approaches for multi-label classification capture dependencies among labels. For example, Multi-Label Search [8] explores a search space of label sets to capture such dependencies while learning a mapping of instances to multi-labels. A more widely used

method is a Classifier Chain [14, 15], which consists of multiple binary classifiers like those used in Binary Relevance, one classifier per label. The chain is constructed based on an input label ordering. To capture relationships among labels, the *feature-vector* used to represent an input instance given to a classifier F includes label assignments obtained from all classifiers preceding classifier F in the chain. Systems based on Classifier Chains include probabilistic variants [5, 6, 16], and others that explicitly learn label inter-dependencies such as a chain of Support Vector Machines [26], a chain of naïve Bayes classifiers [25], and an ensemble of Bayesian networks [1]. Other approaches that employ graphical models, however not based on Classifier Chains, include Conditional Dependency Networks [10], which use a fully-connected graphical model, and systems that utilize probabilistic generative models [13, 17, 23], typically built for classifying text. The latter class of systems have not been extensively tested against other MLC systems and on datasets other than text.

In the context of MLC, a feature-value of an instance typically depends on some subset of the instance labels and thus may be *conditionally independent* of the other labels given this subset. For example, the *grade* feature value of college students who are classified (labeled) as *Excelled in entrance tests* and *Admitted into a graduate program* is typically *High*, regardless of any other student labels. Furthermore, dependence of a feature-value on a label is likely to suggest its dependence on other inter-dependent labels. Current systems do not account for the conditional independence between a feature-value and other labels given a subset of labels. Moreover, performance of current methods leaves much room for improvement. Our hypothesis is that explicitly modeling the dependencies between feature values and *inter-dependent* labels, as part of the classifier model, can support a more accurate assignment of multi-labels to instances.

We present a probabilistic generative model that captures dependencies among labels as well as between features and labels, by means of a Bayesian network. We introduce a mixture model to represent conditional independencies between features and labels given subsets of *inter-dependent* labels, and further develop a multi-label classifier. Unlike previous approaches, our system uses an iterative process to infer values for multiple labels simultaneously. In each iteration, the Bayesian network is modified to reflect inter-dependencies among the *most recently inferred label values*; the accuracy of the updated label assignments is thus improved by capturing specific feature-label correlations and dependencies. We evaluate our system on several multi-label datasets used before for evaluating MLC systems, and demonstrate that the performance of our system improves upon that obtained by current Classifier-Chain systems.

¹ Computational Biomedicine and Machine Learning lab, Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716 USA. Email: {rsimha, shatkay}@udel.edu

In the next section we introduce relevant notations and present our probabilistic generative model. Section 3 discusses procedures used for learning structure and parameters of the model, and inference techniques applied for multi-label classification. Section 4 provides details about the multi-label datasets we use, performance evaluation measures, and experimental results, while Section 5 concludes and summarizes our findings and outlines future directions.

2 A Dependency-Based Mixture Model for Multi-Label Data

Let D be a dataset containing m instances, and $C = \{c_1, \dots, c_q\}$ be a set of q class-labels. Each instance in D is associated with a subset of labels. As others have done before in the context of multi-label classification (MLC) [1, 5], we represent an instance $I \in D$ as a feature vector, $\vec{f}^I = \langle f_1^I, \dots, f_d^I \rangle$, and I 's labels as a label vector, $\vec{l}^I = \langle l_1^I, \dots, l_q^I \rangle$. Here d is the number of features, and $l_i^I = 1$ if instance I is associated with label c_i , $l_i^I = 0$ otherwise. Each feature value f_j is viewed as a value taken by a feature random variable F_j , and each label-indicator value l_i is viewed as a value taken by a label random variable L_i . The task of multi-label classification thus amounts to developing a classifier that takes as input an instance represented by a feature vector, and outputs a q -dimensional label vector.

2.1 Model Framework

We use a Bayesian network framework to model inter-dependencies among labels as well as between features and labels. Each node represents either a label variable L_i ($1 \leq i \leq q$) or a feature variable F_j ($1 \leq j \leq d$), and each directed edge indicates a dependence relationship between a pair of variables.

Representing label inter-dependencies

In the context of multi-label classification, labels may be directly correlated with one another, regardless of their association with any specific instance. As a simple example, drivers that are labeled as *Speeding* are also likely to be labeled *Accident-Prone*, regardless of any specific driver characteristics (features). We represent each *unconditional dependence* between a pair of labels c_i and c_j as a directed edge from label variable L_i to variable L_j . As another example, Figure 1 shows the more complex inter-dependency structure among label variables that we learn as part of our experiments (see Section 4) in the context of the *Emotions* dataset [21]; in this example, instances are songs and labels are emotions. A directed edge, e.g. from *Amazed-Surprised* to *Sad-Lonely*, represents the assertion that knowing that an instance is associated with the label *Amazed-Surprised* influences the level of belief about the instance's association with the label *Sad-Lonely*.

A label may often depend on a small set of a few labels while being conditionally independent of other labels given this set. To continue

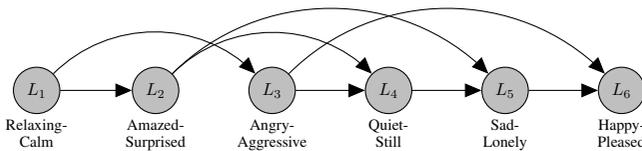


Figure 1: An example Bayesian network structure over labels that we learn using the *Emotions* dataset [21].

the previous simple example, the label *Accident-prone* is conditionally independent of the label *New-driver* given the label *Speeding*. To capture such *conditional independencies*, we introduce the concept of *label dependency sets*. A *dependency set* for a label c_i is a minimal set of labels, c_{i_1}, \dots, c_{i_m} such that knowing an instance's association with each label c_{i_j} in the set is sufficient to infer the likelihood of the instance to be associated with c_i . We utilize the Bayesian network framework to obtain a practical representation of a label dependency set. The network structure captures both direct dependencies between pairs of labels and conditional independencies among labels given certain subsets of them. More specifically, each label variable L_i directly depends on its parents $\text{Pa}(L_i)$ while being conditionally independent of its non-descendants given $\text{Pa}(L_i)$; the joint distribution of the label variables is thus given by:

$$\Pr(L_1, \dots, L_q) = \prod_{i=1}^q \Pr(L_i | \text{Pa}(L_i)).$$

Employing the above conditional independence, we refer to a label variable L_i and its parents in the network as the *label dependency set* for L_i . Thus, for each variable L_i ($1 \leq i \leq q$), we define a label dependency set: $LS_i = \{L_i\} \cup \text{Pa}(L_i)$.

Representing dependencies between features and labels.

An instance's association with certain labels is clearly correlated with the instance feature values. Additionally, the value of a feature may be correlated with multiple labels and not just with one. For example, in the *Emotions* dataset [21], where instances are songs represented using *rhythm* and *tone* features and labels are *emotions*, the value of the tone feature is typically *Low* when a song is labeled as *Sad-Lonely* while it is typically *High* when a song is simultaneously labeled as both *Sad-Lonely* and *Amazed-Surprised*.

As explained earlier while introducing label dependency sets (LDS), the association of an instance with certain labels typically provides information about its association with other labels. For instance, a song that has a *High* tone feature value and is labeled as *Amazed-Surprised* is likely to also be labeled as *Sad-Lonely*. We represent the *dependence* of a feature, F_j , on a subset of inter-dependent labels, LS_i (to which we refer as a label dependency set) by plotting directed edges connecting each label variable in the set with the feature variable. We thus capture the *conditional dependence among labels* in the set LS_i given the feature F_j . Figure 2 extends the ex-

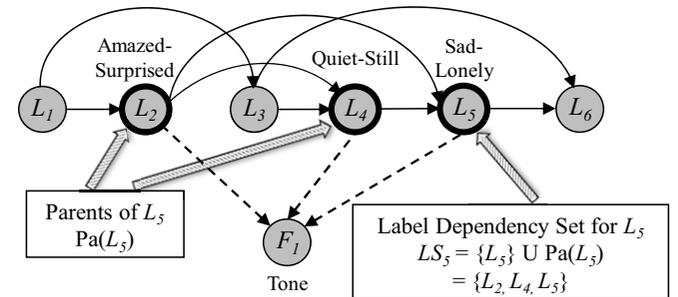


Figure 2: An extension of the network shown in Figure 1, where labels are emotions and features are rhythms/tones. The feature associated with the variable F_1 is tone. Bold-faced nodes (L_5 and its parents L_2 and L_4) form the *label dependency set*, LS_5 . Solid directed edges represent dependencies among the labels while dashed edges represent dependence between the feature, F_1 , and the label dependency set, LS_5 .

ample Bayesian network presented in Figure 1. The dashed arrows from labels L_2 , L_4 , and L_5 (i.e., *Amazed-Surprised*, *Quiet-Still*, and *Sad-Lonely*) to the feature, F_1 (i.e. *Tone*) in Figure 2 capture the dependence of the feature on the subset of inter-dependent labels comprising the label L_5 and its parents, L_2 and L_4 ; this label subset is referred to as the *label dependency set*, $LS_5 = \{L_5\} \cup Pa(L_5)$.

Moreover, when an instance is associated with multiple labels, a feature-value of the instance may depend only on a subset of these labels. As an example, the *tone* feature value of a song that is labeled as *Sad-Lonely* and *Amazed-Surprised* is likely to be *High* regardless of the song's association with any other labels. That is, the *tone* is conditionally independent of all other labels, given the two labels *Sad-Lonely* and *Amazed-Surprised*. By explicitly representing *dependence* between a feature and the labels in an LDS as discussed above, our model captures *conditional independence* of the feature from all other labels given the LDS.

We next present a probabilistic generative model that captures the label inter-dependencies and dependencies between features and labels discussed above.

2.2 Model Description

Generative models have been used before for multi-label classification [13, 17, 23], typically for classifying text. While these models address dependencies among labels, they do not represent intricate dependencies between feature values and subsets of labels. In contrast, our proposed model captures conditional independencies of features from labels by directly representing the dependencies between feature values and label subsets. (In addition, our model is developed in the general context of multi-label classification — not limited to text.) We next discuss the *instance generation process*, based on a Bayesian network structure, and provide further detail about our model.

The generation process comprises two steps:

- I **Labels assignment:** To generate an instance I , its class-labels are first determined, i.e., a label value l_i^I is assigned to each label variable L_i ($1 \leq i \leq q$). We view each label assignment as a *Bernoulli* event, where $l_i^I=1$ when I is associated with the label c_i , and $l_i^I=0$ otherwise. Based on the Bayesian network structure, the conditional probability of L_i to be assigned 1 given the values of its parents denoted $\mathcal{V}_{Pa(L_i)}$ is denoted as: $\alpha_i = \Pr(L_i=1|\mathcal{V}_{Pa(L_i)})^2$ while its probability to be assigned 0 is $1-\alpha_i$. The order in which label values are assigned is based on the topological order of label variables, L_{t_1}, \dots, L_{t_q} in the Bayesian network. The assigned label values form a label-vector \vec{l}^I for instance I .
- II **Features assignment:** Based on the label-vector \vec{l} , a label dependency set LS_{F_j} is selected for each feature F_j . We expect this set to constitute a small subset of labels such that F_j 's value depends only on the label subset. We introduce a *Multinomial* random variable λ^{F_j} that takes on a value $k \in \{1, \dots, q\}$ with probability: $\theta_{j,k}^{\vec{l}} = \Pr(\lambda^{F_j} = k|\vec{l})$; $\lambda^{F_j} = k$ indicates the selection of the k 'th label dependency set. We denote this set as: $LS_k = \{L_k\} \cup Pa(L_k)$. We refer to $\theta_{j,k}^{\vec{l}}$ as the *mixture parameter* as it models the dependence between the labels in each label dependency set, LS_k and the value of feature F_j .

The value for feature F_j is thus selected based on the values taken by the random variables in the set LS_k , denoted as

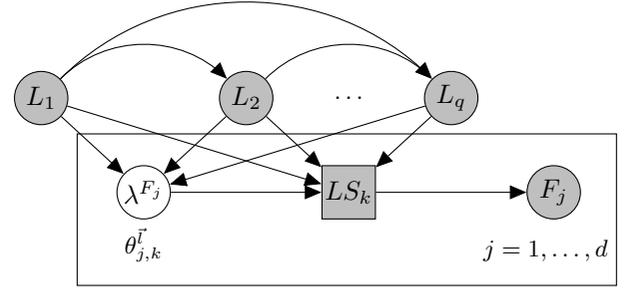


Figure 3: Bayesian network representing the generative mixture model of multi-label data.

\mathcal{V}_{LS_k} . We view each feature value selection as a *Multinomial* event, where the probability of F_j to take on a value v is: $\phi_{j,k}(v) = \Pr(F_j=v|\lambda^{F_j}=k, \mathcal{V}_{LS_k})$. Here $\phi_{j,k}(v)$ denotes the conditional probability of feature F_j to take on the value v given the label dependency set LS_k . In the model, all feature variables are assumed to take on discrete values. We thus discretize each real-valued feature in the datasets used for our experiments, as done by earlier studies [1] (see Section 4 for further details regarding discretization). The selected values for all features together form a complete feature-vector \vec{f}^I representing the instance I .

We note that the process of assigning feature values enforces several independence relationships. Having selected LS_k as the label dependency set, we denote by \bar{L}_k the set of all label variables other than $\{L_k\} \cup Pa(L_k)$, i.e., $\bar{L}_k = \{L_1, \dots, L_q\} - LS_k$. The value selection — for each feature F_j — is *conditionally independent* of all labels in \bar{L}_k given LS_k . Furthermore, the selected feature-value for F_j is also *conditionally independent* of all other feature values given the label vector \vec{l}^I .

Formally stated, the *independence assumptions* enforced by our model are:

- (i) The feature values f_1^I, \dots, f_d^I of an instance I , are *conditionally independent* of each other given the instance's label vector \vec{l}^I :

$$\Pr(\vec{f}^I|\vec{l}^I) = \prod_{j=1}^d \Pr(f_j^I|\vec{l}^I). \quad (1)$$

Although this assumption over-simplifies feature inter-dependencies, it has been proven effective [1, 25].

- (ii) Given the values taken by a label variable L_k and its parents $Pa(L_k)$ in a *selected* label dependency set LS_k for an instance I , a value f_j^I of a feature is *conditionally independent* of all other labels of I :

$$\begin{aligned} \Pr(F_j = f_j^I | \lambda^{F_j} = k, L_1 = l_1^I, \dots, L_q = l_q^I) &= \\ &= \Pr(F_j = f_j^I | \lambda^{F_j} = k, L_k = l_k^I, \mathcal{V}_{Pa(L_k)}) . \end{aligned} \quad (2)$$

Figure 3 shows a graphical representation of the generative mixture model presented above. Nodes represent random variables, and directed edges represent dependencies among variables. Label and feature random variables are denoted as circles. In contrast, the label dependency set, LS_k is denoted by a square as its value is deterministically assigned based on values taken by the label variable L_k and its parents $Pa(L_k)$. Notably, the node for LS_k represents a set, and as such the directed edge from LS_k to the feature F_j is a short-hand

² Throughout the paper, for any set, S , of random variables, we denote by \mathcal{V}_S the values taken by the variables in this set.

for multiple edges connecting each label variable in LS_k with F_j as described in Section 2.1. Variables representing labels and features are *observed*, that is, their values are provided within the training dataset. These variables are shown as shaded in the figure. The value of the variable λ^{F_j} is governed by the mixture parameter θ_j^I and is not given as part of the dataset. As such, it is *latent* and shown as unshaded.

Under all the above mentioned independence assumptions and based on the structure of our generative model, the joint probability of the label vector \vec{l}^I and the feature vector \vec{f}^I is expressed as:

$$\begin{aligned} \Pr(\vec{l}^I, \vec{f}^I) &= \Pr(\vec{l}^I) \Pr(\vec{f}^I | \vec{l}^I) = \\ &= \prod_{i=1}^q \Pr(L_i = l_i^I | \mathcal{V}_{\text{Pa}(L_i)}) \times \\ &\quad \times \prod_{j=1}^d \sum_{k=1}^q \theta_{j,k}^{\vec{l}^I} \Pr(F_j = f_j^I | \lambda^{F_j} = k, L_k = l_k^I, \mathcal{V}_{\text{Pa}(L_k)}), \end{aligned} \quad (3)$$

where:

- (a) $\prod_{i=1}^q \Pr(L_i = l_i^I | \mathcal{V}_{\text{Pa}(L_i)})$ is the factorization of the joint probability $\Pr(\vec{l}^I) = \Pr(L_1 = l_1^I, \dots, L_q = l_q^I)$, based on the individual q label values, given the conditional independencies encoded in the network;
- (b) $\Pr(F_j = f_j^I | \lambda^{F_j} = k, L_k = l_k^I, \mathcal{V}_{\text{Pa}(L_k)})$ denotes the conditional probability of a feature value f_j^I ($1 \leq j \leq d$, where d is the total number of features), given the values taken by a label variable L_k and its parents $\text{Pa}(L_k)$; $L_k \cup \text{Pa}(L_k)$ comprises the label dependency set LS_k (under the current generative mixture model);
- (c) $\theta_{j,k}^{\vec{l}^I}$ denotes the probability that the label dependency set LS_k is selected given a label-vector \vec{l}^I for a feature F_j .

3 Model Learning and Inference

We next introduce a procedure for learning the structure and the parameters of our generative model, and present an inference technique for multi-label classification.

3.1 Structure and Parameter Learning

We employ an iterative procedure to learn the Bayesian network structure, specifically the structure of inter-dependencies among the label nodes shown at the top of Figure 3, and to estimate the model parameters shown on the RHS of Equation 3. This iterative procedure is summarized in the pseudocode shown in Figure 4.

In each iteration, we first learn a label inter-dependency structure using the BANJO package [20]; we then estimate the model parameters through an Expectation Maximization process; following that, we infer multi-label values for instances in the training set. The inter-dependency structure is learned in the first iteration from the training-set labels, and in subsequent iterations, from the most recently inferred label values. The model parameters are estimated throughout the learning procedure using the training-set labels.

We use this iterative process as it modifies the network structure to reflect inter-dependencies among the most recently inferred label values. We expect such a network to allow the system to capture specific feature-label correlations and conditional independencies, which in turn, may improve the accuracy of the updated label assignments. As shown in the experiments section, this assumption is indeed supported by the improved performance of our system.

```

1 Initialize Bayesian network structure using the BANJO
  package [20], based on training-set labels;
2 Initialize model parameters,  $\alpha_i$  and  $\phi_{j,k}(v)$  using maximum
  likelihood estimation, and  $\theta_{j,k}^{\vec{l}}$  using EM algorithm, based on
  training-set labels;
3 Set initial inferred label values (i.e. each  $l_i^I$ ,  $i = 1, \dots, q$ ) for
  each instance  $I \in D$  to 0;
4 Set  $t$  to 0 and  $P$  to Hamming accuracy (or  $F_1$ -score) of initial
  model over training set;
5 while True do
6   Update Bayesian network structure using BANJO, based on
  most recently inferred label values;
7   Update model parameters,  $\alpha_i$ ,  $\phi_{j,k}(f_j^I)$ , and  $\theta_{j,k}^{\vec{l}}$ , based on
  training-set labels;
8   while True do
9     Infer values taken by random variables in each label
  dependency set  $LS_k$  (see Figure 5 for details);
10    if Hamming accuracy (or  $F_1$ -score) of model does not
  improve then
11      | break;
12    end
13  end
14  Set  $P'$  to Hamming accuracy (or  $F_1$ -score) of updated
  model over training set;
15  if  $P' \leq P$  then
16    | break;
17  end
18   $P \leftarrow P'$ ;  $t \leftarrow t + 1$ ;
19 end

```

Figure 4: Summary of model learning.

At the end of each iteration we assess the classification performance of our model over the training set; the iterative procedure is terminated when there is no improvement in performance between two successive iterations. For assessing model performance, we utilize the F_1 -score metric when using the dataset of multi-localized proteins, and the *Hamming accuracy* when using other multi-label datasets; these performance measures are described later in Section 4. The number of iterations needed to learn our model, which we denote by t , may vary across different datasets and also depends on the number of class-labels q ; in our experiments, the number of iterations did not exceed 10.

We use maximum likelihood estimation to compute the two sets of *observed* model parameters (shown in Equation 3): (a) The conditional probability of a label l_i^I given the values taken by L_i 's parents, $\alpha_i = \Pr(L_i = l_i^I | \mathcal{V}_{\text{Pa}(L_i)})$ and (b) The conditional probability of a feature value f_j^I given the values taken by all variables in each label dependency set (LDS), LS_k ($1 \leq k \leq q$), $\phi_{j,k}(f_j^I) = \Pr(F_j = f_j^I | \lambda^{F_j} = k, L_k = l_k^I, \mathcal{V}_{\text{Pa}(L_k)})$. To estimate the *latent* parameters, namely, the probability of each label dependency set, LS_k , $\theta_{j,k}^{\vec{l}}$, for a given label vector \vec{l} and a feature F_j , we developed an Expectation Maximization algorithm [7]:

1. **Expectation step.** For each instance I , we compute the probability of each LDS LS_k , to be selected for feature F_j , that is, $\lambda^{F_j} = k$, given I 's label vector \vec{l} and feature-value f_j^I , as:

$$\Pr(\lambda^{F_j} = k | F_j = f_j^I, \vec{l}^I) = \frac{\theta_{j,k}^{\vec{l}^I} \Pr(F_j = f_j^I | L_k = l_k^I, \mathcal{V}_{Pa(L_k)})}{\sum_{k=1}^q \theta_{j,k}^{\vec{l}^I} \Pr(F_j = f_j^I | L_k = l_k^I, \mathcal{V}_{Pa(L_k)})}$$

2. **Maximization step.** Using the probabilities computed in the Expectation step, we marginalize over all instances in the training set to re-estimate the mixture parameter, $\theta_{j,k}^{\vec{l}}$, for each feature F_j and label vector \vec{l} as:

$$\theta_{j,k}^{\vec{l}} = \frac{\sum_{v_j} \sum_{\{I | \vec{l}^I = \vec{l}, f_j^I = v_j\}} \Pr(\lambda^{F_j} = k | F_j = f_j^I, \vec{l}^I) \Pr(F_j = f_j^I | \vec{l}^I)}{\sum_{k=1}^q \sum_{v_j} \sum_{\{I | \vec{l}^I = \vec{l}, f_j^I = v_j\}} \Pr(\lambda^{F_j} = k | F_j = f_j^I, \vec{l}^I) \Pr(F_j = f_j^I | \vec{l}^I)}$$

where v_j takes on all possible values for feature F_j .

We denote by $\vec{l}_{LS_k}^I$ the *restriction* of the label vector \vec{l}^I to only those labels that are in the set LS_k . The conditional probability of a feature F_j to be assigned a value v given the values taken by the label variables in the label dependency set, LS_k , $\Pr(F_j = v | \mathcal{V}_{LS_k})$, is calculated as:

$$\Pr(F_j = v | L_k = l_k, \mathcal{V}_{Pa(L_k)}) = \Pr(F_j = v | \mathcal{V}_{LS_k}) = \frac{\sum_{\{I | \vec{l}_{LS_k}^I = \vec{l}_{LS_k}, f_j^I = v\}} \Pr(\lambda^{F_j} = k | F_j = f_j^I, \vec{l}^I) \Pr(F_j = f_j^I | \vec{l}^I)}{\sum_{v_j} \sum_{\{I | \vec{l}_{LS_k}^I = \vec{l}_{LS_k}, f_j^I = v_j\}} \Pr(\lambda^{F_j} = k | F_j = f_j^I, \vec{l}^I) \Pr(F_j = f_j^I | \vec{l}^I)}$$

Throughout the estimation process, we apply Laplace smoothing [18] by adding fractional pseudocounts to observed counts of events to all the parameters to avoid overfitting. The process of alternating between the Expectation step and the Maximization step is carried out until convergence is reached. To determine convergence, we test that changes to the latent parameter values between iterations are smaller than 0.05.

We next present the inference procedure for assigning multiple labels to instances.

3.2 Probabilistic Multi-label Classification

Probabilistic inference in the context of multi-label classification (MLC) amounts to assigning the most probable label vector \vec{l}^I to an instance I based on its feature vector \vec{f}^I . Inferring the conditional probability, $\Pr(\vec{l}^I | \vec{f}^I)$ for each label vector \vec{l} requires 2^q calculations, where q denotes the number of labels. To avoid this exponential number of calculations, some current probabilistic methods for multi-label classification assign a value to each label l_i ($1 \leq i \leq q$) such that the conditional probability $\Pr(L_i = l_i | \vec{f}^I)$ is maximized (see e.g. [1]). Others estimate the joint probability of the labels, $\Pr(L_1 = l_1, \dots, L_q = l_q | \vec{f}^I)$ and eventually infer each label value based on estimates of other labels (see e.g. [5]; [25]). These methods typically infer each label value by utilizing a fixed set of feature-label dependencies captured by their respective models.

In contrast, our system iteratively infers values for sets of multiple labels by capturing in each iteration specific feature-label dependencies based on the most recently inferred label values. We assign values to label variables in each label dependency set (LDS) LS_i (see Section 2.1 for the LDS definition), such that the conditional probability $\Pr(\mathcal{V}_{LS_i} | \vec{f}^I)$ is maximized.

To ensure that our method is practically applicable, we set a limit on the maximum number of parents, p , per label variable in the network. In the experiments described here, we restrict the dependency-set size to three (i.e. we set $p=2$) because the mean number of labels per dataset is at most three; the number of inference calculations is thus $2^{p+1}q = 2^3q = 8q$, where q ranges between 6 and 27. To gauge the influence of changes to the values of p on classifier performance, we ran experiments by varying the maximum number of parents in the range 1-3 using *Emotions* and *Scene* datasets, which have a relatively low number of labels. While increasing the value of p leads to a notable increase in the *Subset accuracy* measure of the classifier, there is no significant improvement in the classifier's *Hamming accuracy* measure (see Section 4 for details about these measures). We anticipate that higher values of p can further improve classifier performance when running experiments on datasets with higher numbers of labels.

As our system considers *multiple* dependency structures between features and labels, we expect that setting a relatively low bound on the dependency-set size considered in each structure, as we do here, will still allow the system to capture the significant dependencies and independencies among features and label subsets, even in larger datasets. Moreover, unconditional direct dependencies are not the only ones our model captures. While each label depends on two parent-labels—thus conditionally independent of other labels, indirect inter-dependencies are still captured throughout the network structure. As demonstrated by the results in Section 4, our utilization of label subsets of even a small size still significantly improves the performance of our system compared to that of current systems.

Given a feature vector \vec{f}^I of an instance I , our task is to predict its label vector \vec{l}^I , which involves assigning a 0/1 value to each of its labels l_i^I ($1 \leq i \leq q$). According to our probabilistic model, since the value of each label variable L_i depends only on values of its parent nodes $Pa(L_i)$ in a Bayesian network setting, for each L_i , we infer the values of variables in the label dependency set, $LS_i = \{L_i\} \cup Pa(L_i)$. To infer these label values, we follow an iterative process, which is summarized in the pseudocode shown in Figure 5. In each iteration, for all possible value assignments, l_i and $\mathcal{V}_{Pa(L_i)}$ to the label variable L_i and its parents, respectively, we

```

1 foreach label dependency set  $LS_i = L_i \cup Pa(L_i)$  do
2   foreach value assignment to  $L_i$  and to  $Pa(L_i)$  do
3     Calculate conditional probability:
4      $\Pr(L_i = l_i, \mathcal{V}_{Pa(L_i)} | \vec{f}^I, \mathcal{V}_{L_i}^I)^3$ ;
5   end
6   Select value assignment that maximizes the above
   conditional probability;
7   Update inferred values for labels in  $LS_i$  if classification
   performance over training set improves;
8 end

```

Figure 5: Summary of label inference.

Characteristic	Our System	EBN-M/EBN-J	ECC-NB	ECC-J48	BR-NB
Captures dependencies among labels	Using a probabilistic graphical model		Using classifier chains		No label inter-dependencies represented
Captures conditional independence between labels and features (given subsets of other labels)	Using label dependency sets and a mixture model	Do not capture such conditional independence			
Employs a generative model for data	Using Bayesian network		Using naïve Bayes	No generative model used	Using naïve Bayes

Table 1: Comparison of current multi-label classification systems characteristics.

calculate the conditional probability: $\Pr(L_i = l_i, \mathcal{V}_{Pa(L_i)} | \bar{f}^I, \mathcal{V}_{L_i}^I)$.³ The value assignment to L_i and to its parents, $Pa(L_i)$, that maximizes this probability is used as their current estimates. We note that label dependency sets do overlap, that is, the value of the same label variable L_i may be inferred multiple times, once for each dependency set in which it participates. As such, once the value of L_i is inferred within an iteration, it is only going to be updated during the same iteration if this improves the overall predictive performance of the model. While we currently use the standard inference techniques for Bayesian network models [18], there is much room for optimization by using methods for approximate inference that consider only the likely label combinations and fewer label sets, which we shall pursue in the future.

4 Experiments and Results

We present in this section two sets of experiments. In the first, we utilize the standard collection of multi-label datasets that were previously used to assess the performance of multi-label classification (MLC) systems. In the second, we employ a dataset used for a more concrete application in computational biology, namely predicting locations of proteins within the cell, also known as *protein multi-location prediction*. Details of these experiments are provided below.

4.1 Datasets and Performance Measures

For the first set of experiments, we use the same multi-label datasets that have been previously used in several comprehensive studies (e.g. [1, 8, 26]) to assess MLC system performance, namely: *Emotions* (72 features, 6 labels), *Scene* (294 features, 6 labels), *Yeast* (103 features, 14 labels), and *Genbase* (1186 features, 27 labels). We compare the performance of our system to that of state-of-the-art multi-label classification systems that were evaluated in a comprehensive study by Alessandro et al. [1]. The study focused primarily on MLC systems based on Classifier Chains, and included: ensemble of Bayesian networks, namely, *EBN-J/EBN-M* [1], ensemble of chain classifiers [15] using Naïve Bayes (denoted *ECC-NB*) and using J48 (denoted *ECC-J48*), and Binary Relevance using Naïve Bayes (denoted *BR-NB*) [22]. We note that the last of these four systems is not a classifier-chain but was still included in that study and is thus included here as well. As done in Alessandro’s study, we discretize each real-valued feature into four bins, select features using a correlation-based feature selection technique [24], and employ the

³ Recall that \bar{L}_i denotes the set of all label variables *other than* L_i and $Pa(L_i)$ and that the values taken by the variables in \bar{L}_i is denoted as $\mathcal{V}_{\bar{L}_i}$.

stratified 10-fold cross-validation for evaluating system performance. Table 1 summarizes the main distinguishing properties of the compared systems.

In the second set of experiments, we use a protein multi-location dataset, derived from DBMLoc [27], where each protein is represented by 30 features, and the 9 possible subcellular locations correspond to 9 class-labels (see [19] for detail). We compare the performance of our system to that of state-of-the-art multi-location prediction systems as reported by Briesemeister et al. [2], in their assessment of the YLoc⁺ system [2], including Euk-mPLoc [3], WoLF PSORT [11], and KnowPred_{site} [12]. According to the methods used in the previous assessment [2], we employ minimal entropy partitioning technique [9] for feature discretization, and stratified 5-fold cross-validation for training/testing the classifiers.

Under our current unoptimized implementation, wall clock time for model learning using training instances and inferring multi-labels of test instances combined is on the order of several minutes for datasets with a few labels, (lowest being ≤ 10 minutes for *Emotions*), and on the order of hours for datasets with more labels, (highest being ~ 20 hours for *Yeast*). We note that while the run-time of the prototypical system grows quadratically with the number of labels, it grows only linearly with the dataset size. For example, the run-time for the *protein multi-location* dataset (containing 8503 instances, with only 9 labels) is about 0.25 of the run-time for the smaller *Yeast* dataset (2417 instances) that has 13 labels.

Throughout the experiments, we use the valuation measures described below, which are the same as those applied in the corresponding previous work. For a given instance I , let $M^I = \{c_i \mid l_i^I = 1, \text{ where } 1 \leq i \leq q\}$ be the set of labels associated with I according to the dataset, and let $\hat{M}^I = \{c_i \mid \hat{l}_i^I = 1, \text{ where } 1 \leq i \leq q\}$ be the set of labels assigned to I by a classifier, where each \hat{l}_i^I is a 0/1 label assignment. The *Hamming* (H_{acc}) and the *Subset* (S_{acc}) accuracies used for the evaluation of multi-label prediction systems [1] are computed as:

$$H_{acc} = 1 - \frac{1}{|D|} \sum_{I \in D} \frac{1}{|C|} |M^I \Delta \hat{M}^I|, \text{ and}$$

$$S_{acc} = \frac{1}{|D|} \sum_{I \in D} \mathcal{I}(M^I = \hat{M}^I),$$

where Δ is the symmetric difference between M^I and \hat{M}^I . Additionally, the *Multi-label accuracy* (ML_{acc}) and F_1 -label score used

Measure	Dataset	Our system	EBN-M / EBN-J	ECC-J48	ECC-NB	BR-NB
H_{acc}	Emotions	.793 ($\pm .021$)	.780 ($\pm .022$)	.780 ($\pm .027$)	.781 ($\pm .026$)	.776 ($\pm .023$)
	Scene	.898 ($\pm .010$)	.880 ($\pm .010$)	.883 ($\pm .008$)	.835 ($\pm .007$)	.826 ($\pm .008$)
	Yeast	.786 ($\pm .007$)	.773 ($\pm .008$)	.771 ($\pm .007$)	.703 ($\pm .009$)	.703 ($\pm .011$)
	Genbase	.998 ($\pm .001$)	.998 ($\pm .001$)	.998 ($\pm .001$)	.996 ($\pm .001$)	.996 ($\pm .001$)
S_{acc}	Emotions	.319 ($\pm .036$)	.263 ($\pm .062$)	.260 ($\pm .038$)	.295 ($\pm .060$)	.261 ($\pm .049$)
	Scene	.610 ($\pm .030$)	.575 ($\pm .030$)	.531 ($\pm .038$)	.294 ($\pm .022$)	.276 ($\pm .017$)
	Yeast	.158 ($\pm .029$)	.127 ($\pm .018$)	.132 ($\pm .023$)	.102 ($\pm .023$)	.091 ($\pm .020$)
	Genbase	.956 ($\pm .022$)	.965 ($\pm .015$)	.934 ($\pm .015$)	.897 ($\pm .031$)	.897 ($\pm .0031$)

Table 2: Hamming and Subset accuracies, H_{acc} and S_{acc} , for multi-label prediction systems. All values except ours are taken directly from Tables 2, 3, 4, 6 in the paper by Alessandro et al. [1]. Highest values are shown in boldface. Standard deviations are shown in parenthesis.

Measure	Our system	YLoc ⁺	Euk-mPLoc	WoLF PSORT	KnowPred _{site}
F_1 -label	0.71 (± 0.02)	0.68	0.44	0.53	0.66
ML_{acc}	0.68 (± 0.01)	0.64	0.41	0.43	0.63

Table 3: F_1 -label and ML_{acc} scores shown for protein multi-location prediction systems. All values except ours are taken directly from Table 3 in the paper by Briesemeister et al. [2]. Standard deviations are not available there. Highest values are shown in boldface.

for evaluating multi-location prediction systems [2] are computed as:

$$ML_{acc} = \frac{1}{|D|} \sum_{I \in D} \frac{|M^I \cap \hat{M}^I|}{|M^I \cup \hat{M}^I|}, \text{ and}$$

$$F_1\text{-label} = \frac{1}{|C|} \sum_{c_i \in C} \frac{2 \times Pre_{c_i} \times Rec_{c_i}}{Pre_{c_i} + Rec_{c_i}},$$

where Pre_{c_i} and Rec_{c_i} for label c_i are adapted measures of multi-label precision and recall given by Briesemeister et al. [2]:

$$Pre_{c_i} = \frac{1}{|\{I \in D | c_i \in \hat{M}^I\}|} \sum_{I \in D | c_i \in \hat{M}^I} \frac{|M^I \cap \hat{M}^I|}{|\hat{M}^I|}, \text{ and}$$

$$Rec_{c_i} = \frac{1}{|\{I \in D | c_i \in M^I\}|} \sum_{I \in D | c_i \in M^I} \frac{|M^I \cap \hat{M}^I|}{|M^I|}.$$

4.2 Classification Results

Table 2 shows the *Hamming* and the *Subset* accuracies (H_{acc} and S_{acc} , respectively) of our system compared to that obtained by current MLC systems (as reported by Alessandro et al. [1], Tables 2, 3, 4, 6 there), obtained over the same multi-label datasets and evaluation measures. The results show that our system has higher H_{acc} and S_{acc} than all other systems over all datasets except *Genbase*. The differences in the improved performance values are statistically significant ($p \ll 0.05$, according to the 2-sample *t*-test [4]). Over the

Genbase dataset, our system has the same H_{acc} as the others and a slightly lower S_{acc} , although the latter difference is not statistically significant. The reason for the lack of improvement in this case can be attributed to the fact that in the *Genbase* dataset, the mean number of labels per instance is much lower than in the other datasets. As such, there are relatively few dependencies and independencies among labels and features to be utilized by our system.

Table 3 shows the F_1 -label score and *Multi-label* accuracy (ML_{acc}) of our system compared with those obtained by top multi-location prediction systems (as reported by Briesemeister et al. [2], Table 3 there), obtained over the same set of multi-localized proteins and evaluation measures. The table shows that our system improves over the performance of all other systems. The differences between scores obtained by our system and those of the closest top performing system, YLoc⁺, are highly statistically significant ($p \ll 0.001$).

Thus, the results clearly demonstrate that our system, which utilizes the intricate dependence and independence structure among features and labels, improves upon current multi-label classification methods, as shown over a variety of multi-label datasets previously used for systems-comparison.

5 Conclusions and Future Work

We presented a probabilistic generative model that captures inter-dependencies among labels as well as dependencies between features and labels. Unlike other approaches for multi-label classification (MLC), our model represents conditional independencies between feature values and labels given subsets of other labels, par-

ticularly by introducing the concept of *label dependency sets*. For example, in the *Emotions* dataset, the *tone* feature of songs depends on the class labels *Quiet-Still*, *Sad-Lonely*, *Amazed-Surprised*, and *Angry-Aggressive*. Typically, songs labeled as belonging to the first two classes have a *Low* tone while those in the last two classes have a *High* tone. Our system directly captures the conditional independence between the tone feature and the first two labels given the other two labels. Notably, current systems do not attempt to capture such subtle and informative dependencies and independencies. Our experiments over diverse datasets indeed show that directly modeling these dependence and independence relationships contributes to improved accuracy in multi-label classification, compared to previously studied systems based on Classifier Chains.

While we employ relatively small dependency sets in this study, the improved performance of our system strongly suggests that even such small sets can still help model the significant dependencies between feature and label subsets. We plan to develop approximate methods for inference by considering only the likely label combinations and fewer label sets to enable the practical use of larger label dependency sets.

Since utilizing label dependency sets has proven useful, our next aim is to directly learn label combinations that are most likely to strongly influence feature values. We will conduct experiments over larger datasets from different application domains, including more complex label combinations. We anticipate that employing such label subsets in the mixture model framework will be crucial to effectively integrate features from different sources, for example, from text and non-text data, and improve multi-label classification performance.

REFERENCES

- [1] A. Alessandro, G. Corani, D. Mauá, and S. Gabaglio, ‘An ensemble of Bayesian networks for multilabel classification’, in *International Joint Conference on Artificial Intelligence*, pp. 1220–1225, (2013).
- [2] S. Briesemeister, J. Rahnenfuhrer, and O. Kohlbacher, ‘Going from where to why – interpretable prediction of protein subcellular localization’, *Bioinformatics*, **26**(9), 1232–1238, (2010).
- [3] K. Chou and H. Shen, ‘Euk-mPLoc: A fusion classifier for large-scale eukaryotic protein subcellular location prediction by incorporating multiple sites’, *Journal of Proteome Research*, **6**(5), 1728–1734, (2007).
- [4] M. DeGroot and M. Schervish, *Probability and Statistics*, Pearson Education, New Jersey, USA, 4th edn., 2012.
- [5] K. Dembczynski, W. Cheng, and E. Hüllermeier, ‘Bayes optimal multilabel classification via probabilistic classifier chains’, in *International Conference on Machine Learning*, pp. 279–286, (June 2010).
- [6] K. Dembczynski, W. Waegeman, and E. Hüllermeier, ‘An analysis of chaining in multi-label classification.’, in *European Conference on Artificial Intelligence*, volume 242, pp. 294–299, (2012).
- [7] A. Dempster, N. Laird, and D. Rubin, ‘Maximum likelihood from incomplete data via the em algorithm’, *Journal of the Royal Statistical Society, Series B*, **39**(1), 1–38, (1977).
- [8] J. Doppa, J. Yu, C. Ma, A. Fern, and P. Tadepalli, ‘HC-search for multi-label prediction: An empirical study.’, in *AAAI Conference on Artificial Intelligence*, pp. 1795–1801, (2014).
- [9] U. Fayyad and K. Irani, ‘Multi-interval discretization of continuous-valued attributes for classification learning’, in *International Joint Conference on Artificial Intelligence*, pp. 1022–1029, (1993).
- [10] Y. Guo and S. Gu, ‘Multi-label classification using conditional dependency networks.’, in *International Joint Conference on Artificial Intelligence*, pp. 1300–1305, (2011).
- [11] P. Horton, K. Park, T. Obayashi, N. Fujita, H. Harada, C. Adams-Collier, and K. Nakai, ‘WoLF PSORT: Protein localization predictor’, *Nucleic Acids Research*, **35**(Web Server issue), W585–W587, (2007).
- [12] H. Lin, C. Chen, T. Sung, S. Ho, and W. Hsu, ‘Protein subcellular localization prediction of eukaryotes using a knowledge-based approach’, *BMC Bioinformatics*, **10**(Suppl 15), 8, (2009).
- [13] A. McCallum, ‘Multi-label text classification with a mixture model trained by em’, in *AAAI Workshop on Text Learning*, (1999).
- [14] J. Read, L. Martino, and D. Luengo, ‘Efficient monte carlo optimization for multi-label classifier chains.’, in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3457–3461, (2013).
- [15] J. Read, B. Pfahringer, G. Holmes, and E. Frank, ‘Classifier chains for multi-label classification’, *Machine Learning*, **85**(3), 333–359, (2011).
- [16] A. Romero and L. de Campos, ‘A probabilistic methodology for multilabel classification.’, *Intell. Data Anal.*, **18**(5), 911–926, (2014).
- [17] T. Rubin, A. Chambers, P. Smyth, and M. Steyvers, ‘Statistical topic models for multi-label document classification’, *Machine Learning*, **88**(1-2), 157–208, (2012).
- [18] S. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach*, Pearson Education, New Jersey, USA, 3rd edn., 2010.
- [19] R. Simha, S. Briesemeister, O. Kohlbacher, and H. Shatkay, ‘Protein (multi-) location prediction: utilizing interdependencies via a generative model’, *Bioinformatics*, **12**, i365–i374, (2015).
- [20] A. Smith, J. Yu, T. Smulders, A. Hartemink, and E. Jarvis, ‘Computational inference of neural information flow networks’, *PLoS Computational Biology*, **2**(11), e161, (2006).
- [21] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, ‘Multilabel classification of music into emotions’, in *International Conference on Music Information Retrieval*, pp. 325–330, (2008).
- [22] G. Tsoumakas, I. Katakis, and I. Vlahavas, ‘Mining multi-label data’, in *Data Mining and Knowledge Discovery Handbook*, pp. 667–685, (2010).
- [23] H. Wang, M. Huang, and X. Zhu, ‘A generative probabilistic model for multi-label classification’, in *International Conference on Data Mining*, pp. 628–637, (2008).
- [24] I. Witten, E. Frank, and M. Hall, *Data Mining: Practical Machine Learning Tools and Techniques.*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edn., 2011.
- [25] J. Zaragoza, L. Sucar, E. Morales, C. Bielza, and P. Larrañaga, ‘Bayesian chain classifiers for multidimensional classification’, in *International Joint Conference on Artificial Intelligence*, pp. 2192–2197, (2011).
- [26] M. Zhang and K. Zhang, ‘Multi-label learning by exploiting label dependency’, in *International Conference on Knowledge Discovery and Data Mining*, pp. 999–1008, (2010).
- [27] S. Zhang, X. Xia, J. Shen, Y. Zhou, and Z. Sun, ‘DBMLoc: A Database of proteins with multiple subcellular localizations’, *BMC Bioinformatics*, **9**, 127, (2008).

Accelerating Norm Emergence Through Hierarchical Heuristic Learning

Tianpei Yang¹ and Zhaopeng Meng^{1,2} and Jianye Hao³ and Sandip Sen⁴ and Chao Yu⁵

Abstract. Social norms serve as an important mechanism to regulate the behaviours of agents and to facilitate coordination among them in multiagent systems. One important research question is how a norm can rapidly emerge through repeated local interaction within agent societies under different environments when their coordination space becomes large. To address this problem, we propose a hierarchically heuristic learning strategy (HHLS) under the hierarchical social learning framework. Subordinate agents report their information to their supervisors, while supervisors can generate instructions (rules and suggestions) based on the information collected from their subordinates. Subordinate agents heuristically update their strategies based on both their own experience and the instructions from their supervisors. Extensive experiment evaluations show that HHLS can support the emergence of desirable social norms more efficiently and can be applicable in a much wider range of multiagent interaction scenarios compared with previous work. The influence of key related factors (e.g., different topologies, population, neighbourhood and action space size, cluster size) are also investigated and new insights are obtained as well.

1 INTRODUCTION

In multiagent systems, social norms play an important role in regulating agents' behaviors to ensure coordination among agents and functioning of agent societies. One commonly adopted characterization of a norm is to model it as a consistent equilibrium that all agents follow during interactions where multiple equivalent equilibria coexist [20]. How social norms can emerge efficiently in agent societies is a key research problem in the area of normative multiagent systems.

There exist two major approaches for addressing norm emergence problem: the top-down approach and the bottom-up approach. The former approach investigates how to efficiently synthesize a norm for all agents beforehand, while the latter one focuses on investigating how a norm can emerge through repeated local interactions by learning among agents. In distributed multiagent interaction environments, it is usually difficult to come up with any norm before agents interactions start since there may not exist such a centralized controller and also the optimal norm may vary frequently as the environment dynamically changes and therefore, the bottom-up approach via local learning promises to be more suitable for such kinds of distributed and dynamic environments.

Until now, significant efforts have been devoted to investigating norm emergence problem from the bottom-up research direction [2, 3, 5–9, 12–17, 21–25]. Sen and Airiau [13] investigated the norm emergence problem in a population of agents within randomly connected networks where each agent is equipped with certain existing multiagent learning algorithms. The local interaction among each pair of agents is modeled as two-player normal-form games, and a norm corresponds to one consistent Nash equilibrium of the coordination/anti-coordination game. Later a number of papers [2, 9, 12, 17] subsequently extended this work by using more realistic and complex networks (e.g., small-world network and scale-free network) to model the diverse interaction patterns among agents. Additionally, different learning strategies and mechanisms have been proposed to better facilitate norm emergence among agents within different interaction environments [3, 6, 8, 11, 25].

Most of the previous works only focus on games with relatively small size, which do not accurately reflect the practical interaction scenarios where the action space of agents can be quite large. With the increasing of the action space, unfortunately, most of the existing approaches usually result in very slow norm emergence or even fail to converge. Recently Yu et al. [21] proposed a hierarchical learning strategy to improve the norm emergence rate for the huge action space problem. However, this work only considers the case in which a norm corresponds to a Nash equilibrium where all agents select the same action. This usually can be modelled as a two-player n -action *coordination game*. One simple example with $n=2$ is shown in Table 1. In contrast, in realistic interaction scenarios, a norm may correspond to agents coordinating using different actions. One notable example is considering two drivers arriving at a road intersection from two neighbouring roads. To avoid collision, one possible norm is "yield to the left", i.e., waiting for the car on the left-hand side to go through the intersection first. This kind of scenario can naturally be modelled as an *anti-coordination game* shown in Table 2, which exist two different norms (a, b) and (b, a).

Furthermore, agents may be faced with the challenge of high mis-coordination cost and stochasticity of the environment. One representative example is shown in Table 3, which we call it *fully stochastic coordination game with high penalty*. In this game, there exist two optimal Nash equilibriums each of which corresponds to one norm, and one suboptimal Nash equilibrium. Two major challenges coexist in this game: agents are vulnerable to converge to the suboptimal Nash equilibrium due to the high penalty when agents mis-coordinate on the outcomes; agents need to effectively distinguish between the stochasticity of the environment and the explorations of other learners. It is not clear, a priori, how a population of agents can efficiently evolve towards a consistent norm given the large space of possible norms in such challenging environments.

¹ School of Computer Software, Tianjin University, China, email: {tpyang, mengzp}@tju.edu.cn

² Tianjin University of Traditional Chinese Medicine, China

³ School of Computer Software, Tianjin University, China, email: jianye.hao@tju.edu.cn; Corresponding author

⁴ University of Tulsa, USA, email: sandip-sen@utulsa.edu

⁵ Dalian University of Technology, China, email: cy496@dlut.edu.cn

Table 1. An example of coordination game

		Agent 2's actions	
		a	b
Agent 1's actions	a	1	-1
	b	-1	1

Table 2. An example of anti-coordination game

		Agent 2's actions	
		a	b
Agent 1's actions	a	-1	1
	b	1	-1

Table 3. Fully stochastic coordination game with high penalty

		Agent 2's actions		
		a	b	c
Agent 1's actions	a	8/12	-5/5	-20/-40
	b	-5/5	0/14	-5/5
	c	-20/-40	-5/5	8/12

To answer this question, in this paper we propose a novel hierarchical heuristic learning strategy (HHLS) under the hierarchical social learning framework to facilitate the rapid norm emergence in agent societies. In the hierarchical social learning framework, the agent society is separated into a number of clusters of subordinate agents, where each cluster's strategies are monitored and guided by one supervisor agent. For each supervisor agent, in each round, it collects the interaction information of the subordinate agents under its supervision and generates guided instructions in the forms of rules and suggestions for its subordinates. On the other hand, for each subordinate agent, apart from learning from its local interaction, it also adjusts its strategy based on the instructions from its supervisor. The main feature of the proposed framework is that through hierarchically supervised subordinate agents, an effective compromise solution can be generated to effectively balance distributed interactions and centralized control towards efficient and robust norm emergence. We evaluate the performance of HHLS under a wide range of games and experimental results show that HHLS can efficiently facilitate the rapid emergence of norms compared with the state-of-the-art approaches. We also investigate the influence of a number of key factors on norm emergence: the population size, the neighbourhood size, the size of action space, cluster size, different network topologies, etc.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces the hierarchical social learning framework and the heuristic learning strategy. Section 4 presents experimental evaluation results comparing with two representative state-of-the-art approaches. Finally Section 5 concludes the paper and points out future directions.

2 RELATED WORK

Norm emergence problem has received a wide range of attention in MASs literature. Shoham and Tennenholtz [14] firstly investigated the norm emergence problem in agent society based on a simple and natural strategy - the highest cumulative reward (HCR). In this study, they showed that HCR achieved high efficiency on social conventions in a class of games. Sen and Airiau [13] investigated the norm emergence problem in a population of agents within randomly connected networks where each agent is equipped with certain existing multiagent learning algorithms. They firstly proposed the model of learning *social learning*, where each agent learns from repeated interactions with multiple agents in a given scenario. In this study, the local interaction among each pair of agents is modeled as two-player normal-form games, and a norm corresponds to one consistent Nash equilibrium of the game. Later a number of papers [2, 9, 12, 17] subsequently extended this work by leveraging more realistic and complex networks (e.g., small-world network and scale-free network) to model the interaction patterns among agents and evaluated the influence of heterogeneous agent systems and space-constrained interactions on norm emergence. Savarimuthu [11] re-

capped the existing mechanisms on the multiagent-based emergence, and investigated the role of three proactive learning methods in accelerating norm emergence. The influence of the presence of liars on norm emergence is also considered and simulation results showed that norm emergence can still be sustained in the presence of liars. Villatoro et al. [17] proposed a reward learning mechanism based on interaction histories. In this study, they investigated the influence of different network topologies and the effects of memory of past activities on convention emergence. Later, they [15, 16] introduced two rules (i.e., re-wiring links with neighbors and observation) to overcome the suboptimal norm problems. They investigated the influence of Self-Reinforcing Substructure (SRS) in the network on impeding full convergence towards society-wide norms, which usually results in reduced convergence rates. Hao et al. [5] investigated the problem of coordinating towards optimal joint actions in cooperative games under the social learning framework by introducing two types of learners (IALs and JALs). Yu et al. [24] proposed a novel collective learning framework to investigate the influence of agent local collective behaviours on norm emergence in different scenarios and defined two strategies (collective learning-l and collective learning-g) to promote the emergence of norms where agents are allowed to make collective decisions within networked societies. Later Hao et al. [6] proposed two learning strategies under the collective learning framework: collective learning EV-l and collective learning EV-g to address the problem of high mis-coordination cost and stochasticity in complex and dynamic interaction scenarios. Recently Yu et al. [22] proposed an adaptive learning framework for efficient norm emergence. However, all the aforementioned works usually focus on relatively small-size games, and do not address the issue of efficient norm emergence in large action space problems.

Hierarchical learning framework, as a promising solution to accelerate coordination among agents, has been studied in different multi-agent applications (e.g., package routing [25], traffic control [1], p2p network [4] and smart-grid [19]). For example, Zhang et al. [25, 26] studied the package routing problem and proposed a multi-level organizational structure for automated supervision and a communication protocol for information exchange between higher-level supervising agents and subordinate agents. Simulation shows that the organization-based control framework can significantly increase the overall package routing efficiency than traditional non-hierarchical approaches. Abdoos et al. [1] proposed a multi-layer organizational controlling framework to model large traffic networks to improve the coordination between different car agents and the overall traffic efficiency. Until recently, Yu et al. [21] firstly proposed a hierarchical learning framework to study the norm emergence problem. In this study, they proposed a two-level hierarchical framework. Agents in the lower level interact with each other and report information to their supervisors in the higher level, while agents in the higher level called supervisors pass down guidance to the lower level. Agents in the lower level follow guidance in policy update. However, their

framework is designed for coordination game only where each agent only needs to coordinate on the same action for norm emergence.

3 HIERARCHICAL SOCIAL LEARNING FRAMEWORK

3.1 Framework Overview

We consider a population N of agents where each agent is connected following the underlying network topology. In each round, each agent interacts with one randomly selected agent from its neighbourhood. An agent's neighbourhood consists of all agents which it is physically connected with. We model the interaction between each pair of agents as a normal-form game. At the beginning of each interaction, one agent is randomly assigned as the row player and the other as the column player. We assume that each agent can only have access to its own action and payoff information during interaction. On the other hand, the population of agents are divided into multiple levels, and the agents in each level supervise the behaviours of agents from its neighbouring lower level. For the sake of exposition, we present the hierarchical social learning framework in two levels. However, it is straightforward to extend the hierarchical social learning framework into $k > 2$ levels.

One illustrating example of two-level hierarchical network is shown in Figure 1. Each supervisor agent i in the higher level is in charge of a group of subordinate agents (denoted as $sub(i)$) in the bottom level surrounded by dashed lines. For each subordinate agent j , its supervisor agent is denoted as $sup(j)$. For subordinates, the topological connections between them are determined by the original network topology; for supervisors, a pair of supervisors are neighbouring agents if the corresponding group of subordinates they supervise are connected. Note that a supervisor agent can be viewed as a special subordinate agent within the original network, which is also allowed to communicate with its neighbouring supervisor agents.

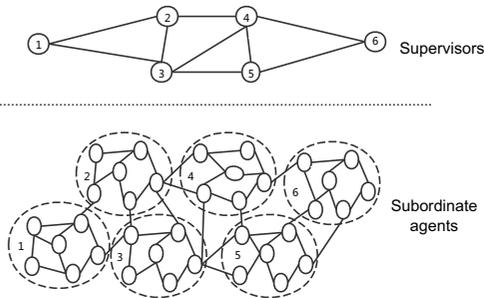


Figure 1. An example of the two-level hierarchical network

The interaction protocol of agents under the hierarchical social learning framework is summarized in Algorithm 1. In each round, each agent is paired with another agent randomly selected from its neighbourhood to interact with (Line 3), and their roles are randomly assigned (Line 4). Each agent then chooses an action following its learning strategy (Line 5), and then updates its strategy based on its current-round feedback (Line 6). After that, each subordinate agent reports its action and reward information to its supervisor (Line 7-9). At the end of each round, each supervisor collects all subordinate agents' information, generates and issues the instructions to its sub-

ordinate agents (Line 12-14). Finally each subordinate agent updates its strategy based on the instructions accordingly (Line 15-17).

Algorithm 1 The interaction protocol of hierarchical framework

```

1: for each round of interaction do
2:   for each agent  $i \in N$  do
3:     Randomly choose a neighbouring agent  $j$  to interact;
4:     Assign distinct roles randomly  $i \rightarrow$  state  $s_i, j \rightarrow$  state  $s_j$ 
5:     Select actions  $a_i$  and  $a_j$  and get rewards  $r_i$  and  $r_j$ ;
6:     Update its strategy based on  $\langle s_i, a_i, r_i \rangle$ .
7:     if agent  $i$  is a subordinate agent then
8:       Reporting its experience  $\langle s_i, a_i, r_i \rangle$  to  $sup(i)$ ;
9:     end if
10:  end for
11:  for each supervisor agent  $j$  do
12:    Generate instructions based on the information from  $sub(j)$ ;
13:    Provide the instructions to  $sub(j)$ ;
14:  end for
15:  for each subordinate agent  $k$  do
16:    Update its strategy based on the instructions from  $sup(k)$ ;
17:  end for
18: end for

```

3.2 Information Exchange between Supervisors and Subordinates

In the hierarchical social learning framework, subordinate agents send their feedback information to their corresponding supervisors, while supervisors pass down instructions to their corresponding subordinate agents. In details, each subordinate agent i reports its current-round interaction experience $\langle s_i, a_i, r_i \rangle$ to its supervisor $sup(i)$. For supervisors, we distinguish two different forms of instructions that they can provide to their subordinates: *suggestion* and *rule* [25]. Intuitively, a *rule* is a hard constraint that specifies an action that subordinate agents are forbidden to select under certain state next round; in contrast, a *suggestion* is a soft constraint which indirectly affects the strategies of the subordinate agents next round.

A set F of *rules* consists of all the forbidden actions for subordinates under different states. Formally we have,

$$F = \{ \langle s, a \rangle \mid a \in A, s \in S \} \quad (1)$$

where each element $\langle s, a \rangle$ denotes that action a is forbidden to take under state s ; A and S are the action space and state space of the subordinates.

A set D of *suggestions* specifies the recommendation degrees for different state-action pairs, which can be formally represented as follows,

$$D = \{ \langle s, a, d(s, a) \rangle \mid a \in A, s \in S \} \quad (2)$$

where $d(s, a)$ is the recommendation degree of action a under state s . Given an action and a state $\langle s, a \rangle$, if $d(s, a) < 0$, it indicates that action a is not recommended to select under state s ; if $d(s, a) > 0$, it indicates subordinate agents are encouraged to select action a when they are in state s . The way of determining rules and suggestions will be covered in details in Section 3.3.2.

3.3 Learning strategy

In this section, we first present the learning strategy of supervisors and how the rules and suggestions are generated in Section 3.3.1 and

$$freq(s, a) = \frac{|\{\langle s_k, a_k, r_k \rangle \mid \langle s_k, a_k, r_k \rangle \in RepInf, s_k = s, a_k = a, r_k = r_{max}(s, a)\}|}{|\{\langle s_k, a_k, r_k \rangle \mid \langle s_k, a_k, r_k \rangle \in RepInf, s_k = s, a_k = a\}|} \quad (3)$$

3.3.2 respectively. Following that, we describe the learning strategy of subordinate agents and how they utilize the instructions from supervisors in Section 3.3.3. Without loss of generality, let us assume that there is a set S of supervisors, and each supervisor $i \in S$, it supervises the set $sub(i)$ of subordinate agents. Each subordinate agent j has a set $neigh(j)$ of neighbours, and each supervisor agent i communicates with a set $com(i)$ of other supervisors.

3.3.1 Supervisor's strategy

We propose that each supervisor i holds a Q -value $Q_i(s, a)$ for each action a under each state s (row or column player). Let us denote the set of information from its subordinates as $RepInf_i = \{\langle s_k, a_k, r_k \rangle \mid k \in sub(i)\}$. For each piece of information $\langle s, a, r \rangle \in RepInf_i$, supervisor agent i updates its Q -value following the optimistic assumption shown in Equation (4),

$$Q_i(s, a) = (1 - \alpha_i) * Q_i(s, a) + \alpha_i * r \quad (4)$$

where α_i is its learning rate reflecting its updating degree between using the past experience and using the current round information.

After that, supervisor i further updates its Q -values based on optimistic assumption and the frequency information of each action similar to the FMQ heuristic [7]. Formally we have,

$$FMQ_i(s, a) = Q_i(s, a) + freq(s, a) * r_{max}(s, a) * C \quad (5)$$

where $r_{max}(s, a)$ is the max reward of each action a , $freq(s, a)$ is the frequency of receiving the reward of $r_{max}(s, a)$ by choosing action a under state s and C is a weighting factor.

The value of $r_{max}(s, a)$ is obtained from the reported information of its subordinate agents $sub(i)$. Specifically, The value of $r_{max}(s, a)$ is computed as the maximum reward that all of its subordinates receives under state s by choosing action a in the current round experience. Formally we have,

$$\mathcal{R}(s, a) = \{r_k \mid \langle s, a, r_k \rangle \in RefInf\} \quad (6)$$

$$r_{max}(s, a) = \max\{\mathcal{R}(s, a)\} \quad (7)$$

The frequency information $freq(s, a)$ is calculated as the empirical probability of receiving the maximum reward $r_{max}(s, a)$ under state s when action a is selected based on the reported information $RepInf$ collected from the subordinates which is shown in Equation (3).

After updating the strategy based on the information collected from its subordinates, we also allow each supervisor to learn from its neighboring peers (supervisors). Specifically each supervisor communicates with a neighboring supervisor randomly selected and imitates the neighbor's strategy. The motivation of imitating peers comes from evolutionary game theory [18], which provides a powerful methodology to model how strategies evolve over time based on their relative performance. One of the widely used imitation rules is the proportional imitation [10], which is adopted here as shown in Equation (8),

$$p = \frac{1}{1 + e^{-\beta * (FMQ_j(s, a) - FMQ_i(s, a))}} \quad (8)$$

where parameter β controls the degree of imitating the strategy (the FMQ-value) of the neighbouring supervisor.

Finally each supervisor i updates its strategy (denoted as E -value $E_i(s, a)$) for each action a under state s as the average between the FMQ-values of its own and its neighbour j weighted by parameter p . Formally we have,

$$E_i(s, a) = (1 - p) * FMQ_i(s, a) + p * FMQ_j(s, a) \quad (9)$$

3.3.2 Supervisor Instruction Generation

Next we introduce how a supervisor generates instructions for its subordinates at the end of each round. As previously mentioned, there are two forms of instructions from a supervisor: rules and suggestions. First each supervisor i normalizes the E -values, which serves as the basis for generating instructions for its subordinates. Formally we have,

$$E'_i(s, a) = \frac{E_i(s, a) - \overline{E_i(s, a)}}{\sigma} \quad (10)$$

where $\overline{E_i(s, a)}$ is the mean of E -values averaged over all state-action pairs shown in Equation (11),

$$\overline{E_i(s, a)} = \frac{\sum_{a \in A} E_i(s, a)}{|A|} \quad (11)$$

The parameter σ is the standard deviation of FMQ-value following Equation (12),

$$\sigma = \sqrt{\frac{1}{|A|} \sum_{a \in A} (E_i(s, a) - \overline{E_i(s, a)})^2} \quad (12)$$

Given a state-action pair $\langle s, a \rangle$, if the E' -value $E'(s, a)$ is smaller than a given threshold, it indicates that selecting action a is not a wise choice under state s , thus it is encoded as a rule. Formally we have,

$$F = \{\langle s, a \rangle \mid E'(s, a) < \delta\} \quad (13)$$

where δ is the threshold which is set to the value of -0.5 in this paper.

For each state-action pair (s, a) , its recommendation degree $d(s, a)$ is set to the value of $E'_i(s, a)$. Thus the set of suggestions from supervisor i can be represented as follows,

$$D = \{\langle s, a, E'_i(s, a) \rangle \mid a \in A, s \in S\} \quad (14)$$

Given a state-action pair (s, a) , if $E'_i(s, a) < 0$, it indicates that selecting action a is not recommended under state s ; if $E'_i(s, a) > 0$, it indicates subordinate agents are encouraged to select action a when they are in state s .

3.3.3 Learning Strategy of Subordinates

Similar to the strategies of supervisors, each subordinate agent j also keeps a record of a Q -value $Q_j(s, a)$ for each action $a \in A_j$ under each state s . The Q -value $Q_j(s, a)$ indicates the past performance of choosing action a under state s and serves as the basis for making decisions [3]. For each subordinate agent j , let us first denote its feedback information received by the end of round t as $FeedInf_j^t = \{\langle s_m, a_m, r_m \rangle \mid m \in [1, t]\}$. At the end of each round

t , subordinate agent j updates its Q -value based on its feedback $\langle s_t, a_t, r_t \rangle$ as follows,

$$Q_j(s_t, a_t) = (1 - \alpha_j) * Q_j(s_t, a_t) + \alpha_j * r_t \quad (15)$$

where α_j is the learning rate modelling its updating degree between using the previous experience and using the most recent information.

Additionally each subordinate agent also updates its Q -values by taking into consideration both the optimistic assumption and the frequency information [7]. Formally we have,

$$FMQ_j(s, a) = Q_j(s, a) + freq(s, a) * r_{max}(s, a) * C \quad (16)$$

where $r_{max}(s, a)$ is the max reward of each action a based on its own experience, $freq(s, a)$ is the frequency of getting the payoff of $r_{max}(s, a)$ until now for action a and C is a weighting factor defining the trade-off between updating using Q -values and maximum payoff information. $freq(s, a)$ is calculated the same as shown in Equation (3).

After receiving supervisor's suggestions, each subordinate further adjusts its estimation of the goodness of each state-action pair based on the FMQ-values as follows,

$$E_j(s, a) = FMQ_j(s, a) * (1 + d(s, a) * \rho) \quad (17)$$

where $d(s, a)$ is the suggestion degree on the state-action pair (s, a) , and ρ is a weighting factor controlling the influence of the recommendation degree on the E -values.

Besides, supervisors also influence the subordinate agents' exploration rates. Let us suppose a subordinate agent j selects action a under current state s . If the supervisor i 's recommendation degree $d(s, a) < 0$, which indicates subordinate agent j 's current choice is not recommended, and agent j should increase the exploration rate to have more chance to select the recommended actions next time. On the other hand, if the recommendation degree $d(s, a) > 0$, it indicates the subordinates' current choice is recommended. Thus subordinate agent j decreases its exploration rate to avoid selecting discouraging actions in the future. For both cases, the adjustment degree varies depending on the absolute value of the state-action pair's recommendation degree. Formally each subordinate agent updates its exploration rate as follows,

$$\epsilon_j = \epsilon_j * (1 - d(s, a) * \gamma) \quad (18)$$

where γ is a weighting factor controlling the influence degree of the supervisor's suggestion on the subordinates' exploration rates.

Finally, given the current state s , each subordinate agent j chooses its action from those actions whose corresponding state-action pair do not belong to the set F of rules based on the corresponding set of E -values according to the ϵ -greedy mechanism. Specifically each agent chooses its action with the highest E -value with probability $1 - \epsilon_j$ to exploit the action with best performance currently (randomly selection in case of a tie), and makes random choices with probability ϵ_j to explore new actions with potentially better performance.

4 EXPERIMENTAL SIMULATION

In this section, we start with evaluating the norm emergence performance of our approach HHLS under different types of games by comparing with the state-of-the-art strategies. Following that we explore the influence of some parameters on norm emergence. Unless otherwise mentioned, all simulation results are obtained under a population of 500 agents within a small-world network. The average connection degrees of small-world and scale-free network are set to 6. All results are averaged over 1000 runs. The parameter settings are shown in Table 4.

Table 4. The initial value of parameters.

Parameters	α	ϵ	β	γ	ρ	θ
Value	0.99	0.93	0.1	0.05	0.01	0.005

4.1 Performance evaluation

We compare our approach HHLS with two previous works: hierarchical learning in [21] and social learning in [2]. All these three learning approaches are within the same social learning environment, i.e., each agent is allowed to interact with only one of its neighbours each round. The work in [2] is the representative state-of-the-art approach tackling norm emergence problem under multiagent social learning framework without considering any hierarchical organization. The work in [21] is the most recent approach introducing hierarchical learning into multiagent social learning framework to improve norm emergence efficiency. Four representative 6-action games are considered shown from Table 5 to Table 8.

Table 5. The payoff matrix of coordination game.

		Agent 2's actions					
		a	b	c	d	e	f
Agent 1's actions	a	1	-1	-1	-1	-1	-1
	b	-1	1	-1	-1	-1	-1
	c	-1	-1	1	-1	-1	-1
	d	-1	-1	-1	1	-1	-1
	e	-1	-1	-1	-1	1	-1
	f	-1	-1	-1	-1	-1	1

4.1.1 Coordination game (CG)

We first consider agents playing a 6-action coordination game (Table 5) in which there exist six norms. Agents are preferred to choose the same action. Figure 2 shows the dynamics of the average payoffs of agents with the number of rounds averaged for the three learning approaches. We can observe that all learning methods enable agents to achieve an average payoff of 1. Our hierarchically heuristic learning strategy converges faster than the hierarchical learning method [21], and the social learning method [2] is the slowest. This is because HHLS enables supervisor to influence subordinate agents in a more efficient manner, thus accelerating norm emergence.

4.1.2 Anti-coordination game (ACG)

Similarly, we consider agents playing a 6-action anti-coordination game (Table 6) in which there also exist six equivalently optimal norms. However, different from coordination game, each norm requires agents to choose different actions. Figure 3 shows the dynamics of the average payoffs using three learning methods. We can observe that both social learning [2] and HHLS enable agents to achieve an average payoff of 1, while the hierarchical learning fails. Besides, our HHLS converge faster than the social learning approach [2], which justifies the efficiency of introducing a hierarchical learning

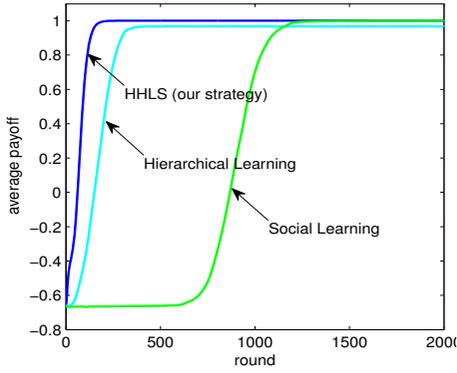


Figure 2. The dynamics of the average payoffs of agents in coordination games under different strategies

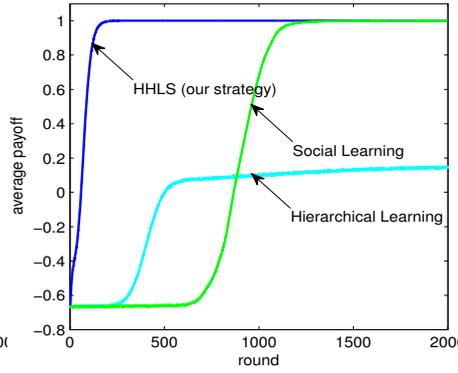


Figure 3. The dynamics of the average payoffs of agents in anti-coordination games under different strategies

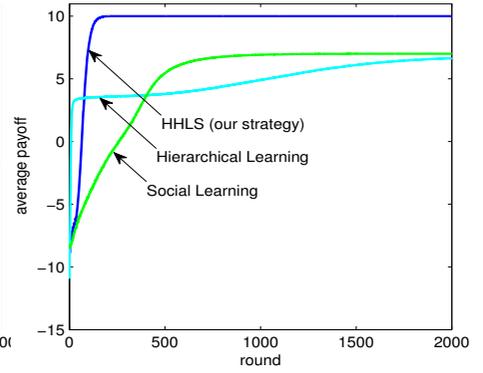


Figure 4. The dynamics of the average payoffs of agents in CGHP under different strategies

structure. For the hierarchical learning [21], it does not distinguish the state information and thus cannot adaptively select different actions for different states.

Table 6. The payoff matrix of anti-coordination game.

		Agent 2's actions					
		a	b	c	d	e	f
Agent 1's actions	a	-1	-1	-1	-1	-1	1
	b	-1	-1	-1	-1	1	-1
	c	-1	-1	-1	1	-1	-1
	d	-1	-1	1	-1	-1	-1
	e	-1	1	-1	-1	-1	-1
	f	1	-1	-1	-1	-1	-1

Table 7. The payoff matrix of coordination game with high penalty.

		Agent 2's actions					
		a	b	c	d	e	f
Agent 1's actions	a	10	0	-30	-30	0	-30
	b	0	7	0	0	0	0
	c	-30	0	10	-30	0	-30
	d	-30	0	-30	10	0	-30
	e	0	0	0	0	7	0
	f	-30	0	-30	-30	0	10

4.1.3 Coordination game with high penalty (CGHP)

Next, we consider 100 agents play a 6-action coordination game with high penalty (Table 7), in which there exist four optimal norms and two suboptimal norms. In this kind of games, agents are vulnerable to converge to suboptimal norms due to the existence of high mis-coordination cost (-30). Figure 4 shows the dynamics of the average payoffs of agents with the number of rounds for the three learning approaches. We can see that only HHLS enables agents to achieve an average payoff of 10 (i.e., converging to one optimal norm). The other two learning methods converge to one of the suboptimal norms, and they also converge slower than HHLS. We hypothesize the superior performance of HHLS is due to the integration of optimistic assumption during strategy update (to overcome mis-coordination cost effect) and efficient hierarchical supervision (to accelerate norm emergence speed).

4.1.4 Fully stochastic coordination game with high penalty (FSCGHP)

Last, we consider agents playing a 6-action fully stochastic coordination game with high penalty (Table 8). In FSCGHP, each outcome is associated with two possible payoffs and the agents receive one of

them with probability 0.5, which models the uncertainty of the interaction results. This game is in essence the same with the CGHP in which there also exist four optimal norms and two suboptimal norms. But it is more complex and difficult to emerge norms due to the stochasticity of the environments. Figure 5 shows the dynamics of the average payoffs of agents as the number of rounds for the three learning strategies. We can observe that in this challenging game, only HHLS enables agents to achieve an average payoff of 10 (one optimal norm is converged to). In contrast, the other two learning strategies converge to one of the suboptimal norms with a slower convergence rate. Finally it is worth to mention that if the size of norm space is further increased, the social learning method [2] and hierarchical learning method [21] cannot converge (to a suboptimal norm) within 10000 runs. However HHLS still can support converging to one optimal norm within approximately 200 rounds. The influence of action size will be discussed in Section 4.2.2 in details.

4.2 Influence of key parameters

In this section, we turn to investigate the influence of key parameters on the performance of norm emergence. We present the results for hierarchically heuristic learning under the small-world network and the CGHP game. The rest of parameters follow the same settings in Section 4 except the parameter being evaluated is changed.

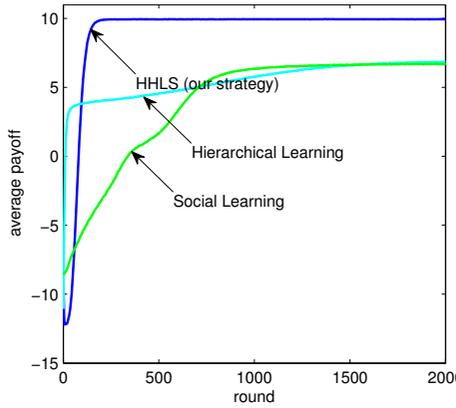


Figure 5. The dynamics of the average payoffs of agents in FSCGHP under different strategies

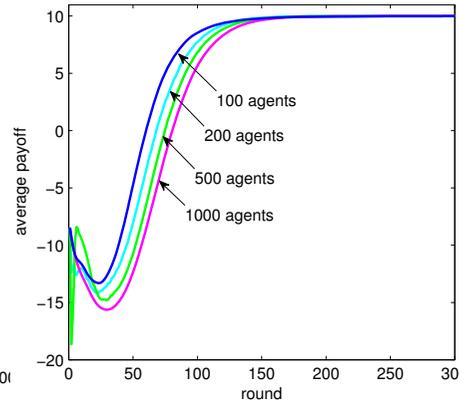


Figure 6. The influence of population size

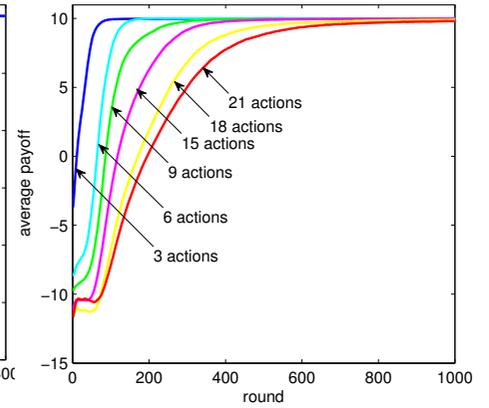


Figure 7. The influence of action size

Table 8. The payoff matrix of fully stochastic coordination game with high penalty.

		Agent 2's actions					
		a	b	c	d	e	f
Agent 1's actions	a	12/8	5/-5	-20/-40	-20/-40	5/-5	-20/-40
	b	5/-5	14/0	5/-5	5/-5	5/-5	5/-5
	c	-20/-40	5/-5	12/8	-20/-40	5/-5	-20/-40
	d	-20/-40	5/-5	-20/-40	12/8	5/-5	-20/-40
	e	5/-5	5/-5	5/-5	5/-5	14/0	5/-5
	f	-20/-40	5/-5	-20/-40	-20/-40	5/-5	12/8

4.2.1 Influence of population size

The influence of population size is shown in Figure 6. We can clearly observe the norm emergence efficiency is reduced as the increase of the population size. Given the cluster size unchanged, the number of clusters increase as the population size becomes larger. Thus it takes more time for each supervisor to coordinate between each other, and also more efforts are required for supervisors to guide all of their subordinate agents towards a consistent norm.

4.2.2 Influence of action size

Figure 7 shows the dynamics of the average payoffs of agents for different action sizes. We can see that the average convergence rate is decreased as the increase of the action space. This is reasonable because the coordination space becomes larger when the action size increases. Besides, larger action size usually results in more chances of mis-coordination cost and suboptimal norms, which additionally increases the coordination difficulty for agents toward a consistent norm. Finally it is worth noting that with the increase the action space, our framework can still efficiently support norm emergence without significantly degrading the performance (supporting norm

emergence within 1000 rounds for all cases). In contrast, in previous socially learning framework without utilizing a hierarchical organization [2], the norm convergence speed is decreased significantly when the action space is increased.

Table 9. The average number of rounds needed before convergence under different network topologies.

Convergence Speed	Game type				
	CG	ACG	CGHP	FSCGHP	
Network topology	Grid	142	141	131	153
	Ring	144	146	135	157
	Random	146	136	122	162
	Small-world	141	141	124	162
	Scale-free	149	144	129	163

4.2.3 Network topology

We evaluate the influence of five different networks: random network, grid network, ring network, small-world and scale-free network. Table 9 shows the average number of rounds needed before convergence. We find that hierarchical social learning framework is robust to different network topologies. HHLs enables agents to converge to norms in approximately the same number of runs under all the above five network topologies for different types of games.

Table 10. The influence of neighborhood size

Neighbour size	2	6	8	10	20	30	50	99
Convergence Rate	144	141	143	139	141	141	142	139

4.2.4 Influence of neighborhood size

We empirically evaluate the influence of neighborhood size varying it from 2 up to 99 (fully connected) with a population of 100 agents.

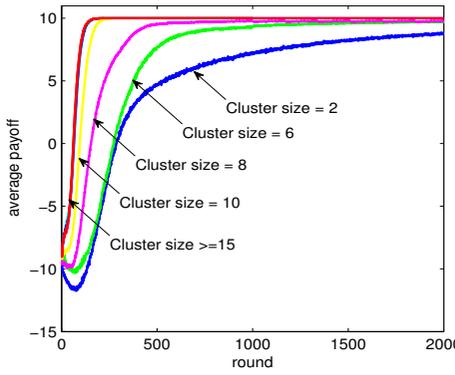


Figure 8. The influence of cluster size

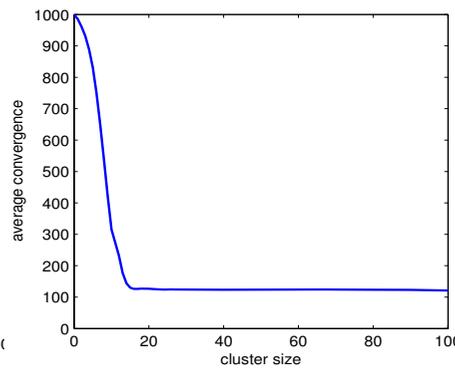


Figure 9. The average number of rounds needed before convergence under different cluster sizes

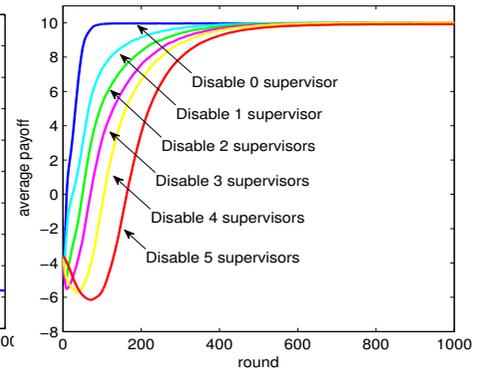


Figure 10. The influence of disabling supervisors

Table 10 shows the the average number of rounds needed before convergence for different neighborhood sizes. We can see that the average number of rounds required is stabilized around 140 rounds. This finding is different from the results usually observed in the traditional socially learning framework without a hierarchical structure [2]. This is because in hierarchical social learning framework, each supervisor supervises and guides a cluster of subordinate agents, which can overcome the low connectivity disadvantage when the neighbourhood size is small.

4.3 Influence of cluster size

One unique feature of the hierarchical social learning framework is the division of clusters of agents. Figure 8 and 9 show the influence of cluster size on norm emergence with a population of 100 agents. From Figure 8, we can see that the norm emergence rate is gradually increased as the increase of the cluster size, and stabilized when the cluster size is larger than 15. This phenomenon can be observed more clearly in Figure 9, which shows the average no. of rounds needed before convergence is reduced with the increase of cluster size and stabilized around 100 rounds.

When the cluster size is increased to 100 in the extreme case, it is essentially reduced to centralized control in which only one supervisor agent supervises all the rest of agents. In this case, all the communication and computation burden would fall on this single supervisor agent. When the cluster size is 1, it is essentially equivalent with the case of the traditional social learning without a hierarchical structure. When the cluster size varies between 1 and 100, with the increase of the cluster size, each supervisor agent can supervise more subordinate agents and thus it is easier for agents to coordinate among each other. However, as the cluster size exceeds certain threshold, the advantage of centralized supervision diminishes. This property is desirable since the same level performance as fully centralized supervision can be achieved under distributed supervision, which not only increases the robustness of the HHLS and the framework itself but reduces the communication and computation burden of supervisor agents.

Next we examine the robustness of HHLS in details by investigating the following questions: whether a consist norm can still rapidly emerge and how is the emergence efficiency changed when certain amount of supervisors are disabled? Figure 10 shows the dynamics of expected payoffs of agents with some supervisors disabled, and the results are averaged over 6-action coordination game with high

penalty. We can see that hierarchically heuristic learning still enables agents to converge to a consist norm when certain amount of supervisors are disabled. Though the convergence rate is gradually decreased as the increased of the number of disabled supervisors, better performance can still be achieved than the traditional social learning framework. This is expected since those subordinate agents without supervisors can only learn based on their local information, and the hierarchical social learning framework would be reduced into the traditional social learning framework when all supervisors are disabled.

5 CONCLUSION AND FUTURE WORK

We propose a hierarchically heuristic learning strategy to ensure efficient norm emergence in different distributed multiagent environments. Extensive simulation shows that our strategy can enable agents to reach consistent norms more efficiently and in a wider variety of games compared with previous approaches. The influence of different key parameters (e.g., population size, action space, neighbourhood size and network topology) is also investigated in details. We also evaluate the influence of centralized and decentralized hierarchically design by examining the effects of different cluster sizes (e.g., different number of supervisors). We find that sufficient degree of distributed supervision (large number of supervisors) can achieve the same performance as fully centralized supervision (only one supervisor), and thus making the HHLS robust towards the failure of certain supervisors.

In this paper, we divide subordinate agents randomly into several clusters. As future work, it is worthwhile investigating whether there exists an optimal way of clustering agents in terms of maximizing norm emergence rate and how an optimal clustering structure can be formed automatically among agents.

6 ACKNOWLEDGEMENTS

This work is partially supported by the subproject of the National Key Technology R&D Program of China (No.: 2015BAH52F01-1), National Natural Science Foundation of China (No.: 61304262) and Tianjin Research Program of Application Foundation and Advanced Technology (No.: 16JCQJNC00100).

REFERENCES

- [1] Monireh Abdoos, Nasser Mozayani, and Ana LC Bazzan, 'Holonic multi-agent system for traffic signals control', *Engineering Applications of Artificial Intelligence*, **26**(5), 1575–1587, (2013).
- [2] Stéphane Airiau, Sandip Sen, and Daniel Villatoro, 'Emergence of conventions through social learning', *Autonomous Agents and Multi-Agent Systems*, **28**(5), 779–804, (2014).
- [3] Reinaldo AC Bianchi, Carlos HC Ribeiro, and Anna Helena Reali Costa, 'Heuristic selection of actions in multiagent reinforcement learning', in *International Joint Conference on Artificial Intelligence*, pp. 690–695, (2007).
- [4] Jordi Campos, Marc Esteve, Maite López-Sánchez, Javier Morales, and Maria Salamó, 'Organisational adaptation of multi-agent systems in a peer-to-peer scenario', *Computing*, **91**(2), 169–215, (2011).
- [5] Jianye Hao and Ho-fung Leung, 'The dynamics of reinforcement social learning in cooperative multiagent systems.', in *Proceedings of 23rd International Joint Conference on Artificial Intelligence*, volume 13, pp. 184–190, (2013).
- [6] Jianye Hao, Jun Sun, Dongping Huang, Yi Cai, and Chao Yu, 'Heuristic collective learning for efficient and robust emergence of social norms', in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1647–1648, (2015).
- [7] Spiros Kapetanakis and Daniel Kudenko, 'Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems', in *Adaptive Agents and Multi-Agent Systems II*, 119–131, Springer, (2005).
- [8] Mihail Mihaylov, Karl Tuyls, and Ann Nowé, 'A decentralized approach for convention emergence in multi-agent systems', *Autonomous Agents and Multi-Agent Systems*, **28**(5), 749–778, (2014).
- [9] Partha Mukherjee, Sandip Sen, and Stéphane Airiau, 'Norm emergence under constrained interactions in diverse societies', in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pp. 779–786. International Foundation for Autonomous Agents and Multiagent Systems, (2008).
- [10] Jorge M Pacheco, Arne Traulsen, and Martin A Nowak, 'Coevolution of strategy and structure in complex networks with dynamical linking', *Physical review letters*, **97**(25), 258103, (2006).
- [11] Bastin Tony Roy Savarimuthu, Remy Arulanandam, and Maryam Purvis, 'Aspects of active norm learning and the effect of lying on norm emergence in agent societies', in *Agents in Principle, Agents in Practice*, 36–50, Springer, (2011).
- [12] Onkur Sen and Sandip Sen, 'Effects of social network topology and options on norm emergence', in *Coordination, Organizations, Institutions and Norms in Agent Systems V*, 211–222, Springer, (2010).
- [13] Sandip Sen and Stéphane Airiau, 'Emergence of norms through social learning', in *International Joint Conference on Artificial Intelligence*, volume 1507, p. 1512, (2007).
- [14] Yoav Shoham and Moshe Tennenholtz, 'On the emergence of social conventions: modeling, analysis, and simulations', *Artificial Intelligence*, **94**(1), 139–166, (1997).
- [15] Daniel Villatoro, Jordi Sabater-Mir, and Sandip Sen, 'Social instruments for robust convention emergence', in *International Joint Conference on Artificial Intelligence*, volume 11, pp. 420–425, (2011).
- [16] Daniel Villatoro, Jordi Sabater-Mir, and Sandip Sen, 'Robust convention emergence in social networks through self-reinforcing structures dissolution', *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, **8**(1), 2, (2013).
- [17] Daniel Villatoro, Sandip Sen, and Jordi Sabater-Mir, 'Topology and memory effect on convention emergence', in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*, pp. 233–240. IEEE Computer Society, (2009).
- [18] Jörgen W Weibull, *Evolutionary game theory*, MIT press, 1997.
- [19] Dayong Ye, Minjie Zhang, and Danny Sutanto, 'A hybrid multiagent framework with q-learning for power grid systems restoration', *Power Systems, IEEE Transactions on*, **26**(4), 2434–2441, (2011).
- [20] H Peyton Young, 'The economics of convention', *The Journal of Economic Perspectives*, **10**(2), 105–122, (1996).
- [21] Chao Yu, Hongtao Lv, Fenghui Ren, Honglin Bao, and Jianye Hao, 'Hierarchical learning for emergence of social norms in networked multiagent systems', in *AI 2015: Advances in Artificial Intelligence*, 630–643, Springer, (2015).
- [22] Chao Yu, Hongtao Lv, Sandip Sen, Jianye Hao, Fenghui Ren, and Rui Liu, 'An adaptive learning framework for efficient emergence of social norms', in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1307–1308. International Foundation for Autonomous Agents and Multiagent Systems, (2016).
- [23] Chao Yu, Guozhen Tan, Hongtao Lv, Zhen Wang, Jun Meng, Jianye Hao, and Fenghui Ren, 'Modelling adaptive learning behaviours for consensus formation in human societies', *Scientific reports*, **6**, (2016).
- [24] Chao Yu, Minjie Zhang, Fenghui Ren, and Xudong Luo, 'Emergence of social norms through collective learning in networked agent societies', in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 475–482, (2013).
- [25] Chongjie Zhang, Sherief Abdallah, and Victor Lesser, 'Integrating organizational control into multi-agent learning', in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 757–764, (2009).
- [26] Chongjie Zhang, Victor Lesser, and Sherief Abdallah, 'Self-organization for coordinating decentralized reinforcement learning', in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pp. 739–746, (2010).

Entity Embeddings with Conceptual Subspaces as a Basis for Plausible Reasoning

Shoab Jameel¹ and Steven Schockaert²

Abstract. Conceptual spaces are geometric representations of conceptual knowledge in which entities correspond to points, natural properties correspond to convex regions, and the dimensions of the space correspond to salient features. While conceptual spaces enable elegant models of various cognitive phenomena, the lack of automated methods for constructing such representations have so far limited their application in artificial intelligence. To address this issue, we propose a method which learns a vector-space embedding of entities from Wikipedia and constrains this embedding such that entities of the same semantic type are located in some lower-dimensional subspace. We experimentally demonstrate the usefulness of these subspaces as approximate conceptual space representations by showing, among others, that important features can be modelled as directions and that natural properties tend to correspond to convex regions.

1 INTRODUCTION

Despite the fact that several large-scale open-domain knowledge bases are now available (e.g. CYC, SUMO, Freebase, Wikidata and YAGO), few knowledge-driven applications rely on logical reasoning. An important reason for this is that available knowledge is often inconsistent. For example, the concept *ice cream shop* is asserted to be disjoint from *restaurant* in CYC, while it is considered a type of *restaurant* on Wikipedia³. Another challenge for logical reasoning is that available knowledge is seldom complete. For example, SUMO encodes⁴ knowledge about chess, darts and poker, but mentions nothing about checkers.

Humans are remarkably adept at overcoming such challenges [5, 11]. For example, we can recognize that the aforementioned conflict between CYC and Freebase is caused by the vagueness of the categories *restaurant* and *shop*, which both have *ice cream shop* as a borderline case. Similarly, we can deal with knowledge gaps by making inductive inferences, e.g. assuming that properties which hold for chess, darts and poker should hold for checkers as well. Automating such forms of plausible reasoning has proven challenging, among others because they rely on an underlying notion of similarity, which is difficult to characterize using purely symbolic methods.

The solution offered by the theory of conceptual spaces [13] is to represent concepts as regions in a suitable metric space. The points of this space correspond to (actual or possible) entities of a given semantic type, such that similar entities are located close to each other.

It is furthermore posited that most natural properties correspond to convex regions, in accordance with prototype theory [26]. Furthermore, the dimensions of a conceptual space correspond to the salient features of the considered domain. For example, a conceptual space of wines could have dimensions relating to sweetness, acidity, fruitiness, amount of tannins, etc. Using conceptual space representations, many cognitive phenomena, including vagueness and induction, can be modelled in a natural way [13, 8, 27, 19]. However, existing applications have focused on a few particular domains in which conceptual space representations can be derived from available metric information. For example, several authors have considered conceptual spaces for music perception [12, 4]. In such cases, the definition of the conceptual space, and its relationship to e.g. audio signals, relies on well-understood insights from the field of music cognition.

The research question we consider in this paper is whether we can automatically obtain approximate conceptual space representations for a wide range of domains, by combining information found in existing knowledge bases with representations derived from large text corpora such as Wikipedia.

Our approach builds on existing work for learning word embeddings from text corpora. Similar to conceptual spaces, word embeddings [21, 24, 29] represent the meaning of words in a high-dimensional Euclidean space, typically as vectors⁵. There are, however, two important differences between word embeddings and conceptual spaces. First, while word embeddings represent all words in a single vector space, conceptual spaces model the entities (and their properties) of a particular semantic type only (e.g. people and cities would be modelled in separate conceptual spaces). Because of this restriction, conceptual spaces can have dimensions that reflect the salient properties of the underlying domain. This allows us to use conceptual spaces for ranking entities (e.g. a conceptual space of cities should have a dimension corresponding to the population, allowing us to rank cities from the least to the most populous), modelling context effects⁶, and for describing how two entities or concepts are semantically related (e.g. that the rules of chess are more complex than the rules of checkers). In contrast, the dimensions of a word embedding space are essentially meaningless. Second, conceptual spaces clearly differentiate entities, which are modelled as points, from properties, which are modelled as regions. As a result, conceptual spaces can be used to model that a given entity has a given property or belongs to a given category, to model typicality (e.g. an *ice cream shop* could be located in the region modelling *shop* but to-

¹ Cardiff University, UK, email: JameelS1@cardiff.ac.uk

² Cardiff University, UK, email: SchockaertS1@cardiff.ac.uk

³ https://en.wikipedia.org/wiki/Category:Types_of_restaurants

⁴ <https://github.com/ontologyportal/sumo/blob/master/Sports.kif>

⁵ Two notable exceptions are [9] and [30], where words are represented using densities

⁶ The context-dependent nature of similarity is modelled in conceptual spaces by allowing dimensions to be rescaled, depending on the importance of the corresponding property in the given context.

wards the border), and to model semantic relations between different properties and categories.

In [6] an approach was proposed for learning conceptual space representations, which consists of (i) representing each entity of a given semantic type as a bag of words (e.g. each movie is represented as its set of user reviews), (ii) converting that bag of words representation to a vector space representation using multi-dimensional scaling (MDS), and (iii) identifying directions corresponding to salient properties of the considered domain in a post-hoc analysis.

An important limitation of the approach from [6] is that it cannot take advantage of relationships between different conceptual spaces. To address this, in this paper we propose a method that learns a single domain-independent vector space, in which each semantic type corresponds to a particular subspace. In other words, we learn conceptual space representations which are themselves embedded in a higher-dimensional vector space. Among others, this allows us to model semantic type hierarchies (e.g. the conceptual space of humans, in our model, is a subspace of the conceptual space of living things). Furthermore, different conceptual spaces can be aligned by taking into account semantic relations between entities of the corresponding types (e.g. the subspaces representing actors, directors and genres can help to obtain a more accurate representation of movies). Another important limitation of the approach from [6] is that MDS requires a distance matrix whose size is quadratic in the number of entities, which severely limits its scalability. In contrast, our model can easily learn representations for millions of entities.

2 RELATED WORK

2.1 Word embedding

Word embeddings are vector space representations which are used to model the meaning of words. Several existing models construct a vector for each word by applying some form of matrix factorization to a term-term co-occurrence matrix; see [29] for an overview of such approaches. Recently, a number of models have been proposed which instead explicitly optimize the predictive power of the word vectors. For example, the popular Skip-gram model [21] tries to find word vectors that can be used to predict the probability of seeing a context word, given an occurrence of the word being modelled, while the related continuous bag-of-words (CBOW) model focuses on the probability of seeing the word being modelled, given the occurrence of a context word.

An interesting property of word embeddings is that they often capture several kinds of semantic relations, beyond simple similarity. For example, in [21] it is shown that analogical proportions of the form a is to b what c is to d correspond to approximate parallelograms in the space obtained by Skip-gram. They also found that vector addition sometimes corresponds to a form of semantic composition, e.g. adding the vectors for *Germany* and *capital* resulted in a vector which is close to the vector for *Berlin*.

The fact that the vector space obtained by the Skip-gram model satisfies such linear regularities is at first glance somewhat surprising. In [24], the authors analyze what characteristics of a word embedding model can explain this effect, and propose a new model, called GloVe, which is explicitly aimed at capturing linear regularities. Since our model will build on GloVe, we briefly review its formulation. The GloVe model relies on a term-term co-occurrence matrix $X = (x_{ij})$, where x_{ij} is the number of times that word i appears in the context of word j . For each term t_i in the vocabulary, two word vectors w_i and \tilde{w}_i and a bias b_i are chosen by minimizing the follow-

ing objective:

$$J = \sum_{i=1}^V \sum_{j=1}^V f(x_{ij})(w_i \cdot \tilde{w}_j + b_i + b_j - \log x_{ij})^2 \quad (1)$$

where V is the number of words in the vocabulary. The function f is used to limit the impact of rare terms, whose co-occurrence counts are considered to be noisy. It is defined as follows:

$$f(x_{ij}) = \begin{cases} \left(\frac{x_{ij}}{x_{\max}}\right)^\alpha & \text{if } x_{ij} < x_{\max} \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

where x_{\max} is a constant which was fixed as 100. Intuitively, w_i reflects the meaning of term t_i while \tilde{w}_j reflects how the occurrence of that term in the context of another term t_j impacts the meaning of t_j .

2.2 Knowledge graph embedding

Knowledge bases such as Freebase and Wikidata can essentially be seen as collections of (subject, predicate, object) triples, and can thus be encoded as a graph, where nodes correspond to entities and edges are labelled with relation types. Several authors have looked at the problem of automatically expanding such knowledge graphs [7]. Here, we focus on models that rely on embedding knowledge graphs in a vector space, as we will use similar ideas for aligning different conceptual subspaces. The idea of embedding knowledge graphs in a vector space was proposed in [3]. In particular, they propose the model SE, in which each entity e_i is represented as a vector and each relation r_k is represented using two matrices R_k^{lhs} and R_k^{rhs} . The constraint they impose is that the following distance should be small for triples (e_i, r_k, e_j) in the knowledge graph and large for other triples:

$$d(R_k^{lhs} e_i, R_k^{rhs} e_j)$$

where d is either the Euclidean or Manhattan distance. An important drawback of this model is that it requires learning a large number of parameters, which was empirically found to lead to underfitting [2]. In [2] a simpler alternative, called TransE, was proposed, which represents each relation as a vector and considers the following scoring function instead:

$$d(e_i + r_k, e_j)$$

Despite the simplicity of this model, it was shown to substantially outperform SE in practice. However, as noted in [32], TransE is mostly suitable for one-to-one relations. To obtain a more faithful modelling of one-to-many, many-to-one and many-to-many relations, the model TransH is proposed. In this model, both a hyperplane H_k and an $(n-1)$ dimensional vector r_k is associated with each relation type (with n the dimension of the embedding space), and the following scoring function is considered:

$$d(e_i^{H_k} + r_k, e_j^{H_k})$$

where $e_i^{H_k}$ and $e_j^{H_k}$ are the orthogonal projections of e_i and e_j on the hyperplane H_k . The TransR model, introduced in [20], follows a similar strategy, but instead associates an m -dimensional vector r_k and an $m \times n$ matrix M_k with each relation, and uses the following scoring function:

$$d(e_i M_r + r_k, e_j M_r)$$

The underlying idea is to use the TransE model, after projecting the entities onto a relation-specific space. While in general it is not required that $n = m$, this particular choice was used in all experiments. Finally, [20] also proposes a variant CTransR, in which each entities are clustered, and each relation can have a different representation for each cluster.

In our model, the semantic types of entities play a crucial role. One other approach that explicitly takes semantic type into account is [14]. In particular, they add a regularization term to the objective function of existing embedding models to encode the requirement that entities of the same semantic type should be represented using similar vectors, which they formalize based on two manifold learning algorithms. Unfortunately, the scalability of the resulting method is relatively limited.

In [31] a model is proposed that combines word embedding with knowledge graph embedding. In particular, they jointly learn a representation for words, entities and relations, where the word representations are constrained similarly as in the Skip-gram model and the entities and relations are constrained similarly as in the TransE model. The entity and word representations are aligned either based on Wikipedia anchors or based on the entity names. An improvement of this method was proposed in [35], where the alignment is instead based on the text of the Wikipedia article of the entity. Along similar lines, [34] proposes a model in which the objective functions of Skip-gram and TransE are combined. A third component in their objective function allows the model to take into account an external similarity relation, by imposing the requirement that similar terms should have similar vectors. It is shown that the resulting model improves the word embeddings from Skip-gram.

The model we propose in this paper also combines a word based entity embedding component with a knowledge graph embedding component, although our motivation is different. In particular, [31] and [35] add a word based entity embedding component to a knowledge graph embedding model to improve the predictive performance for entities about which little or nothing is included in the knowledge graph. For example, if the knowledge graph contains the fact that entity a is in relation R with entity b , and from the word based component we can derive that entity a' is similar to entity a , then we can plausibly derive that entity a' might also be in relation R with entity b . Intuitively, we can thus view these approaches as using word based entity embedding to add a kind of smoothing to the knowledge graph embedding model. In contrast, our aim is to model how different entities of the same type are related. The kind of semantic relatedness in which we are interested (e.g. modelling that one building is *taller than* another one) is typically not captured by existing knowledge graphs. Intuitively, we use the word based entity embedding component to learn domain-specific vector space representations, and then use a knowledge graph embedding model to align these spaces. This allows us to improve the representation for semantic types about which little information is available in the considered textual descriptions. In this sense, we can view our model as using the knowledge graph embedding component to add a kind of smoothing to the word based entity embedding. Our motivation is somewhat similar in spirit to [34], but that model focuses on word embeddings rather than entity embeddings.

To the best of our knowledge, our model is the first to use semantic type information to learn domain-specific subspaces.

3 DESCRIPTION OF THE MODEL

Our aim is to learn a vector-space embedding of a set of entities E , in which entities of the same semantic type lie in some lower-dimensional subspace. Let S be the set of all semantic types. For $s \in S$, we write E_s for the set of all entities of type s . We furthermore assume that a set of binary relations R is available, and a set $G \subseteq E \times R \times E$ of triples of the form (e, k, f) , encoding that entities e and f are in relation k . Finally, we assume that for every entity e , a bag of words W_e describing that entity is available. The model we propose has the following form:

$$J = \alpha J_{text} + (1 - \alpha)(J_{type} + J_{rel}) + \beta J_{reg} \quad (3)$$

where $\alpha \in [0, 1]$ and $\beta \in [0, +\infty[$ are parameters controlling the relative importance of the different components of the model. Component J_{text} will be used to constrain the representation of the entities based on their textual description, J_{type} will impose the constraint that entities of the same type belong to a particular subspace, J_{rel} will use the relations in R to improve the alignment between these subspaces, and J_{reg} is a regularization component which will allow the model to automatically select the most appropriate number of dimensions for every subspace. We now discuss each of these components in more detail.

3.1 Word based entity embedding

From the bag of words representations W_e , we want to find a point $p_e \in \mathbb{R}^n$ for each entity e such that similar entities correspond to nearby points and such that salient features can be interpreted as directions in the space. Specifically, let f be a feature of interest, and let $x_i \in \mathbb{R}$ be the value of feature f for entity e_i , i.e. x_i reflects how much e_i has feature f . Then there should be a vector $w_f \in \mathbb{R}^n$ such that the orthogonal projection p'_{e_i} of the point p_{e_i} on the line $L_f = \{q \mid q = \lambda \cdot w_f, \lambda \in \mathbb{R}\}$ is given by $p'_{e_i} = c_f x_i w_f + b_f$ for b_f and c_f constants in \mathbb{R} . In other words, in a coordinate system where L_f coincides with one of the axes, the corresponding coordinate of p_e should be proportional to x_i . This requirement is equivalent to⁷:

$$p_{e_i} \cdot w_f = (c_f x_i + b_f) \cdot \|w_f\| \quad (4)$$

Unfortunately, we do not actually know what are the salient features in most domains. Following [6], we therefore use word co-occurrence as a proxy for feature values. In particular, we assume that each word potentially corresponds to a salient feature, and that the number of times a word co-occurs with a given entity reflects how much that entity has the corresponding feature. This leads to the following constraint

$$p_{e_i} \cdot w_j = g(y_{ji}) + b_j \quad (5)$$

where y_{ji} is the number of times word t_j occurs in W_{e_i} , g is a monotonic function that maps co-occurrence statistics to feature values, and b_j is a constant. Typically it will not be possible to satisfy the constraint (5) for all entities and all context words. The assumption underlying this model is that the salient features of an entity affect the co-occurrence statistics of many context words, and that the words for which (5) is (approximately) satisfied, in an optimal solution, will therefore be those that are strongly related to important features of the entity e_i .

Note that the requirement in (5) closely resembles the constraints that are optimized by the GloVe model. Moreover, as in the GloVe

⁷ We are abusing notation here, using p_{e_i} as a notation for the vector $\overrightarrow{Op_{e_i}}$.

model, we can choose $g(y_{ji}) = \log(y_{ji}) - b_i$ and formalize the objective function as a least squares regression problem, weighted such that frequent terms have a stronger impact on the objective function:

$$J_{\text{ext}}^E = \sum_{e_i \in E} \sum_{t_j \in W_{e_i}} f(y_{ji})(p_{e_i} \cdot w_j + b_i + b_j - \log y_{ji})^2$$

where f is defined as in (2). The resulting model is essentially the same as GloVe, but instead of modelling word-word co-occurrence we now model entity-word co-occurrence. The geometric interpretation, however, is different, as we view entities as points and context words as vectors. We can further constrain the word vectors w_j by adding a second component, capturing word-word co-occurrences, which corresponds to the original GloVe model. In particular, we define $J_{\text{ext}} = J_{\text{ext}}^E + J_{\text{glove}}$, where J_{glove} is the objective function J defined in (1).

3.2 Subspace constraints

A key distinguishing feature of our model is that all entities of a given type s are imposed to belong to the same subspace. To formalize this constraint, we associate with each semantic type s a set of $n + 1$ points $p_0^s, \dots, p_n^s \in \mathbb{R}^n$ and express that for each entity e_i of type s , the point p_{e_i} can be written as a convex combination of the points p_0^s, \dots, p_n^s :

$$J_{\text{type}} = \sum_{s \in S} \sum_{e \in E_s} \|p_e - \sum_{j=0}^n \lambda_j^{e,s} p_j^s\|^2$$

where we impose that $\lambda_j^{e,s} \geq 0$ and $\sum_{j=0}^n \lambda_j^{e,s} = 1$. Note that on its own, this component is trivial, as it suffices to choose any set of points p_0^s, \dots, p_n^s in general linear position. However, we will additionally require that the space spanned by the points p_0^s, \dots, p_n^s is as low-dimensional as possible. In particular, let M_s be the $n \times n$ matrix whose i^{th} row vector is $p_i^s - p_0^s$. Then clearly the rank of M_s is equal to the dimension of the space spanned by p_0^s, \dots, p_n^s . We now want to add a regularization term to penalize high-rank matrices M_s . Unfortunately, no efficient methods exist for directly minimizing the rank of a matrix M . The relaxation suggested in [10] is to minimize the nuclear norm $\|M\|_*$ instead (i.e. the sum of the singular values of M). This technique was empirically shown to lead to low-rank matrix solutions in many applications, and is known to be equivalent to rank minimization in certain cases [25]. The regularization term associated with J_{type} is thus given by

$$J_{\text{reg}}^1 = \sum_{s \in S} \|M_s\|_*$$

To implement nuclear norm regularization, we have used the recently proposed method from [16].

We will also consider a variant in which the points p_0^s, \dots, p_n^s are additionally required to be close to each other:

$$J_{\text{type}}^{\text{comb}} = \sum_{s \in S} \left(\sum_{e \in E_s} \|p_e - \sum_{j=0}^n \lambda_j^{e,s} p_j^s\|^2 + \sum_{j=0}^n d(p_j^s, c_j^s) \right)$$

where $c_j^s = \frac{1}{n+1} \sum_j p_j^s$ is the center-of-gravity of the points p_0^s, \dots, p_n^s .

3.3 Modelling relations

Often we have information about how entities of different types are related, e.g. the fact that Steven Spielberg is the director of Jurassic

Park. Such relationships can help us to align the subspaces corresponding to different types. Since our main aim is to improve the entity embeddings, rather than predicting relationships between entities of different types, methods such as TransH and TransR, which rely on projecting the entities to a different space, are not directly suitable. On the other hand, TransE is only suitable for one-to-one relations.

We propose an alternative to TransE which is inspired by our modeling of semantic types. As in TransE, we assume that every relation k is represented as a vector r_k . We furthermore write $\text{rhs}(e, k) = \{f \mid (e, k, f) \in G\}$ and $\text{lhs}(k, f) = \{e \mid (e, k, f) \in G\}$. Rather than imposing that $e + r_k = f$ if $(e, k, f) \in G$, as in TransE, we require that the points in $P_{e,k} = \{p_f \mid f \in \text{rhs}(e, k)\} \cup \{p_e + r_k\}$ lie in a low-dimensional subspace and, similarly, that the points in $P_{k,f} = \{p_e \mid e \in \text{lhs}(k, f)\} \cup \{p_f - r_k\}$ lie in a low-dimensional subspace. Note that in the case of one-to-many or many-to-one relations, this part of the model is similar to TransH in the special case where the considered subspaces are one-dimensional. Note that in the case of a one-to-many or many-to-many relation, the set of entities $\text{rhs}(e, k)$ is essentially treated as an additional semantic type (e.g. the set of all films directed by Stephen Spielberg), and similar for many-to-one relations and the set $\text{lhs}(k, f)$. As for the semantic types we will consider a number of variants:

$$\begin{aligned} J_{\text{rel}}^{\text{dim}} &= \sum_{k \in R} \sum_{p \in P_{e,k}} \|p - \sum_{j=0}^n \mu_j^{e,k} q_j^{e,k}\|^2 \\ &\quad + \sum_{p \in P_{k,f}} \|p - \sum_{j=0}^n \mu_j^{k,f} q_j^{k,f}\|^2 \\ J_{\text{rel}}^{\text{dist}} &= \sum_{f \in \text{rhs}(e,k)} d(p_f, p_e + r_k)^2 + \sum_{e \in \text{lhs}(k,f)} d(p_e, p_f - r_k)^2 \\ J_{\text{rel}} &= J_{\text{rel}} + J_{\text{rel}}^{\text{dist}} \end{aligned}$$

where we write e.g. $p \in P_{e,k}$ to sum over all entities e and all points p in $P_{e,k}$. Note that the variant $J_{\text{rel}}^{\text{dist}}$ essentially corresponds to TransE. For the variants J_{rel} and $J_{\text{rel}}^{\text{comb}}$ we again use nuclear norm regularization to enforce low-dimensional subspaces. Let the i^{th} row vector of the matrix $M_{e,k}$ be given by $q_i^{e,k} - q_0^{e,k}$ and similar for $M_{k,f}$. We define:

$$J_{\text{reg}}^2 = \sum_{k \in R} \|M_{e,k}\|_* + \|M_{k,f}\|_*$$

Note that we only need to consider the combination (e, k) or the combination (k, f) if there is at least one triple of the form (e, k, f) in G , since otherwise we can trivially choose $M_{e,k}$ and $M_{k,f}$ as the zero matrix. The full regularization term is given by $J_{\text{reg}} = J_{\text{reg}}^1 + J_{\text{reg}}^2$.

4 EVALUATION

4.1 Data acquisition

In our experiments, we have used Wikidata to obtain a set of entities E and their corresponding semantic types. To generate the bag-of-words representation W_e of a given entity, we take advantage of the fact that Wikidata entities e are linked to their corresponding Wikipedia article d_e . The set W_e contains the words occurring in d_e , as well as the m words before and after any mentions of the entity in other Wikipedia articles. Following [24], we have used a window size of $m = 10$ (but without crossing sentence boundaries). In particular, we treat every link from some Wikipedia article d_x to d_e as a mention of e , as well as any repeated occurrences of the corresponding anchor

text in d_x . The word-word co-occurrence in the J_{glove} component of our model has been obtained from the entire Wikipedia corpus, as in the standard GloVe model. Using the Wikidata dump from October 26, 2015 and the Wikipedia dump from November 02, 2015, we have then selected those entities e which are mentioned in at least 10 Wikipedia articles, resulting in a set E containing 1,292,702 entities. For each semantic type s , the set E_s contains those entities which are asserted to be of type s via the *instance of* property as well as all instances which are asserted to belong to one of the super-types of s , which was determined using the *subclass of* property. As the set of binary relations R we considered all Wikidata properties whose value is another entity, apart from *instance of* and *subclass of* which have already been used to determine the sets E_s . In the case of Wikipedia, we adopted a fairly straightforward preprocessing strategy, as used in many other works such as [31]. In particular, we removed punctuations, lower-cased the tokens, and conducted sentence segmentation using the NLTK library⁸. We also removed words whose term frequency in the entire collection was less than 10. A script has been made available online⁹, which generates an exact copy of our data set, starting from the publicly available dumps of Wikipedia and Wikidata. The implementation of all variants of our model has also been made available at the same link.

Most knowledge graph embedding models have been evaluated on fragments of Freebase and WordNet. Our choice of Wikidata is motivated by the fact that it has relatively clean semantic type information. For example, while Barack Obama is of type *Human* on Wikidata, Freebase among others mentions the following types: *film subject*, *musical artist* and *building occupant*. Furthermore, while Freebase contains information about tens of millions of entities, the standard benchmark datasets, called FB15k [2] and FB13 [28], are relatively small: FB15k covers 14,951 entities and 1,345 relation types, while FB13 covers 74,043 entities and 13 relation types. For completeness, we will include a comparison of our model on these standard benchmark sets for link prediction and triple classification, which are the two standard evaluation tasks for knowledge graph embedding. Our other experiments will be oriented more towards evaluating the usefulness of our model for learning conceptual space representations, in particular their ability to capture semantic relations between entities of the same type (which are not covered in the knowledge graph). This requires a sufficient number of entities for each of the considered semantic types, and a sufficiently clean semantic type structure. Accordingly these tasks will be evaluated only on the WikiData fragment described above. Finally, WordNet has a rich semantic type hierarchy, but contains relatively few instances of these types (e.g. of the 51K leaf nodes in WordNet 1.7 only 7K were found to be instances in [1]) and is thus not suitable for our purposes.

The semantic types of the entities occurring in FB15k and FB13 have been obtained from the “type/instance” field in the Freebase dump¹⁰. To link Freebase entities to Wikipedia, we have made use of existing Freebase-WikiData mappings¹¹.

4.2 Variants and baseline methods

Our main baseline is pTransE, which also learns an embedding of entities by combining a word embedding model with a knowledge

name	type	relation	regularization
EECS _{full}	J_{type}	J_{rel}	$J_{reg}^1 + J_{reg}^2$
EECS _{no rel}	J_{type}	-	J_{reg}^1
EECS _{no type}	-	J_{rel}	J_{reg}^2
EECS _{no NN}	-	J_{rel}	-
EECS _{text}	-	-	-
EECS _{rel-dim}	J_{type}	J_{rel}^{dim}	$J_{reg}^1 + J_{reg}^2$
EECS _{rel-dist}	J_{type}	J_{rel}^{dist}	J_{reg}^1
EECS _{type-comb}	J_{type}^{comb}	J_{rel}	$J_{reg}^1 + J_{reg}^2$
EECS _{type-dist}	J_{type}^{comb}	J_{rel}	J_{reg}^2

Table 1: Overview of considered variants of our model.

graph embedding model. We used the it’s publicly available implementation¹². We consider three variants of this baseline: pTransE_{anch} is the version proposed in [31], which uses anchor text for aligning word vectors and entity vectors; pTransE_{art} is the improvement proposed in [35], which uses the words in the Wikipedia article d_e instead of anchor text (and a slightly different model); pTransE_{full} is a variant of pTransE_{art}, which uses the bag of words representation W_e instead, as in our method. In addition, we compare our method against RESCAL, as well as a number of knowledge graph embedding methods: TransE, TransH, TransR and CTransR. The source codes of these translation-based models are publicly available online¹³. RESCAL [23] is a collective matrix factorization model based on tensor factorization, which has been designed to account for the inherent structure of dyadic relational data. The implementation of RESCAL can be found here¹⁴. For the knowledge graph embedding methods, we used Bernoulli sampling for selecting negative examples (see [31]); we also obtained results for uniform sampling (not shown), and found the results to be very similar to Bernoulli sampling but slightly worse. It is expected that all of these methods will perform worse than both pTransE and our model, as they cannot exploit the text representation W_e of the entities. We also compare our method with Skip-gram and CBOW, which can only use text representations and are thus also expected to perform worse. In particular, to apply these models to learn entity embeddings, we use the same method as for our model to determine entity mentions on Wikipedia, and then apply the standard models based on the words surrounding these mentions. Finally, we have compared our method with the multi-dimensional scaling (MDS) based method from [6], in which case we learn a separate vector space for every semantic type. Because of the limited scalability of the latter model, however, we have only considered this for semantic types with up to 10000 instances. Following [6], for each of the remaining semantic types, a vector space representation of the corresponding entities was obtained using Positive Pointwise Mutual Information (PPMI). We then applied multi-dimensional scaling to obtain a lower-dimensional representation, using the angular difference between the initial vectors as metric. We have used the MDS model implemented in MATLAB. We have also considered the method from [14] as an additional baseline, but found that this method could not scale to even the reduced data set that we used for the MDS experiments.

Throughout this section, we will refer to our model as EECS (Entity Embeddings with Conceptual Subspaces). As an ablation study, we will consider a number of variants of our model in which some components have been removed. EECS_{full} refers to our full model, in which J_{type} is used for modelling semantic types and J_{rel} is used for modelling relations; EECS_{no rel} refers to a variant in which J_{rel}

⁸ <http://www.nltk.org/>

⁹ <https://github.com/bashthebuilder/ECAI-2016/blob/master/README.md>

¹⁰ <https://developers.google.com/freebase/data>

¹¹ <https://developers.google.com/freebase/data#freebase-wikidata-mappings>

¹² https://github.com/Mrlyk423/Relation_Extraction

¹³ https://github.com/Mrlyk423/Relation_Extraction

¹⁴ <https://github.com/mnick/rescal.py>

and the associated regularization component J_{reg}^2 , have been removed; $EECS_{no\ type}$ refers to a variant in which J_{type} and J_{reg}^1 have been removed; $EECS_{no\ NN}$ refers to a variant in which the regularization component J_{reg} has been removed (which also trivializes the component J_{type}); $EECS_{text}$ refers to a variant in which only the component J_{text} is used, reducing our model essentially to a variant of GloVe.

Furthermore, we have considered a few variants of $EECS_{full}$ in which we change the component J_{type} or J_{rel} by one of the proposed alternatives: $EECS_{rel-dim}$ refers to a variant in which J_{rel} is replaced by J_{rel}^{dim} , $EECS_{rel-dist}$ refers to a variant in which J_{rel} is replaced by J_{rel}^{dist} , $EECS_{type-comb}$ refers to a variant in which J_{type} has been replaced by J_{type}^{comb} , and $EECS_{type-dist}$ refers to a variant in which only distance information is considered for modelling semantic types (which corresponds to using J_{type}^{comb} without regularization). An overview of the considered variants of our model is provided in Table 1.

4.3 Methodology

All experiments were evaluated using five-fold cross validation. For tuning the parameter β of our model, based on a tuning/validation set in each experiment, we considered the range $\{50, 100, 150, 200, 250, 300, 350, 400\}$. For the parameter α , we considered values between 0 and 1 with an increment of 0.1. The number of iterations for all models was set to 20, as we found that beyond this number empirical results became fairly consistent in all cases. Based on the tuning set, in each of the experiments the optimal value of β was found to be 300, while the optimal values of α varied between 0.4 and 0.7. The number of dimensions was always set to 300 for our model, noting that because of the nuclear norm regularization this only represents an upper bound on the actual number of dimensions. All parameters of the baseline methods, including the number of dimensions, have been optimized based on the tuning set in each experiment. For the MDS method, the number of dimensions was tuned for each semantic type separately (as this method learns a separate vector space for each semantic type), considering the range from 10 to 100 in steps of 10. For the remaining baselines, which construct a single vector space, the number of dimensions was varied between 50 to 300 in steps of 50.

Our model has been implemented in C using standard POSIX threads, which helps scale our implementation to large text collections. For example, for the considered 1.2 million Wikidata entities our full model takes about 30 minutes per iteration using 8 threads, scaling almost linearly in the number of entities. In contrast, $EECS_{text}$ takes about 18 minutes for each iteration using 8 threads.

4.4 Results

We will evaluate our model on four different tasks: *ranking*, *induction*, *analogy making*, and *knowledge graph embedding*. The first two of these tasks are directly aimed at evaluating to what extent the type-specific subspaces learned by our model are useful as conceptual space representations. In particular, *ranking* will evaluate to what extent important features of a given semantic type can indeed be modelled as directions in the associated subspace, while *induction* assesses to what extent we can use these representations to find new instances of a given concept, given only a few example instances. The *analogy making* task is aimed at evaluating how well the different subspaces are aligned. As discussed above, these first three tasks will be evaluated using a large fragment of WikiData. The motivation behind the fourth tasks relates to the observation that even though our

Semantic Type	Number of Entities	NN-Dimensions
human	191211	288
railway station	4120	121
house	2762	136
organization	1379	88
national park	1307	56
building	1269	52
food	1155	55
college	858	33
automobile	31	12
candy	10	2

Table 2: Number of dimensions selected by the nuclear norm (NN) regularization component of our model for some of the semantic types.

	Ranking ρ	Induction			Analogy
		MAP	P@5	MRR	Acc.
Skip-gram	0.155	0.176	0.356	0.505	0.184
CBOw	0.159	0.182	0.350	0.500	0.213
RESCAL	0.081	0.020	0.189	0.423	0.371
TransE	0.110	0.060	0.200	0.451	0.382
TransH	0.142	0.072	0.210	0.415	0.382
TransR	0.100	0.102	0.302	0.489	0.378
CTransR	0.122	0.132	0.323	0.499	0.402
pTransE _{anch}	0.099	0.101	0.301	0.488	0.476
pTransE _{art}	0.202	0.218	0.475	0.751	0.512
pTransE _{full}	0.213	0.224	0.490	0.756	0.532
$EECS_{full}$	0.319	0.231	0.609	0.883	0.591
$EECS_{no\ rel}$	0.301	0.229	0.588	0.868	0.552
$EECS_{no\ type}$	0.266	0.225	0.585	0.854	0.549
$EECS_{no\ NN}$	0.258	0.220	0.581	0.843	0.545
$EECS_{text}$	0.254	0.218	0.579	0.831	0.540
$EECS_{type-comb}$	0.312	0.231	0.601	0.883	0.595
$EECS_{type-dist}$	0.295	0.231	0.585	0.858	0.550
$EECS_{rel-dim}$	0.309	0.225	0.585	0.859	0.551
$EECS_{rel-dist}$	0.299	0.225	0.585	0.855	0.549

Table 3: Experimental results for the full WikiData test data.

	Ranking ρ	Induction			Analogy
		MAP	P@5	MRR	Acc.
MDS	0.101	0.121	0.231	0.388	0.354
$EECS_{full}$	0.218	0.140	0.301	0.463	0.456

Table 4: Comparison with MDS on a subset of the WikiData test data.

motivation was rather different from the motivation behind pTransE, both our model and pTransE combine a word based entity embedding component with a knowledge graph embedding component. Since pTransE has proven a successful approach for knowledge graph embedding, we want to analyse whether our model has any advantages in such a setting. As explained above, for this task we will use the standard benchmark datasets FB15k and FB13.

An important aspect in the discussion of the result is to analyze the effectiveness of nuclear norm regularization in identifying the most appropriate number of dimensions for each of the semantic types. To illustrate the behaviour of this regularization component, Table 2 shows the number of dimensions that was found for a few notable semantic types (when using the default configuration of our model). As expected, semantic types with more entities generally end up being associated with higher-dimensional subspaces, but other factors affect the choice as well. For example, note that *house* has fewer instances than *railway station*, while being represented by a higher-dimensional subspace. Intuitively, this reflects the idea that the type *house* is more diverse or complex than the type *railway station*.

4.4.1 Ranking

A characteristic feature of conceptual spaces is that they are encoded as Cartesian products of interpretable dimensions. For a vector space model to be meaningful as a conceptual space, it is therefore important that salient properties can be modelled as directions¹⁵. Therefore, we have evaluated the ability of our model to correctly rank entities according to a given property. As we need the ground truth, we have focused on properties with numerical values which are available in Wikidata (but have not been considered when learning the space), e.g. the *date of birth* for entities of type *human*, or the *boiling point* of entities of type *chemical element*. In total, we have retrieved 26 numerical attributes which are available for at least 30 entities. Some of these numerical attributes appear for different semantic types (e.g. the property *inception*, referring to the foundation year, applies to the semantic types *film*, *organization* and *country*, among others). In total, we obtained 73 such property-type combinations, by considering for each numerical attribute all the maximally specific semantic types with at least 30 instances that have the attribute. Each of these 73 combinations was considered as a problem instance. For each problem instance, the corresponding set of entities is split into 60% training, 20% validation and 20% testing sets. The full specification of the 73 problem instances and corresponding splits is available online¹⁶. From the training set, a direction is estimated using the SVMRank model¹⁷ [18]. The parameters of the resulting ranking models are optimized using the validation sets. Table 3 shows the performance on the testing set, in terms of Spearman’s ρ ¹⁸, expressing the correlation between the ranking predicted by the model and the ranking according to the numerical values found in Wikidata.

The results show that standard word and knowledge graph embedding models are not competitive, which is not surprising given that they use less information than our model. However, the results also show that our model substantially outperforms pTransE, even the variant pTransE_{full} which uses the same input as our model. Comparing the results for the variants of our model, we notice that the relation component only has a small impact, i.e. the performance of EECS_{no rel} is close to EECS_{full}, and similarly, the performance of EECS_{no type} is close to EECS_{ext}. Regarding the variants of J_{type} , EECS_{full} and EECS_{type-comb} are clearly better than EECS_{type-dist}, which shows the importance of nuclear norm regularization for identifying low-dimensional subspaces. The results in Table 4 compare our model against the MDS model from [6] on a reduced set of 27 problem instances. These results clearly show that the MDS method is not competitive.

Tables 5 and 6 illustrate the results of the ranking experiment for three attributes, by showing the 5 lowest and 5 highest ranked entities respectively. Note that Table 5 starts with the lowest ranked entity (i.e. the entity that has the lowest value for the considered attribute), while Table 6 starts with the highest ranked entity (i.e. the entity that has the highest value for the considered attribute). While the rankings are not perfect (e.g. Bermuda, Monaco, San Marino and Barbados are all less populous than Malta, Ptolemy lived around 500 years after Plato), the model’s ability to separate high-scoring entities from low-scoring entities is nonetheless remarkable, considering that none

Population	Inception	Date of Birth
Malta	General Electric	Valmiki
Bermuda	IBM	Jesus Christ
Monaco	Hewlett Packard	Cleopatra
San Marino	Microsoft	Ptolemy
Barbados	Oracle Corporation	Plato

Table 5: Five lowest ranked entities for a number of ranking problem instances.

Population	Inception	Date of Birth
China	Alphabet Inc.	Prince George of Cambridge
India	Tencent Holdings	Isabela Moner
USA	Facebook, Inc.	Justin Bieber
Soviet Union	Uber	Lionel Messi
Brazil	Amazon.com	Kim Kardashian

Table 6: Five highest ranked entities for a number of ranking problem instances.

of the information that was used to learn the vector space explicitly referred to these attributes.

4.4.2 Induction

A second characteristic feature of conceptual spaces is that properties correspond to convex regions. Moreover, it is often assumed that the boundaries of these regions are determined based on the distance to a particular point in the space, which acts as a prototype. In this experiment, we test our method’s ability to make inductive inferences based on this view. In particular, given a number of entities of the same type which have some property in common, the task we consider is to identify other entities that also have this property (without any knowledge about the property being considered).

Problem instances in this case were obtained by omitting all triples of the form (\cdot, r, f) for particular choices of r and f , when learning the embeddings. The set of entities e for which $(e, r, f) \in G$ then defines a problem instance. For example, the property being considered could be “films directed by Stephen Spielberg”. Given a few examples of such films, the task is to identify other films directed by Stephen Spielberg (but without the knowledge that this is the property being considered). For each (r, f) combination, the set of entities $\{e \mid (e, r, f) \in G\}$ is split into 60% training, 20% tuning and 20% testing sets. Details on the (r, f) combinations and associated splits are available online¹⁹.

For evaluation purposes, we consider this task as a ranking task. In particular, for each problem instance, we rank the entities of the associated semantic type (defined as the most specific semantic type that contains all the considered entities) based on their distance to the center-of-gravity of the training instances, and evaluate the quality of this ranking using mean average precision (MAP), Precision@5 (P@5) and Mean Reciprocal Rank (MRR); note that in all cases, higher values are better.

The results in Table 3 show that our model again substantially outperforms all of the baselines. Note, however, that in the case of MAP, the differences with pTransE_{full} are rather small. The fact that the differences are clearer for P@5 and MRR suggests that our method is better able to select a few entities with high precision. The MAP score tends to be dominated by outliers, leading to smaller differences. Regarding the different variants of our model, the semantic type component and relation component now play a more equal role, given the rather similar performance of EECS_{no rel} and

¹⁵ Conceptual space representations also encode information about the correlation between the underlying dimensions, which in our case is captured by the angles between these directions.

¹⁶ <https://github.com/bashthebuilder/ECAI-2016>

¹⁷ https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

¹⁸ The reported average ρ values have been obtained using the Fisher z-transformation.

¹⁹ <https://github.com/bashthebuilder/ECAI-2016>

EECS_{no type}, although semantic type information is still more important than the knowledge graph information (as EECS_{no rel} performs better than EECS_{no type}). As for the ranking experiment, we notice that using J_{type}^{dist} in EECS_{no type} leads to worse results, highlighting again the importance of nuclear norm regularization. A before, the MDS model is not competitive.

4.4.3 Analogy making

Finally we have considered the problem of completing analogical proportions of the form “ a is to b what c is to ...”, which is a standard evaluation task for word embeddings. Our main aim in this task is to evaluate how well different subspaces are aligned. We have used the test sets from the GloVe project²⁰ that are about entities, resulting in a total of 8363 problem instances. As there is no need for training data, in this case we randomly split the data into 25% tuning and 75% testing sets. Full details on the test sets and splits that were used have been made available online²¹.

The results are largely consistent with the findings from the previous two experiments. The main difference is that the variant EECS_{type-comb} slightly outperforms EECS_{full} in this case. The rather large difference in performance between EECS_{type-comb} and EECS_{type-dist} again clearly illustrates the impact of nuclear norm regularization on the results.

4.4.4 Knowledge graph embedding

We have also conducted two knowledge graph embedding experiments using the benchmark datasets FB15k and FB13. In particular, we have evaluated our method on the widely used Link Prediction and Triple Classification tasks.

Link prediction For the link prediction task [2], given an entity e and a relation r , the aim is either to find an entity f such that (e, r, f) or to find an entity f such that (f, r, e) . We have used the standard FB15k test set for this evaluation, which allows us to compare our model with the published results of the state-of-the-art knowledge graph embedding models. Two widely used evaluation metrics, which we will also use, are the average rank of correct entities, called “Mean Rank”, and “HITS@10”, which is defined as the proportion of test triples in which the target entity was ranked in the top 10. Note that the Mean Rank score is to be minimized while the HITS@10 score is to be maximized. We have used the standard evaluation protocol, including the Bernoulli sampling trick to corrupt the head or tail entity. Test instances were not filtered (which corresponds to the so-called raw version of the task).

In Table 7 we show that our model clearly outperforms the standard baselines in both metrics. To a large extent, this is due to the fact that, apart from pTransE, the other models do not exploit the bag-of-words representations of the entities. Note that we do not show results for EECS_{text} and EECS_{no rel} in the tables because these models do not take into account any input from the knowledge graph, and is therefore not suitable. The baselines Skip-gram and CBOW are not considered for the same reason.

Triple classification The objective in triple classification [28] is to judge whether a given triplet (e, r, f) is correct or not, i.e. whether

Models	Link Prediction (FB15k)		Triple Classification	
	Mean Rank	HITS@10	FB13	FB15k
RESCAL	683	44.1	65.3	71.6
TransE	125	47.1	81.5	79.8
TransH	87	64.4	83.3	79.9
TransR	77	68.7	82.5	82.1
CTransR	75	70.2	-	84.3
pTransE _{anch}	58	84.6	73.3	74.3
pTransE _{art}	55	85.3	75.8	75.5
pTransE _{full}	51	86.4	76.3	77.4
EECS _{full}	48	89.7	83.1	89.6
EECS _{no type}	56	84.7	71.2	82.1
EECS _{no NN}	59	82.7	70.1	81.4
EECS _{type-comb}	47	89.9	83.3	89.9
EECS _{type-dist}	54	83.2	81.1	82.1
EECS _{rel-dim}	54	85.1	79.3	88.2
EECS _{rel-dist}	52	85.3	78.8	87.1

Table 7: Link prediction and Triple classification results.

entities e and f are in relation r with each other. This can be naturally cast as a binary classification task. We present results on FB13 and FB15k. The FB13 dataset already comes with golden negative triplets, while we followed the methodology from [28] to construct negative samples for FB15k. For the classification task, we need to set a threshold δ_r for each relation r . We obtain δ_r by maximizing the classification accuracies on the validation set. For the given triplet, if the energy score is larger than the relation-specific δ_r , the instance will be classified as positive, otherwise negative. This is the standard experimental setting for this evaluation task.

Our experimental results are shown in Table 7. With the exception of the pTransE variants, we again show the published results for the baseline models. The baseline results have been reported in [33, 15, 17, 22]. On the FB13 dataset, our model matches the performance of the TransH model, although we clearly outperform all baselines on the FB15k dataset. This means that on the FB13 dataset, the bag-of-words representations of the entities cannot be exploited effectively, although our model is still not at a disadvantage. This seems related to the fact that only 13 relation types are considered in FB13, which were moreover specifically selected such that they can be predicted from each other, in the sense that hard-to-predict relation types have been removed [28].

5 CONCLUSIONS

We have proposed a new method for learning vector-space embeddings of entities, based on available semantic information (from Wikidata) and textual descriptions (from Wikipedia). From a technical point of view, the main novelty of our model is the use of nuclear norm regularization to encode the requirement that entities of the same semantic type should lie in a lower-dimensional subspace. In particular, nuclear norm regularization allows the model to automatically select the most appropriate number of dimensions for the subspace corresponding to each type. From an application point of view, our main motivation was to learn subspaces that are useful as approximations of conceptual spaces. To support this view, among others, we have shown that many numerical attributes can be faithfully modelled as directions and that the learned representations allow us to model induction based on distance to a centroid. In addition, we have also obtained good results for analogy making and for two standard knowledge graph embedding tasks.

6 ACKNOWLEDGEMENTS

This work was supported by ERC Starting Grant 637277.

²⁰ <http://nlp.stanford.edu/projects/glove/>

²¹ <https://github.com/bashthebuilder/ECAI-2016>

REFERENCES

- [1] Enrique Alfonseca and Suresh Manandhar, 'Distinguishing concepts and instances in wordnet', in *Proceedings of the First International Conference of the Global WordNet Association*, (2002).
- [2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, 'Translating embeddings for modeling multi-relational data', in *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2787–2795, (2013).
- [3] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, 'Learning structured embeddings of knowledge bases', in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, (2011).
- [4] Antonio Chella, 'A cognitive architecture for music perception exploiting conceptual spaces', in *Applications of Conceptual Spaces*, eds., Frank Zenker and Peter Grdenfors, 187–203, Springer International Publishing, (2015).
- [5] A. Collins and R. Michalski, 'The logic of plausible reasoning: A core theory', *Cognitive Science*, **13**(1), 1–49, (1989).
- [6] J. Derrac and S. Schockaert, 'Inducing semantic relations from conceptual spaces: a data-driven approach to plausible reasoning', *Artificial Intelligence*, 74–105, (2015).
- [7] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, 'Knowledge vault: A web-scale approach to probabilistic knowledge fusion', in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 601–610, (2014).
- [8] I. Douven, L. Decock, R. Dietz, and P. Égré, 'Vagueness: A conceptual spaces approach', *Journal of Philosophical Logic*, **42**, 137–160, (2013).
- [9] Katrin Erk, 'Representing words as regions in vector space', in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pp. 57–65, (2009).
- [10] Maryam Fazel, *Matrix rank minimization with applications*, Ph.D. dissertation, Stanford University, 2002.
- [11] Leon Festinger, *A theory of cognitive dissonance*, Stanford University Press, 1957.
- [12] J. Forth, G. A. Wiggins, and A. McLean, 'Unifying conceptual spaces: Concept formation in musical creative systems', *Minds and Machines*, **20**, 503–532, (2010).
- [13] P. Gärdenfors, *Conceptual Spaces: The Geometry of Thought*, MIT Press, 2000.
- [14] S. Guo, Q. Wang, B. Wang, L. Wang, and L. Guo, 'Semantically smooth knowledge graph embedding', in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pp. 84–94, (2015).
- [15] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao, 'Learning to represent knowledge graphs with gaussian embedding', in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 623–632. ACM, (2015).
- [16] Cho-Jui Hsieh and Peder Olsen, 'Nuclear norm minimization via active subspace selection', in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 575–583, (2014).
- [17] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao, 'Knowledge graph embedding via dynamic mapping matrix', in *Proceedings of ACL*, pp. 687–696, (2015).
- [18] Thorsten Joachims, 'Optimizing search engines using clickthrough data', in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142. ACM, (2002).
- [19] Antonio Lieto, Andrea Minieri, Alberto Piana, and Daniele P. Radicioni, 'A knowledge-based system for prototypical reasoning', *Connection Science*, **27**, 137–152, (2015).
- [20] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu, 'Learning entity and relation embeddings for knowledge graph completion', in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 2181–2187, (2015).
- [21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean, 'Distributed representations of words and phrases and their compositionality', in *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pp. 3111–3119, (2013).
- [22] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson, 'STransE: a novel embedding model of entities and relationships in knowledge bases', in *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, to appear, (2016).
- [23] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel, 'A three-way model for collective learning on multi-relational data', in *Proceedings of the 28th International Conference on Machine Learning*, pp. 809–816, (2011).
- [24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, 'Glove: Global vectors for word representation', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, (2014).
- [25] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo, 'Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization', *SIAM review*, **52**, 471–501, (2010).
- [26] Eleanor H Rosch, 'Natural categories', *Cognitive Psychology*, **4**(3), 328–350, (1973).
- [27] Steven Schockaert and Henri Prade, 'Interpolative and extrapolative reasoning in propositional theories using qualitative knowledge about conceptual spaces', *Artificial Intelligence*, **202**, 86–131, (2013).
- [28] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng, 'Reasoning with neural tensor networks for knowledge base completion', in *Advances in Neural Information Processing Systems*, pp. 926–934, (2013).
- [29] P. D. Turney and P. Pantel, 'From frequency to meaning: Vector space models of semantics', *Journal of Artificial Intelligence Research*, **37**, 141–188, (2010).
- [30] Luke Vilnis and Andrew McCallum, 'Word representations via Gaussian embedding', in *Proceedings of the International Conference on Learning Representations*, (2015).
- [31] Z. Wang, J. Zhang, J. Feng, and Z. Chen, 'Knowledge graph and text jointly embedding', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1591–1601, (2014).
- [32] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen, 'Knowledge graph embedding by translating on hyperplanes', in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1112–1119, (2014).
- [33] Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu, 'TransG: A generative mixture model for knowledge graph embedding', *arXiv preprint arXiv:1509.05488*, (2015).
- [34] C. Xu, Y. Bai, J. Bian, B. Gao, G. Wang, X. Liu, and T.-Y. Liu, 'RC-NET: A general framework for incorporating knowledge into word representations', in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 1219–1228, (2014).
- [35] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen, 'Aligning knowledge and text embeddings by entity descriptions', in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 267–272, (2015).

Automatic Bridge Bidding Using Deep Reinforcement Learning

Chih-Kuan Yeh¹ and Hsuan-Tien Lin²

Abstract.

Bridge is among the zero-sum games for which artificial intelligence has not yet outperformed expert human players. The main difficulty lies in the bidding phase of bridge, which requires cooperative decision making under partial information. Existing artificial intelligence systems for bridge bidding rely on and are thus restricted by human-designed bidding systems or features. In this work, we propose a pioneering bridge bidding system without the aid of human domain knowledge. The system is based on a novel deep reinforcement learning model, which extracts sophisticated features and learns to bid automatically based on raw card data. The model includes an upper-confidence-bound algorithm and additional techniques to achieve a balance between exploration and exploitation. Our experiments validate the promising performance of our proposed model. In particular, the model advances from having no knowledge about bidding to achieving superior performance when compared with a champion-winning computer bridge program that implements a human-designed bidding system.

1 Introduction

Games have always been a challenging testbed for artificial intelligence (AI). Even for games with simple and well-defined rulesets, AI often needs to follow highly complex strategies to gain victory. One set of works on game AI focuses on full information games including chess, go, and Othello [18], whereas the other set studies incomplete information games such as poker and bridge [9, 17, 22]. In both cases, traditional works usually excel by embedding the knowledge of the best human players as computable strategies; however, researchers have recently shifted their focus to machine learning, allowing AI players to develop effective strategies automatically from data [9, 18, 22].

Bridge, a standard 52-card game that requires players to be both cooperative and competitive, is one of the most appraised partial-information games for humans and for AI. The four players of the bridge game are commonly referred to as North, East, West and South, and form two opposing teams (North-South and East-West). Each team aims to achieve the highest score in a zero-sum scenario.

A single bridge game starts with a deal followed by two phases: bidding and playing. A deal distributes 13 random cards to each player, and the cards are hidden from other players—each player only sees partial information about the deal. The bidding phase runs an auction to determine the declarer of the contract, where the contract

affects the score that the declarer's team can get in the playing phase. The auction proceeds around the table in a clockwise manner, where each player chooses from one of the following actions: PASS, increasing the current value of the bid with respect to an ordered set of calls $\{1\clubsuit, 1\diamond, 1\heartsuit, 1\spadesuit, 1NT, 2\clubsuit, \dots, 7NT\}$, DOUBLING and REDOUBLING. The first two actions are general ones for deciding the contract, while the latter two are special, less-used actions that modify the scoring function for the playing phase. The bidding sequence ends when three consecutive PASSes are placed, and the last bid becomes the final contract. The number in the final contract (such as 4 in 4♠) plus 6 represents the number of rounds that the team aims to win in the playing phase to achieve the contract (commonly referred to as “make”), and the symbol (such as ♠) reflects the trump suit in the playing phase.

In the playing phase of the bridge game, there are 13 rounds where each player shows one card from her/his hand and compares the values of the cards based on some rulesets with the trump suit having some priority. The player with the highest-valued card among the four is the winner of the round. After the 13 rounds, the score of the declarer's team is calculated by a lookup table based on the final contract and the number of winning rounds of the declarer's team, where making the contract leads to a positive score for the declarer's team, and not making (failing) the contract results in a positive score for the the opponent's team.

Bidding is an understandably a difficult task because of the incomplete-information setting. Given that each player can only see 13 out of 52 cards, it is impossible for a single player to infer the best contract for her/his team. Thus, each bid in the bidding phase needs to serve as a suggestion towards an optimal contract, information-exchanging between team members, or both. That is, a good bidding strategy should balance between exploration (exchanging information) and exploitation (deciding an optimal contract). Nevertheless, because the bid value needs to be monotonically increasing during the auction, the information exchanging is constrained to the extent that it does not exceed the optimal contract. It is also possible that the two opposing teams may both try to exchange information during the bidding phase, called bidding with competition, which blocks the other team's information-exchanging opportunities.

In real human bridge games, the best human players are often indistinguishable in terms of their professional competence in the playing phase. Thus, their competence in the bidding phase is the primary game-deciding factor. The abovementioned facts indicate the relative difficulty of the bidding phase over the playing phase for human players. The difficulty holds true in the case of designing AI players as well. For the playing phase, it has been shown that AI players are competitive against professional human ones. For example, in 1998, the GIB program finished in the 12th place among 35 professional

¹ Department of Electrical Engineering, National Taiwan University, email: b01901163@ntu.edu.tw

² Department of Computer Science and Information Engineering, National Taiwan University, email: htlin@csie.ntu.edu.tw

human players in a no-bidding bridge contest [8]. Nevertheless, for the bidding phase, most existing AI players are based on replicating human-designed rules of bidding, commonly referred to as human bidding systems [3, 12, 19, 20]. The replication generally makes AI players less competitive to human players in the bidding phase, as explained below.

One of the main difficulties in replicating a human bidding system is the inevitable ambiguity of the bids. Human bidding systems are designed to have rules that cover different situations, but the rules can be overlapping. Therefore, based on the cards of one player and the other players' bids, there can be conflicting suggestions that can be arrived at from different rules, with every suggestion being a legitimate bid under the system. Human players are expected to resolve this ambiguity intelligently and select an appropriate choice from the conflicting suggestions; in addition, professional human players devote a considerable amount of time to practice together with team members to reduce the ambiguity through mutual understanding. When AI players try to replicate human bidding systems, it is extremely challenging to reach the same level of mutual understanding that human players can achieve to resolve the ambiguity, making AI players inferior in the bidding phase.

The ambiguity of the bids arises primarily because human bidding systems need to be simple enough to be memorizable by human players. Thus, the rules within the systems are often simple, too. On the other hand, if there were a bidding system for AI players instead of human players, the rules may not need to be so simple, and the ambiguity issue may be resolved to improve the performance of AI players in the bidding phase.

The aforementioned ideas were the motivation behind some existing works on enhancing AI for bridge bidding. Some works begin by considering a human bidding system and then resolve the ambiguity using different techniques. For instance, combining lookahead search with human bidding system was studied by Gambäck et al. [7] and by Ginsberg [8]. Amit and Markovitch [1] built a decision tree model along with Monte Carlo sampling on top of a human bidding system to resolve the ambiguity of bids. DeLooze and Downey [6] generated examples from a human bidding system, and then used these examples as input for a self-organizing map for ambiguity resolution. In those works, human bidding systems play a central role in AI players' bidding strategy.

A more aggressive route for achieving bridge bidding AI is to teach the AI about bidding without referencing a human bidding system. This was considered by Ho and Lin [9] who proposed a decision tree model along with a contextual bandit algorithm. The model demonstrates the possibility to learn to bid directly in a data-driven manner [9]. Nevertheless, the study is somewhat constrained by the decision tree model, which comes with a restriction of having at most five choices per bid (decision-tree branch). Moreover, because of the simple linear nodes in the decision tree, the model requires a more sophisticated feature representation of the cards. Ho and Lin [9] thus borrowed human knowledge about bidding by encoding the cards as human-designed features, such as the number of cards for each suit. The restrictions on bidding choices and feature representation limit the potential of building a data-driven bidding system for AI.

In all the works discussed thusfar, human-designed features are important either for the human bidding systems within the AI players [6], or for the AI bidding system being learned [9]. In [9], it was actually reported that raw-card features resulted in considerably worse performance than human-designed features. Inspired by the recent success of deep learning in automatically constructing useful features from raw and abstract ones [18], we propose a novel frame-

work that applies deep reinforcement learning for automatic bridge bidding, which contributes to advancing the bridge bidding AI in two aspects:

- learning data representation: Using deep neural networks for feature extraction, our proposed deep reinforcement learning model is the first model that automatically learns the bidding system directly from raw data. Learning from raw data, without relying on human-designed bidding systems or human-designed features, unleashes the full power of machines on using all possible information. The promising performance of our proposed model shows cases that learning a bidding system automatically with no human knowledge is possible.
- resolving bid ambiguity: As discussed previously, the main difficulty of bridge bidding is the ambiguity of the bids. Using the proposed reinforcement learning framework, sophisticated bidding rules can be learned automatically to alleviate the ambiguity problem. The reinforcement learning framework arguably mimics what human players do in establishing mutual understanding by practicing together. In this case, however, the framework does so "together" *with itself*. To the best of our knowledge, it is the first framework that achieves promising mutual understanding using only machines in bridge bidding.

In summary, our proposed deep reinforcement learning framework enables learning complex rules of bidding by nonlinear functions on raw data to avoid ambiguity of the bids and improve bidding performance. In Section 2, we formally establish the problem of bridge bidding as a learning problem. In Section 3, we first introduce reinforcement learning and analyze the key issues in solving the bidding problem. We then propose a novel deep reinforcement learning framework based a modification of Q-learning along with upper-confidence-bound algorithms for balancing between exploration and exploitation. We further introduce a modification of the Bellman's Equation, named penetrative Bellman's Equation, to improve Q-learning. Finally, we discuss our experiments that demonstrate the promising performance of our proposed AI player based on the deep reinforcement learning framework. We demonstrate that the player's bidding performance compares favorably against state-of-the-art AI bidding systems [9] and a contemporary champion-winning bridge software, Wbridge5, which implements a human bidding system.

2 Problem Setup

The general bidding problem can be divided into two subproblems, namely bidding without competition, and bidding with competition [9], both of which contain significant amounts of deals in actual bridge games. Bidding without competition assumes that the opponent's team always calls PASS during bidding, and hence information-exchanging would not be blocked; bidding with competition means that both teams want to bid. In this study, we focus on the subproblem of bidding without competition, as with existing works [1, 6, 9]. The subproblem is a longstanding benchmark for testing computerized bidding systems [12, 19, 20]. It is also called the bidding challenge, which is common for evaluating the performance of human bridge players during practice sessions and some competitions.

For simplicity, we assume that the bidding team is composed of two players, Player 1 and Player 2, sitting at the North-South positions, and their opponents always bid PASS. Player 1 is assumed to bid in round 1 and the other odd rounds without loss of generality, and Player 2 is assumed to bid in the even rounds.

We use \mathbf{x}_1 and \mathbf{x}_2 to denote the cards of Player 1 and Player 2, and \mathbf{b} to denote the bids of the two players. The element $x_i[k]$ of the boolean array \mathbf{x}_i indicates whether Player i holds the k^{th} card in the ordered set of $\{\spadesuit 2, \spadesuit 3, \dots, \spadesuit A, \heartsuit 2, \dots, \heartsuit A, \diamondsuit 2, \dots, \diamondsuit A, \clubsuit 2, \dots, \clubsuit A\}$. Up to the first t rounds, the element $\mathbf{b}^{(t)}[j]$ of the boolean array $\mathbf{b}^{(t)}$ indicates whether Player 1 or Player 2 has made the j^{th} bid in the ordered set of $\beta = \{PASS, 1\clubsuit, 1\diamondsuit, \dots, 7NT\}$. As stated in bridge rules, a new bid has to be higher than all previous bids, and thus it is sufficient to infer which bid was bidden by whom given $\mathbf{b}^{(t)}$.

We define the state $s^{(t)}$ to contain $\mathbf{x}^{(t)}$ and $\mathbf{b}^{(t)}$, where $\mathbf{x}^{(t)} = \mathbf{x}_1$ for an odd t , and $\mathbf{x}^{(t)} = \mathbf{x}_2$ for an even t . The state is defined so that the player bidding in round t can only access her/his own hand. The goal is to find a strategy $G(s^{(t)}) = a^{(t)}$, where the bid $a^{(t)}$ is among the ordered set of $\beta = \{PASS, 1\clubsuit, 1\diamondsuit, \dots, 7NT\}$. Note that a further constraint of $a^{(t)} \geq a^{(t-1)}$ needs to be added to meet the rules of bridge. Furthermore, the ‘‘bid’’ of $a^{(t)} = a^{(t-1)}$ indicates the intent to terminate the bidding procedure, just like the case of $a^{(t)} = PASS$. For any given strategy G , the array $\mathbf{b}^{(t)}$ will be updated by the bid $a^{(t)}$ of the strategy in each bidding round.

We generate the data as follows to learn a good bidding strategy G . For each instance within the data, we directly take the raw card vectors $(\mathbf{x}_1, \mathbf{x}_2)$ as the input features. We also generate the score of each possible contract with respect to each $(\mathbf{x}_1, \mathbf{x}_2)$ to guide learning. Because playing out each possible contract is extremely time-consuming even with computer assistance, the score is calculated without carrying out the playing phase. In particular, we use double dummy analysis as in previous studies [9] to estimate the score for each possible contract.

Double dummy analysis is a technique that attempts to compute the number of tricks taken by each team in the playing phase under perfect information and optimal playing strategy, and is generally considered to be a solved problem in the art of bridge bidding AI. Although the analysis is done with an optimistic assumption of perfect information, it has been shown to achieve considerable accuracy with a more rapid analysis than an actual play.

Nevertheless, it is widely known that goodness of bridge bidding is affected by the opponent team’s hands. A severe distribution of the opponents’ trump cards may cause a good bidden contract to fail. Human players generally bid to maximize the expected score with respect to the opponent team’s hands. We approximate the expected score by randomly dealing the remaining cards to the East and West players for 5 times for each given North-South cards $(\mathbf{x}_1, \mathbf{x}_2)$. Then, we perform double dummy analysis for each deal, and calculate the final score of each possible contract by averaging on the 5 double dummy analysis results. After obtaining the final score for each possible contract, we store the absolute difference between the final score and the highest final score in a cost vector \mathbf{c} , where $c[j]$ indicates the penalty of reaching a final contract j .

We now formally define our learning problem as follows. Given a data set $D = \{(\mathbf{x}_{1n}, \mathbf{x}_{2n}, \mathbf{c}_n)\}_{n=1}^N$, where N is the number of instances, we aim to learn a bidding strategy G . For each $(\mathbf{x}_1, \mathbf{x}_2)$, the strategy G is iteratively fed with the current state $s^{(t)} = \{(\mathbf{x}^{(t)}, \mathbf{b}^{(t)})\}$ until it calls PASS or the same bid at some state $s^{(*)}$. The cost of the final bid (contract), namely $c[G(s^{(*)})]$, is then used to evaluate G . Our goal is to minimize the expected test cost, or equivalently, maximizing the expected test reward of G .

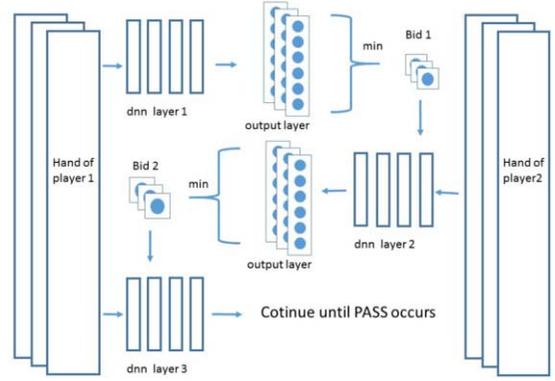


Figure 1. The structure of our bridge bidding deep reinforcement learning framework.

3 Model

The difficulty of creating a bridge bidding AI lies in that evaluation of a bid is not possible until the end of the playing phase. Even after we introduce double dummy analysis, there is still no obtainable ‘‘score’’ until the bidding phase is complete. Therefore, it would be difficult to evaluate bids before the final bid, such as the opening and intermediate bids. It would be most suitable that the intermediate bids are scored by the ability to help the last bid achieve the best score. In this section, we first introduce the reinforcement Q-learning model. Then, we discuss how the model can be extended for conquering the difficulty and solving the bridge bidding problem.

3.1 Reinforcement Q-Learning

Q-learning [21] is a form of the Reinforcement Learning algorithm that does not require modeling of the environment. In Q-learning, the value function of policies are represented by a two-dimensional lookup table indexed by state-action pairs, defined as $Q(s, a)$. The optimal action-value function $Q^*(s, a)$ is defined as the maximum expected return achievable with any strategy after performing an action a in state s . Similarly, $Q^*(s, \mathbf{a})$ is the vector where each value in the vector is obtained by the maximum expected return achievable with any strategy after performing the corresponding action in vector \mathbf{a} and in state s .

The optimal action-value function follows an important equation known as the Bellman equation. The Bellman equation considers that, given the current state s , all possible actions a' , and all resulting states s' , the optimal value of $Q^*(s, a)$ equals the expected sum of the instant reward r and the total rewards after some best action a' is executed. Formally, we have

$$Q^*(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s', a') | s, a] \quad (1)$$

The general idea behind reinforcement learning is to obtain an estimate of the action-value Q-function by continuously modifying it based on feedback from previous actions. After performing an action, the Q-function is updated using the following relation.

$$Q(s, a) \leftarrow \alpha Q(s, a) + (1 - \alpha)[r + \gamma \max_{a'} Q^*(s', a') | s, a] \quad (2)$$

The value iteration algorithm converges to the optimal Q-value eventually. It is important to note that the Q-learning does not specify the next move, and allows arbitrary exploring, because Q-learning constructs a value function on the state-action space, instead of the state space.

For more complicated problems, a non-linear Q-function such as a deep neural network may be used [13, 16]. Usually an error function is introduced in order to measure the difference between the current and newly experienced Q-values. A Q-network can be trained by minimizing a squared-error measure as demonstrated below,

$$L = (Q(s, a) - [r + \gamma \max_{a'} Q^*(s', a') | s, a])^2 \quad (3)$$

Gradient descent techniques such as backpropagation can later be applied to the Q-network in order to minimize the loss function. The update rule is typically applied after each new sample.

3.2 Modeling for Bridge Bidding

The bridge bidding problem we consider can be modeled as a two-player partial-information cooperative game with multiple stages. However, the problem of learning a new bidding system can be complicated. It is considerably very difficult to infer the full state by the bidding sequence alone, because each call can either suggest an optional contract or serve as information exchange between partners. A different intention would lead to different inferences under the same situation, thus partners often maintain a list of agreements about the meaning of their bids, widely known as a bidding system. Interestingly, the same exact bid may be considered good in one bidding system and bad in another.

Therefore, while learning a new bidding system, players would need to modify their interpretation of bids alongside their partners. Moreover, the bidding ability of both players may affect the best possible bid in the early stages. For example, if a player is in placing the last bid, he should be more aggressive in his earlier bids. On the other hand, for a player incapable of bidding well in the later bids, it will be better for him to bid more conservatively early on. This signifies that one's bidding competency in the later stage will affect the bidding in the early stage.

The bidding problem without competition can be viewed as a multi-agent game such that each bidder at different stages of bidding are seen as different players. This model is equivalent to the original bridge bidding problem. In this case, each player has a unique sequence number; players with odd sequence number can share the same 13-card information, while players with even sequence number share another set of 13-card information. Each player knows the bidding result of all the players before him. The game stops when PASS is bid by a player with a sequence number greater than one. Thus, we are able to separate the decision progress of each layer by "training" for a different Q-function for each layer of bidding. The algorithm is defined and illustrated in Algorithm 1.

In traditional Q-learning, the cost of each bid is updated by Bellman equation as in equation 1. Moreover, the exploration behavior is often demonstrated by an ϵ -greedy strategy that follows the greedy strategy with probability $1 - \epsilon$ and selects a random action with probability ϵ . This forms the baseline algorithm.

While Bellman's Equation being a necessary condition for optimality, the convergence time for each Q-function is rather long. Moreover, it has been shown that Q-learning performs considerably poorly in some stochastic environments because of overestimation of action values. In the problem of bridge bidding, being a partial information cooperative game, the overestimation of action values becomes a significant problem.

We define a penetrative Bellman's Equation to deal with the over-

Algorithm 1: The Proposed Learning Algorithm

Input: Data = $\{(x_{1i}, x_{2i}, c_i)\}$ for $i = 1, \dots, n$

Algorithm P to determine cost of action a

Algorithm E to determine exploration and exploitation strategy

Output: A bidding strategy G based on the learned θ_i .

Initialize action-value function Q_j with random weights
for $j = 1, \dots, l$

repeat

Randomly select a data instance (x_{1i}, x_{2i}, c_i)

for round $t = 1$ to l **do**

initialize cost array $c(a^{(t)})$

for all possible action $a^{(t)}$ **do**

determine the cost of action $a^{(t)}$ by P

record resulting cost in $c(a^{(t)})$

save $(S^{(t)}, c(a^{(t)}))$ in Database D

select action $a^{(t)}$ by the highest estimated reward with exploration by E

if $a^{(t)} == PASS$ **then**

Break

update $b^{(t+1)}$ by action $a^{(t)}$

Set $s^{(t+1)} = (x^{(t+1)}, b^{(t+1)})$

for round $t = 1$ to l **do**

Sample random minibatch of $(S^{(t)}, c(a^{(t)}))$ from D

Perform a gradient descent step on

$[(1 - c(a^{(t)})) - Q(s^{(t)}, a^{(t)}; \theta)]^2$

until enough training iterations by early stopping

estimations of action values in the bridge bidding game as follows.

$$Q^{(i)*}(s, a) = \max_{a'} [Q^{(i+1)*}(s', a') | s, a] = Q^{(i+1)*}(s^*, a^* | s, a) \quad (4)$$

where instance reward is zero and $\gamma = 1$. We could further apply Bellman's equation on $Q^{(i+1)*}(s^*, a^*)$ as:

$$Q^{(i+1)*}(s^*, a^*) = \max_{a^{(2)}} [Q^{(i+2)*}(s^{(2)}, a^{(2)}) | (s^*, a^*)] \quad (5)$$

while

$$\max_{a^{(2)}} [Q^{(i+2)*}(s^{(2)}, a^{(2)}) | (s^*, a^*)] = Q^{(i+1)*}(s^{*(2)}, a^{*(2)} | (s^*, a^*)) \quad (6)$$

By combining equation 4, equation 5, and equation 6, we have

$$Q^{(i)*}(s, a) = Q^{(i+2)*}(s^{*(2)}, a^{*(2)} | s, a) \quad (7)$$

where $s^{*(2)}, a^{*(2)}, s^*, a^*, s$, and a satisfy equation 4 and equation 5. By recursively applying Bellman's equation, the process stops when $a^{*(t)}$ is the final bid of the game, and thus, the cost of the game can be decided by the precalculated $c(a^{*(t)})$.

$$Q^{(i)*}(s, a) = Q^{(i+t)*}(s^{*(t)}, a^{*(t)} | s, a) \quad (8)$$

where $s^{*(i)}, a^{*(i)}$ represents the best possible action given $s^{*(i-1)}, a^{*(i-1)}$ due to the Q-function. Typically, there are less than 6 bids in a game of bridge bidding between the two teammates, therefore the penetrative Bellman's Equation is fairly efficient compared with the original variant, while the cost of each action is much more accurate and this leads to better results, as is discussed in the experiment section later.

Because bridge bidding is a multi-agent cooperative game, the traditional ϵ -greedy algorithm would be detrimental for communication

between partners. Note that the main objective in bridge bidding, other than to find the best possible contract, is to convey some information to one’s partner. However, randomly bidding any contract in lieu of a certain possibility would make it difficult for the partner of the bidder to understand the current bidding. This will result in poor communication and slower convergence, which has more disadvantages than it has advantages.

Further, exploration is one of the key elements in reinforcement learning to reach the optimum. Without exploration, reinforcement learning will likely be confined to some local optimal because the value of some actions will never be explored. There are various studies investigating the problem of balancing exploration with exploitation. Previous research on exploration of reinforcement learning proposes the use of a sampling technique such as “Thompson sampling” to enhance the performance of exploring [14, 15].

However, most related approaches are not able to deal with one of the key difference between bridge bidding and tradition reinforcement learning: communication with the partner. One of the difficulties of learning a good bidding strategy is the complexity in exploring the value of an action. In games such as chess or go, one may learn that a move is recommended by evaluating the state through playing afterwards. However, in the game of bridge, a bid is only good if one’s partner understands the bid and is able to react accordingly. Even in the exploration phase, bids would need to be consistent with the opponent’s knowledge.

Moreover, considering information theory, the exchange of information would work best when the use of each bid is distributed equally. Therefore, we design an exploration scheme using a bandit algorithm. The bandit problem has been a popular research topic in the field of machine learning [2, 4, 10, 11]. In the contextual bandit problem, we would like to earn the maximum total rewards within finite tries by pulling a bandit machine from M given machines in a dynamic environment with context. The key is to balance exploration and exploitation. The upper-confidence-bound (UCB) algorithms [4] are some of the most popular contextual bandit algorithms. These algorithms use the uncertainty term to achieve balance. For the bridge bidding problem, we choose to use UCB1 [2] for its simplicity in connecting with deep neural network and its good performance in previous works.

We now relate the bridge bidding problem to the contextual bandit problem. We assume that each possible bid is a bandit machine, with the context being the cards in one’s hand and the previous bids. The reward of each bid is calculated by the penetrative Bellman’s Equation, which relates to the final cost vector and future strategy. Nevertheless, there may be uncertainty in terms of the action-value, especially for bids that are rarely used. Contextual bandit can be applied to balance between using the best action inferred by the Q-function and exploring bids that occurs less. The neural network of the Q-function serves as a non-linear version of the reward, the $W^T X$ term in UCB1. Therefore, we formally define algorithm E using UCB1 by selecting $a^{(t)} = \max_{a \in \mathcal{A}} [Q(s^{(t)}, a^{(t)}; \theta) + \alpha \sqrt{\frac{2 \ln T}{T_a}}]$, where T is the number of total examples used to learn the entire Q, and T_a is the number of examples such that action a (bid a) has been selected. The final algorithm is shown in Algorithm 1.

3.3 Preprocessing and Model Architecture

The features of training the bridge bidding AIs in previous works include bridge-specific features invented by humans such as high card

points³. Deep neural networks are known to contain feature selection function. Therefore, we propose to use 52-dimension raw hand data as our feature. There are debates upon the best form of high card points, since tens and nines may very well serve an important role in certain hands. By using the raw data, we do not limit the computation of the strength of hands by human-designed technique. Moreover, we are able to show that a well-performing bridge bidding system can be designed without human knowledge in bridge.

There are several possible approaches when designing Q-function using a neural network. One approach uses bidding history and the actions as inputs to the neural network, while another involves listing the cost of all possible outputs, only with the state as input. The drawback of the former is that the computation cost will increase linearly with the number of possible actions. Thus, we choose the latter approach and therefore, it can be stated that the output of the Q-function corresponds to a predicted cost vector of all the possible bids. We denote the action vector \mathbf{a} and the true cost vector of all possible bids $\mathbf{c}(\mathbf{a})$. The gradient descent update of the Q-function can be done on $(\mathbf{c}(\mathbf{a}^{(t)}) - Q(s^{(t)}, \mathbf{a}; \theta))^2$. Notably, there may be actions illegal in certain states because they violate the bridge rule. We set the cost of such actions to an extremely high value so that the rule of bridge can be learned by the Q-function explicitly.

We now present the architecture of the Q-function of the bridge bidding problem. We initialize l separate Q-functions, where l is the length of total bids. For the first Q-function, the input is 52-dimension raw data of Player 1’s hand using one-hot encoding, followed by 3 layers of fully connected layers with 128 neurons each. We obtain a 36-dimension output for the cost of each bid. Compared with the first Q-function, in the case of other Q-functions, there is an extra 36-dimension showing the bidding history of the both players, where the 36-dimensions stands for $\{PASS, 1\clubsuit, 1\diamond, \dots, 7NT\}$. The bids that have been bidden by any of the two players have the value one, others have the value zero. The final structure of our learning framework is shown in Figure 1.

4 Experiment

We compare the proposed model with a baseline model, a state-of-the-art model, and a well-known computer bridge software, Wbridge5 [5], which has won the computer bridge championship for several years. We randomly generate a dataset of 140,000 instances to be used in our experiments. We use 100,000 instances for training, and split the other 40,000 instances evenly for validation and testing. We compare the sparse binary features for representing the existence of each card to the condensed feature with second order extension used in previous works [9].

We set the cost vectors $\mathbf{c}(\mathbf{a})$ from International Match Points, which is an integer between 0 and 24 that is commonly used for comparing the relative performance of two teams in most bridge game. The cost vector is obtained by subtracting the cost of action array by the best possible bid followed by a normalization step, that is, $\mathbf{c}(\mathbf{a}) = [\mathbf{c}(\mathbf{a})' - \min_a \mathbf{c}(\mathbf{a})'] / 25$. The result is divided by 25 to ensure the cost is normalized between 0 and 1. $\mathbf{c}(\mathbf{a})'$ denotes the origin cost of each action calculated by the double dummy analysis⁴. The cost can be transformed to the reward using $R(\mathbf{a}) = 1 - \mathbf{c}(\mathbf{a})$. We set the cost of the rule-breaking bids to 1.2, therefore letting the bidding system learn the bidding rules implicitly. Moreover, the bids in the testing phase are chosen from legal bids.

³ High Card Points - total points for the picture cards. A=4; K=3; Q=2; J=1.

⁴ One technical detail is that \mathbf{c} is generated by assuming that the player who can win more tricks in the contract is the declarer

Table 1. Comparisons between the average cost of two exploration methods where 3 values of parameter in each exploration method is tested

layer	2	3	4	5
ϵ -Greedy, $\epsilon = 0.001$	2.9628 \pm 0.0257	2.7748 \pm 0.0328	2.7648 \pm 0.0290	2.7650 \pm 0.0031
ϵ -Greedy, $\epsilon = 0.005$	2.9582 \pm 0.0036	2.8201 \pm 0.0282	2.7773 \pm 0.0419	2.7510 \pm 0.0457
ϵ -Greedy, $\epsilon = 0.01$	2.9857 \pm 0.0227	2.8080 \pm 0.0113	2.7696 \pm 0.0179	2.7716 \pm 0.0305
ϵ -Greedy, $\epsilon = 0.05$	3.0125 \pm 0.0689	2.8331 \pm 0.0413	2.8408 \pm 0.0367	2.8328 \pm 0.0079
ϵ -Greedy, $\epsilon = 0.1$	3.0575 \pm 0.0092	2.8758 \pm 0.0240	2.8679 \pm 0.0228	3.0035 \pm 0.1720
no exploration	2.9600 \pm 0.0372	2.7914 \pm 0.0080	2.7559 \pm 0.0616	2.7949 \pm 0.0358
UCB1, $\alpha = 0.05$	2.9329 \pm 0.0069	2.7776 \pm 0.0128	2.7451 \pm 0.0055	2.7695 \pm 0.0143
UCB1, $\alpha = 0.1$	2.9391 \pm 0.0279	2.7289 \pm 0.0595	2.6984 \pm 0.0207	2.7397 \pm 0.0187
UCB1, $\alpha = 0.2$	2.9542 \pm 0.0052	2.8042 \pm 0.0358	2.7183 \pm 0.0171	2.7465 \pm 0.0221

For deep neural networks, rmsprop is used to speed up the convergence time. In the following experiments, we fix the parameters related to the deep neural network. The following parameters were used in the experiments of the fully connected deep neural network: decay = 0.98, momentum = 0.82, step rate = 0.83, batchsize = 50 and $\eta = 0.05$. These parameters remain unchanged during our experiments because the focus of our study is not on deep neural network parameters. We use early stopping from the validation result to determine the number of epochs to end the training.

4.1 Exploration Method

We compare the two exploration model, ϵ -greedy exploration and UCB1, for the exploration algorithm E in algorithm 1. The parameter in ϵ -greedy exploration is to choose a random action with possibility ϵ and follow the best action given the Q-function otherwise. The parameter in UCB1 exploration is to select $a^{(t)} = \max_{a^{(t)}} [Q(s^{(t)}, a^{(t)}; \theta) + \alpha \sqrt{\frac{2 \ln T}{T_a}}]$, where T is the number of total examples used to learn the complete Q, and T_a is the number of examples such that action a (bid a) has been selected. We perform experiments on $\epsilon \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$ and $\alpha \in \{0.05, 0.1, 0.2\}$. We also list the result where no exploring methods are used.

We can see from the table that the UCB1 exploration generally outperforms that by ϵ -greedy, showing that the UCB1 exploration fits well with the model. The ϵ -greedy exploration method performs even worse than in the case of no exploration for $\epsilon \geq 0.05$, which is arguably because of the enhanced ambiguity of random exploration. It is noteworthy that the no exploration method does include some sense of exploration in the dnn structure itself, because all the possible actions are updated in certain Q-functions, thus they contain the actions which are not likely to be chosen as well. However, deep exploration such as in the case of UCB1 can further improve the result as shown in Table 1. The best parameter for UCB1 exploration is $\alpha = 0.1$ for layer ≤ 3 , and $\alpha = 0.05$ for layer = 2. These parameter values will be used in the experiments discussed in the remainder of this paper.

Table 2. Comparisons between different methods of updating the Q-functions

Total bids	2	3	4	5
Baseline	2.9308	2.8585	2.8795	2.9225
Penetrative Bellman's Equation	2.9329	2.7289	2.6984	2.7397

4.2 Penetrative Bellman's Equation

For the baseline model, we use Bellman's Equation and UCB1 exploration as Algorithm P and Algorithm E in Algorithm 1. We compare

the result of Bellman's Equation and penetrative Bellman's Equation for candidates of algorithm P.

In Table 2, the average test cost is shown with the total bids as variables. UCB1 is used as exploration method with the exploration parameter α set to 0.1. We can see that when the total bids are larger than 2, Penetrative Bellman's Equation outperforms the baseline method by a considerable margin. In fact, the baseline model has little variance of performance with varying total bids. This can be attributed to the cumulation of estimation errors of the Q-function when the total bids increase. Therefore, the performance improvement of the complexity of the model is cancelled out by the cumulated error. We can infer that the primary reason that Penetrative Bellman's Equation is effective is that it enables the possibility to learn a deep Q-learning model with more total bids, by providing a more accurate estimation of cost.

Table 3. Comparisons of average cost between different models and [9] and wbridge5

Model	training	validation	testing
layer = 2	2.8013 \pm 0.0368	2.9150 \pm 0.0049	2.9329 \pm 0.0069
layer = 3	2.6725 \pm 0.0392	2.7363 \pm 0.0465	2.7289 \pm 0.0595
layer = 4	2.5992 \pm 0.0474	2.6700 \pm 0.0245	2.6984 \pm 0.0207
layer = 5	2.6442 \pm 0.0261	2.7150 \pm 0.0123	2.7397 \pm 0.0187
[9] layer = 4	2.9730 \pm 0.0315	3.0697 \pm 0.0388	3.0886 \pm 0.0479
[9] layer = 6	2.9136 \pm 0.0384	3.1267 \pm 0.0092	3.1657 \pm 0.0199
Wbridge5	N/A	N/A	3.0039

4.3 Comparison with the State-of-the-Art

We now discuss the experiment with different model structures. We consider the Q-learning model with total bids $\in \{2, 3, 4, 5\}$. For each model structures, we use the validation result to determine the parameter α , where $\alpha \in \{0.05, 0.1, 0.2\}$. We compare the bridge bidding result with that in the work of [9], and a well-known computer bridge software, Wbridge5 [5], which has won the computer bridge championship for several years. It is known that Wbridge5 adopts some human bidding conventions. It is believed that wbridge5 uses a Monte-Carlo search in the bidding process, so it takes a considerable time to make a bid. The result will be influenced by potential limits on bidding time. On the other hand, despite the long training time of deep neural network, our bridge bidding model is able to decide the bid instantaneously. We run the same 140,000 data on our model and that proposed [9], while running the 20,000 testing data on Wbridge5, using the code provided by the author of [9].

The results in Table 3 shows that the deep reinforcement learning model with layer = 4 has the best performance among all models. Moreover, each deep reinforcement learning model outperforms the result reached by Wbridge5. This showcases that deep reinforcement

Table 4. 5 examples where features of Player 1 is listed in the actual column, and the estimation from Player 2 is listed in the estimate column. The bidding history along with the best bid and cost are listed in the table too.

	actual	estimate	actual	estimate	actual	estimate	actual	estimate	actual	estimate
number of spades	5	5.1024	2	2.2202	3	3.1463	4	4.5931	3	2.7333
number of hearts	2	2.2983	4	4.9368	4	3.5169	3	2.5334	1	1.0620
number of diamonds	3	2.3366	5	2.9966	4	3.8936	2	3.0599	6	5.9515
number of clubs	3	3.2061	2	2.9111	2	3.3676	4	2.8925	3	3.2824
HCP	5	4.9745	21	19.5970	9	9.7680	19	18.5185	5	5.7930
bidding history	$P-1NT-2♠-4♥$		$1♠-1NT-3♠-4♥$		$P-1♣-1NT-1NT$		$1♠-2♣-5♥-6♥$		$P-1♥-2♦-2♦$	
best contract	4♥		4♥ or 3NT		2♦		7NT or 7♥		2♦ or 3♦	
cost(IMP)	0		0		4		11		0	

learning achieves a good result even with a simple model structure. The result justifies that there is indeed a considerable potential to improve the traditional approach of bridge bidding AI by "borrowing" human bidding systems.

4.4 Computational Time

In this section, we discuss the computing time for training our models. The code is written in MATLAB and executed on a Ubuntu Linux 12.04 LTS AMD64 system, using Intel Xeon X5560 CPU with 60 GB RAM. We list the training time for one epochs and the total training time with the model with different layers. In Table 5, we

Table 5. approximate computation time for each model

layer	2	3	4	5
running time per epoch (sec)	121	278	492	713
running time until converge (hrs)	0.5	2.3	6.8	17.9

can observe that the training for models of 2 and 3 layers is quite efficiency, whereas the training time becomes considerably long for larger models. The total converging time is approximately in the order of l^3 , where l is the total layer (or total bid) in the model. This is because the complexity of penetrative Bellman's Equation has an extra order of l compared with the complexity of Bellman's Equation.

4.5 Understanding the Bidding System

One may wonder by learning a new bidding system, is the result of new bidding system understandable? In this section, we show that the learned bidding system is understandable by the bidder's partner, and show the opening table of our bidding system.

We design an experiment testing whether the feature of the partner's hand will be learned along the deep learning training process. In section 3.3, we define the output layer of the dnn as a 36 dimension vector containing the cost of each possible bid. We add a 5 dimension vector to the output layer representing the number of cards in each suit and the high card points, which is usually used as the deciding feature of human bidding. That is, we let our bidding algorithm learn a 5- dimension representative of the partner's hand along with the bidding system. We use the model with total bid = 4 and $\alpha = 0.1$ in the experiment. The results in Table 6 indicate whether adding the

Table 6. The resulting average cost with and without the extra 5 dimension containing partner's hand information in the training objective, where ours* contains the extra 5 dimension

Model	Ours	Ours*
Cost	2.6984 ± 0.0207	2.6910 ± 0.0275

extra 5-dimension vector to the training objective leads to a considerable difference in the average cost. In order to give a more concrete idea of the result, we randomly choose 5 examples in the testing data where the bidding remains for at least 4 turns. After Player 2 received the third bid (which is bid by Player 1), we compare the estimation of the feature of Player 1 by Player 2 with the actual feature of Player 1. This result is shown in Table 4. The result demonstrates that Player 2 is able to precisely estimate the hand of Player 1, even when Player 1 has only made two bids.

We list the opening table for the model trained in Table 7. The result is compared with SAYC and [9]. The abbreviation "bal" refers to a balanced distribution of cards in each suits.

Table 7. Opening Table comparison

Bid	ours	SAYC	[1]
PASS	0-10 HCP	0-11 HCP	0-12 HCP
1♣	11+ HCP	12+ HCP, 3+♣	9-19 HCP, 4-6♥
1♦	10+ HCP, 5+♥	12+ HCP, 3+♦	8-18 HCP, short♠ and 4-6♣
1♥	12+ HCP, 5+♠	12+ HCP, 5+♥	12-23 HCP, w/o long suit
1♠	16+HCP, bal	12+ HCP, 5+♠	10-19 HCP, 4-6♠
1NT	12+ HCP, 6+♦	15-17 HCP, bal	Not used
2♣	Not used	22+ HCP	0-17 HCP, long♣
2♦	Not used	5-11 HCP, 6+♦	0-17 HCP, long♦
2♥	18+ HCP, 5-6♠	5-11 HCP, 6+♥	0-13 HCP, long♥
2♠	Not used	5-11 HCP, 6+♠	0-13 HCP, long♠
2NT	15-17 HCP, 6+♠	20-21 HCP, bal	Not used

4.6 Different Initialization of Opening Bid

Thus far, this paper has been focusing on learning the bidding system without the aid of current human bidding system. In this section, we discuss an initial attempt to study the possibility to use deep reinforcement learning to enhance the current human bidding system. There is an opening bid for each human bidding system; a well-known bidding system is the Standard American Yellow Card (SAYC). The bidding system is used widely by both amateur and professional players. In this section, we implement a structure combining the SAYC open bid and our deep reinforcement learning algorithm by fixing the first bid with a written open table of SAYC. Therefore, our learning algorithm learns all the bids after the opening bid, where the opening bid is fixed. There is a bid called weak bid in SAYC opening serving the purpose of interfering with opponents (2♦, 2♥, 2♠ in Table 7). Because there is no need of a weak bid in this subproblem of bidding with no competitions, we perform experiments on both SAYC opening with and without weak bidding, which is denoted as ours-SAYC and ours-SAYCNW. We run experiment with our best model, where the total bid is 4 and UCB1 α 0.1. The result is compared with Wbridge5 as a comparison baseline.

Table 8. Comparisons of average cost between initialization models

Model	Wbridge5	Ours	Ours - SAYC	Ours - SAYCNW
Cost	3.0039	2.6793	2.8004	2.7795

We can see that in Table 8, our model with SAYC initialization for opening bid outperforms the result of Wbridge5 by a considerable margin. This verifies that our bidding model is able to not only learn well, but also is an improvement on the existing human bidding system. Moreover, our model has a lower cost than the model with SAYC initialization, hinting that using human bridge bidding system may be limiting the potential power of computer bridge bidding.

5 Conclusion and Future Works

We propose a novel model that automatically learns to bid from raw hand data by coupling deep reinforcement learning with improved exploration and update techniques. To the best of our knowledge, our proposed model is the first to tackle automatic bridge bidding from raw data without additional human knowledge. We demonstrate that our proposed model outperforms champion-winning programs and state-of-the-art models by a considerable margin. The superior performance validates the potential of deep learning for reaching a competitive bidding system on its own.

We believe that it is possible to extend our model for the other subproblem: bidding with competition. In particular, the flexibility of the proposed model allows it to improve its bidding strategy, with or without competition, by self-playing as its own opponent team, or by playing with other human or AI teams.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for valuable suggestions. We also thank Chun-Yen Ho and other members of the NTU CLLab for insightful discussions. This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-15-1-4012, and by the Ministry of Science and Technology of Taiwan under number MOST 103-2221-E-002-148-MY3.

REFERENCES

- [1] Asaf Amit and Shaul Markovitch, ‘Learning to bid in bridge’, *Machine Learning*, **63**(3), 287–327, (2006).
- [2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer, ‘Finite-time analysis of the multiarmed bandit problem’, *Machine Learning*, **47**(2-3), 235–256, (2002).
- [3] Gay Loran Carley, *A program to play contract bridge*, Ph.D. dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering, 2004.
- [4] Wei Chu, Lihong Li, Lev Reyzin, and Robert E Schapire, ‘Contextual bandits with linear payoff functions’, in *International Conference on Artificial Intelligence and Statistics*, pp. 208–214, (2011).
- [5] Yves Costel. Wbridge5 bridge software, 2014.
- [6] Lori L DeLooze and James Downey, ‘Bridge bidding with imperfect information’, in *IEEE Symposium on Computational Intelligence and Games*, (2007).
- [7] Björn Gambäck, Manny Rayner, Millers Yard, and Barney Pell. Pragmatic reasoning in bridge, 1993.
- [8] Matthew Ginsberg, ‘Gib: Steps toward an expert-level bridge-playing program’, in *International Joint Conference on Artificial Intelligence*, pp. 584–593, (1999).
- [9] Chun-Yen Ho and Hsuan-Tien Lin, ‘Contract bridge bidding by learning’, in *Proceedings of the Workshop on Computer Poker and Imperfect Information at the Twenty-Ninth AAI Conference on Artificial Intelligence*, (2015).
- [10] John Langford and Tong Zhang, ‘The epoch-greedy algorithm for multi-armed bandits with side information’, in *Advances in neural information processing systems*, pp. 817–824, (2008).
- [11] Wei Li, Xuerui Wang, Ruofei Zhang, Ying Cui, Jianchang Mao, and Rong Jin, ‘Exploitation and exploration in a performance based contextual advertising system’, in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 27–36, (2010).
- [12] Torbjörn Lindelof, *COBRA: the computer-designed bidding system*, v. Gollancz, 1983.
- [13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, ‘Playing atari with deep reinforcement learning’, *arXiv preprint arXiv:1312.5602*, (2013).
- [14] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy, ‘Deep exploration via bootstrapped dqn’, *arXiv preprint arXiv:1602.04621*, (2016).
- [15] Ian Osband and Benjamin Van Roy, ‘Bootstrapped thompson sampling and deep exploration’, *arXiv preprint arXiv:1507.00300*, (2015).
- [16] Martin Riedmiller, ‘Neural fitted q iteration—first experiences with a data efficient neural reinforcement learning method’, in *European Conference on Machine Learning*, pp. 317–328, (2005).
- [17] Tuomas Sandholm, ‘The state of solving large incomplete-information games, and application to poker’, *AI Magazine*, **31**(4), 13–32, (2010).
- [18] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., ‘Mastering the game of go with deep neural networks and tree search’, *Nature*, **529**(7587), 484–489, (2016).
- [19] Alan M Stanier, ‘Bribip: A bridge bidding program’, in *International Joint Conference on Artificial intelligence*, pp. 374–378, (1975).
- [20] Anthony Wasserman, ‘Realization of a skillful bridge bidding program’, in *Proceedings of the Fall Joint Computer Conference*, pp. 433–444, (1970).
- [21] Christopher Watkins and Peter Dayan, ‘Q-learning’, *Machine Learning*, **8**(3-4), 279–292, (1992).
- [22] Nikolai Yakovenko, Liangliang Cao, Colin Raffel, and James Fan, ‘Poker-cnn: A pattern learning strategy for making draws and bets in poker games using convolutional networks’, in *the AAAI Conference on Artificial Intelligence*, (2016).

Hierarchical Strategy Synthesis for Pursuit-Evasion Problems

Rattanachai Ramaithitima¹ and Siddharth Srivastava² and Subhrajit Bhattacharya¹ and Alberto Speranzon² and Vijay Kumar¹

Abstract. We present a novel approach for solving pursuit-evasion problems where multiple pursuers with limited sensing capabilities are used to detect all possible mobile evaders in a given environment. We make no assumptions about the number, the speed, or the maneuverability of evaders. Our algorithm takes as input a map of the environment and sensor models for the pursuers. We then obtain a graph representation of an environment using a Čech Complex. Even with such a representation, the configuration space grows exponentially with the number of pursuers. In order to address this challenge, we propose an abstraction framework to partition the configuration space into sets of topologically similar configurations that preserve the space of possible evader locations. We validate our approach on several simulated environments with varying topologies and numbers of pursuers.

1 Introduction

With the advent of technology, robotics has come to play a significant role in many applications, including autonomous search and pursuit-evasion. In this work, we address the problem of pursuit-evasion where a team of coordinated pursuers needs to search a given environment for an unknown number of evaders or targets. Pursuit-evasion formulations can be used to represent many useful applications such as surveillance or search and rescue. For instance, consider the problem of deploying a team of mobile sensing robots to patrol a military base in order to detect any intruders breaking into the base, or to search for survivors after a disaster. In such scenarios, pursuers must take into account the fact that evaders are mobile and may avoid being detected; they may also know the location of all pursuers at all times and may move faster than the pursuers. As a result, simply checking all areas is not sufficient and one needs to generate sophisticated pursuit strategies.

The pursuit-evasion problem has been studied extensively from many perspectives including differential game theory [8], graph-based search [14, 10, 7], visibility-based search [12, 5, 11, 4], probabilistic search [2, 3, 6, 9, 13, 15], and sensor placement [1]. Isaacs [8] determined sufficient and necessary conditions for a pursuer to capture an evader in the scenario where a pursuer and an evader alternatively take turn moving in finite space.

Parsons [14] pioneered the graph theory aspect of pursuit-evasion problem in 1976 by solving the problem of searching for a lost man in the known structure cave, which he represents as searching on a discrete graph. In this framework, evaders reside on edges and can be adversarial. To detect the evaders, the pursuer must move along the

edge occupied by the evaders and touch the evader. Initially all edges are contaminated, and become cleared if they do not contain any evaders. The edges can also be recontaminated if an evader moves back to the cleared edge without being detected. The pursuers' goal is to find a trajectory that clears all edges.

Later in 2007, Kolling and Carpin [10] proposed an algorithm to solve a similar problem called GRAPH-CLEAR, where the goal is to compute an optimal solution in clearing the graph under special circumstances. Nevertheless, this form of edge-search, where evaders reside on the edges, is not directly applicable to most robotics applications, especially in unstructured environments where construction of the graph is non-trivial. In more recent work, Hollinger et al. [7] discussed an algorithm called GSST for adversarial search where evaders reside on the node. GSST still suffers from reliance on graph construction in unstructured environments and the solution is limited to tree structures, which may extend to non-tree structures by placing some stationary pursuers to break up the cycles.

On the other hand, LaValle et al. [12, 11], Guibas et al. [5] and Gerkey et al. [4] formulated the pursuit-evasion problem as searching in a continuous space where each pursuer has infinite line of sight and the goal of the pursuers is to see all possible evaders. Their method partitions free space based on the critical points, which are vertices of the polygonal obstacles, and then performs a graph-based searching technique. Nevertheless, their approach only works with very few pursuers and the critical point technique is limited to two dimensional environment.

Probabilistic formulations of pursuit-evasion relax the worst-case scenario with the models of evaders or some uncertainty. Hespanha et al. [6] pioneered the probabilistic framework by using a greedy policy to control swarms of robots. Ong et al. [13] proposed an approximate sampling-based algorithm to solve pursuit-evasion as a POMDP problem, while Isler [9] et al. proposed a randomized strategy that allows one to two pursuers to capture any evader in simply connected polygonal environment. On the other hand, Bourgault et al. [2, 3] applied Bayesian filtering approaches to model the motion of a non-adversarial target which were then extended to multiple targets by Wong et al [15].

In sensor networks, Adams and Carlsson [1] use tools from algebraic topology to determine sufficient and necessary conditions to identify the existence of an evasion path given the positions of each mobile sensor in the space-time dimension of two dimensional environments.

In this paper, we propose an alternative approach for solving the worst-case adversarial pursuit-evasion problems where multiple pursuers equipped with limited-range sensors are used to detect and capture all possible mobile evaders in a given environment. Our main ob-

¹ University of Pennsylvania, USA, email: ramar@seas.upenn.edu

² United Technologies Research Center.

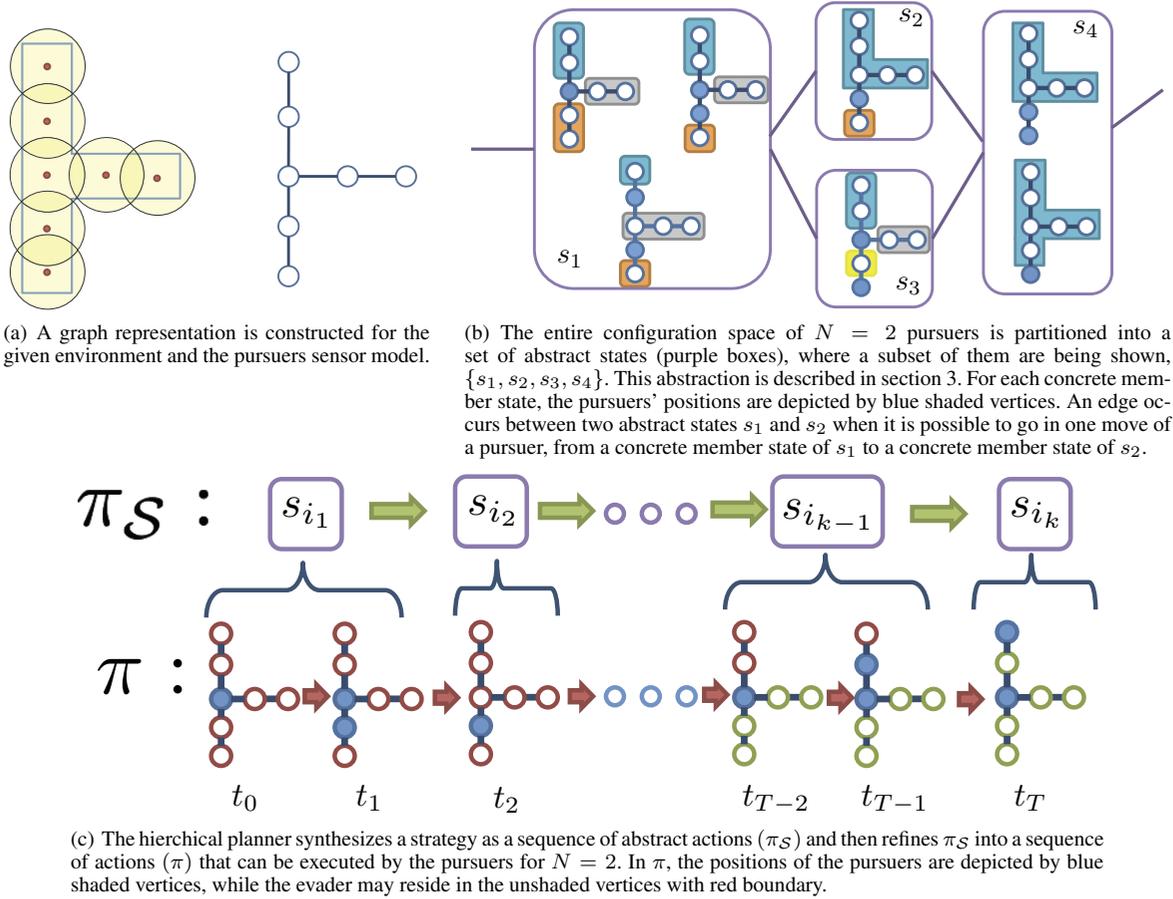


Figure 1: Overview: Illustration of the main steps for synthesize the solution strategy for pursuit-evasion problem with our framework.

jective is to design a general algorithm for automatically computing the strategy that pursuers should follow for detecting all evaders.

The essence of our approach is illustrated in Figure 1. Our algorithm takes as input a map representing the environment, a sensor model of the pursuers, and the number of pursuers. Using the sensor model of the pursuers, we first construct a graph representation of the environment as shown in Figure 1(a). Next, we formulate the configuration space of the pursuers using the graph representation and the number of pursuers N and partition it into the set of abstract states (Figure 1(b)). Finally, we synthesize the strategy as a sequence of abstract actions and then perform the refinement step to map the abstract strategy into the solution strategy in the configuration space of the pursuers as illustrated in Figure 1(c).

We begin with the presentation of the preliminaries including problem description and formulation of pursuit-evasion as a partially observable planning problem in section 2. In section 3, we introduce the abstraction framework. Section 4 presents strategy synthesis over the abstraction framework and our algorithm for retrieving the solution strategy in the original graph representation. We validate our approach with simulations and analysis in section 5.

2 Preliminaries

2.1 Problem Description

We consider the problem of pursuit-evasion (PE) for worst-case adversarial targets with N pursuers, where the number of evaders is unknown and the evader is capable of moving arbitrary fast.

A map is defined as a free space, \mathcal{F} , in an n -dimensional Euclidean space, where n is typically 2 or 3. The position of the i th pursuer is specified by $p_i \in \mathcal{F}$, which can be applied to the sensor model O to get the sensor footprint, a set of points in \mathcal{F} that can be observed from position p_i , $O(p_i) \subseteq \mathcal{F}$. An *evader space* \mathcal{E} is defined as a set of points not being observed by any pursuers,

$$\mathcal{E} = \mathcal{F} \setminus \bigcup_i O(p_i).$$

Each point in the map can be *clear* or *contaminated*. The point is contaminated if an evader could be present in it, otherwise it is clear. The map is said to be clear when all points in \mathcal{F} are clear. A point can be made clear by being observed by any pursuer. However, a clear point $q \in \mathcal{E}$ can become contaminated again if there exists a *path* in the evader space from another contaminated point $p \in \mathcal{E}$ to q , where the path from p to q is defined as a continuous function $\tau_{pq} : [0, T] \rightarrow \mathcal{E}$ such that $\tau_{pq}(0) = p, \tau_{pq}(T) = q$.

The process of clearing and contaminating the map as the pursuers move around is illustrated in Figure 2. Initially, evader can be at any unobserved points, so they are all contaminated. The pursuer then moves forward and clears the points along the path. However, as the pursuer moves further and clear points are exposed to the contaminated ones, they become contaminated again. The objective of PE is then to compute trajectories for each pursuer for clearing all regions in the evader space, where a trajectory of i th pursuer is defined as a continuous function of time, $p_i(t)$ for $t \in [0, T]$.

Definition 1. (Strategy on map) Let \mathcal{F} be a free space representing

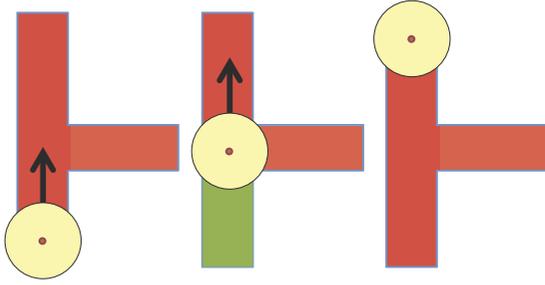


Figure 2: Illustration of the contaminated regions, shaded in red, and cleared regions, shaded in green, as the pursuer moves around in the free space. Initially, evader can be in any regions, so they are all contaminated. The pursuer then moves forward and clear the regions along the path. However, as the pursuer moves further and the cleared regions are connected to the contaminated ones, they become contaminated again.

a map. A strategy (π) is a collection of trajectories for all pursuers, $\pi_{\mathcal{F}} : [0, T] \rightarrow \mathcal{F}^N$, i.e. $\pi_{\mathcal{F}}(t) = [p_1(t), p_2(t), \dots, p_N(t)]$.

Definition 2. (PE problem on a map) Given the map \mathcal{F} with N pursuers with sensor model O , determine a strategy π that clears the map \mathcal{F} .

Synthesizing a solution in continuous space can quickly become intractable, especially when multiple pursuers are required. As a result, we choose to reduce PE problem on a map to PE problem on a graph using Čech complex construction. We will first describe how to construct a graph and then formally introduce PE problem on a graph.

One of the main step in graph construction is choosing a set of representative points such that every point in \mathcal{F} can be observed from at least one of the samples. Ideally, we also want to minimize the size of the representative set. However, this is essentially a minimum set cover problem, one of the well-known NP-complete problems and hence we use a sampling-based method. First, we uniformly distribute the points to cover the convex hull of \mathcal{F} based on the sensor model O . We then keep the sampling points that lie within \mathcal{F} and set aside the rest. Next, we iterate through the points in \mathcal{F} that are not within the sensor footprints of any chosen positions and choose the point in \mathcal{F} closest to the nearest samples from the discarded points.

Assuming that O is convex and the pursuer can move holonomically, we then construct the Čech complex over the sampling points. For Čech complex, a 0-simplex exists for each sampling point; a 1-simplex exists between two 0-simplices whose their corresponding points have a non-empty intersected sensor footprints; and a 2-simplex exists for every 3-tuple of points whose sensor footprints have a non-empty intersection. To assert that we attain the hole-less coverage of the free space, we want the 2-simplices to cover all the points that are sufficiently far away from the obstacle. The points are sufficiently far away if it cannot be observed from the closest boundary of the obstacles.

Definition 3. (Graph representation) $G = (V, E)$, where V is the set of 0-simplices and E is the set of 1-simplices.

Similar to the map, each vertex on G can be either clear or contaminated. The vertex v is clear when pursuer visits. However, v can be recontaminated at any time step if there exists a sequence of unobserved vertices to another point u , i.e. (w_1, \dots, w_k) , where $w_1 = v, w_k = u, (w_i, w_{i+1}) \in E$ and all w_i 's are unobserved. G is clear when all vertices are clear.

The trajectory on a graph is then defined as a sequence of vertices, $(v^0, v^1, \dots, v^T), v^i \in V$ such that $(v^i, v^{i+1}) \in E$. For simplicity, we will discretize the movement of pursuer into time step of 1. In addition, we assume that multiple pursuers can occupied same vertex.

Definition 4. (Strategy on a graph) Let $G = (V, E)$ be a graph. A strategy on graph (π_G or π) is a collection of trajectories on graph, $\pi : \{0, 1, \dots, T\} \rightarrow V^N$, i.e. $\pi(t) = [v_1^t, v_2^t, \dots, v_N^t]$.

Definition 5. (PE on a graph) Let $G = (V, E)$ be a graph. Determine a strategy π that clears G .

Furthermore, the strategies computed on the graph can be translated back into executable trajectories on the map. For any $(u, v) \in E$, a path between u, v is defined as a continuous function $\tau_{uv} : [0, 1] \rightarrow \mathcal{F}$ such that $\tau(0) = u$ and $\tau(1) = v$. Additionally, to prevent v from immediately contaminate u during execution, τ_{uv} must satisfy the following property:

$$\bigcap_{t \in [0, 1]} O(\tau_{uv}(t)) = O(u) \cap O(v),$$

which ensures that the clearing process on the discrete graph would entail the clearing process on the continuous map.

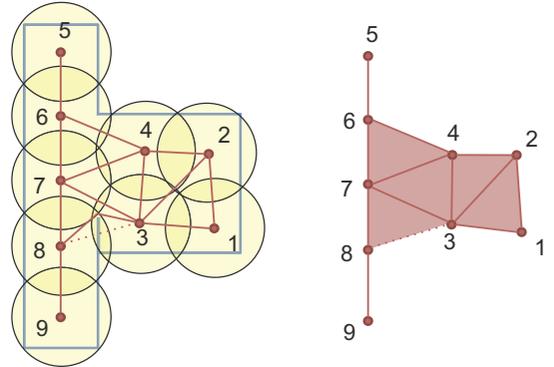


Figure 3: Example of graph representation in 2D environment with holonomic pursuer equipped with circular sensor footprint (left) and its Čech Complex (right). The path between vertices are denoted by solid lines, which are either a straight line or a pair of lines through an intermediate point in the presence of obstacles.

In this paper, we will focus on the holonomic pursuer with sensor model of a ball with radius r . We demonstrate the construction of graph representation using Čech complex on a map with circular sensor model in Figure 3. The path between any vertices will either be a straight line or a pair of straight lines through the point inside the intersection of their sensor footprints due to the presence of obstacles. In both cases, these paths satisfy the property for τ that prevents the immediate contamination during execution. For instance, $\tau_{3,7}$, a straight line from vertex 3 to 7, and $\tau_{3,8}$, a pair of straight lines from vertex 3 to 8, always cover their corresponding intersection as shown in Figure 4.

2.2 Solving PE as a Partially Observable Planning Problem

Since the positions of the evaders are unknown, we cannot fully observe the state during planning. Hence, we introduce the notion of *belief state*. The belief state at any time step, denoted by $x(t)$, consists of the configuration/position of all pursuers, denoted by \mathbf{p} , and

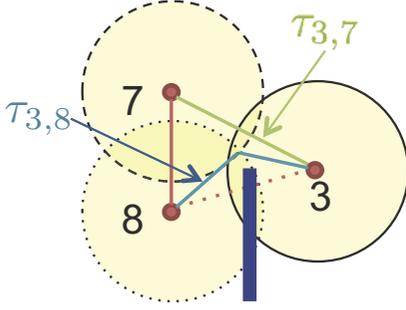


Figure 4: A valid path exist for any edges in G . For instance, any points along $\tau_{3,7}$ and $\tau_{3,8}$ remain observing $O(3) \cap O(7)$ and $O(3) \cap O(8)$ respectively.

the possible positions of the evaders, which will be referred as the *contaminated regions* denoted by \mathbf{c} . We omit the argument in $x(t)$ when it is clear from the context. The collection of all belief states is referred as the *belief space*, X , while the *configuration space*, P , is spanned by the position of the pursuers. The span of contaminated regions will be referred as the *contamination space*, C , so that

$$x = (\mathbf{p}, \mathbf{c})$$

On the graph representation G , the configuration space is spanned by the pursuer positions, $\mathbf{p} = \{p^1, \dots, p^N\}$, where $p^i \in V$. On the other hand, the contamination space can be defined as a set of vertices that the evaders could be present in. Hence, it is a subset of a power of set of V , $\mathbf{c} \in C \subseteq \mathbb{P}(V)$.

The update step occurs when the pursuers take action, i.e. move along an edge in G . With the time discretization on graph, the action can simply be written as the next configuration of the pursuers, \mathbf{p}' , and hence the update function can be defined as

$$\begin{aligned} \text{Update}(x_t, \mathbf{p}') : \\ x_{t+1} = (\mathbf{p}', \text{UpdateContaminate}(\mathbf{c}_t, \mathbf{p}')) \end{aligned}$$

UpdateContaminate updates the contaminated vertices based on the current contamination status and next configuration of the pursuers by computing a set of reachable vertices on $G' = (V \setminus \mathbf{p}', E \setminus \mathbf{p}')$ beginning at $\mathbf{c}_t \setminus \mathbf{p}'$. In addition to all edges that contain occupied vertices, the edge subtraction may require removing some additional edges. The additional edge removal will be explained in section 3.1.

The solution strategy is then a sequence of actions in the belief space such that the contaminated regions become an empty set, i.e. $\pi = \{(\mathbf{p}_0, \mathbf{c}_0), (\mathbf{p}_1, \mathbf{c}_1), \dots, (\mathbf{p}_T, \emptyset)\}$, where $(\mathbf{p}_0, \mathbf{c}_0)$ is the initial state. Solving this as a partially observable planning problem requires search in an intractably large space of belief states, which is exponential in the number of joint pursuer-evader configurations. To address this challenge, we will use a novel abstraction technique that is described in the next section.

3 Abstraction Framework

3.1 Abstract State Space

To cope with the exponential growth of the belief space, we propose a novel method to partition the configuration space into *abstract state space*, denoted by \mathcal{S} , by utilizing the topological invariants of the evader space.

Although the contamination space might appear to be exponential in $|V|$, not all combinations of the contaminated regions are reachable. Since the evader can move arbitrary fast, any adjacent regions in the evader space will both be either contaminated or cleared.

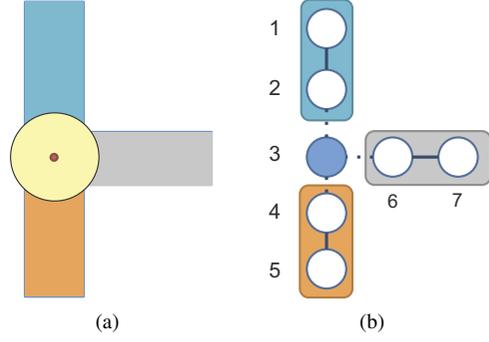
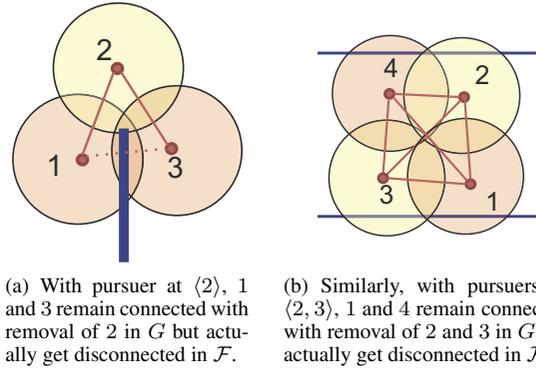


Figure 5: Illustration of an connected component function on G with pursuers at $\mathbf{p} = \langle 3, 3 \rangle$. (a) The sensor footprint is projected onto \mathcal{F} which then separates the evader space into multiple connected components (b) The graph is reconstructed on the evader space where the position of pursuers are depicted by blue-shaded vertices, while the evader space are grouped by shaded boxes for each connected component. $CC(\langle 3, 3 \rangle, G) = \{\{1, 2\}, \{4, 5\}, \{6, 7\}\}$



(a) With pursuer at $\langle 2 \rangle$, 1 and 3 remain connected with removal of 2 in G but actually get disconnected in \mathcal{F} . (b) Similarly, with pursuers at $\langle 2, 3 \rangle$, 1 and 4 remain connected with removal of 2 and 3 in G but actually get disconnected in \mathcal{F} .

Figure 6: Additional edges removal is required when computing connected component on G on these cases.

Utilizing this fact, we define the *connected component (CC)* function which returns the sets of adjacent vertices in the evader space of G based on the assignment of pursuers, \mathbf{p} , denoted by $CC(\mathbf{p}, G)$ or simply $CC(\mathbf{p})$ when G is obvious. The connected component function can be computed by projecting the sensor footprints of the pursuers onto the free space, and then reconstructing the graph representation of the evader space, \mathcal{E} , as illustrated in Figure 5.

The connected component function can also be computed by subtracting the vertices (and their associated edges) occupied by the pursuers from G . Nevertheless, in the present of obstacles, the evader space might remain connected in G while becoming disconnected in \mathcal{F} as illustrated in Figure 6. These exceptions lead to additional edges being removed from G . For 2D environment, there are only two possible scenarios. The first scenario occurs when the intersection of two sensor footprints is completely contained inside the sensor footprint of another vertex (Figure 6(a)). The other scenario occurs when there is a 4-way intersection of sensor footprints, resulting in edge intersection in G (Figure 6(b)).

During graph construction, we can keep track of the intersection between sensor footprints to handle the first scenario, while edge intersection can be easily computed. Hence, the CC function can be computed on G for 2D environment. For higher dimension, the CC function can be computed on G only if all exceptions are tractable. Otherwise, completeness is not guaranteed.

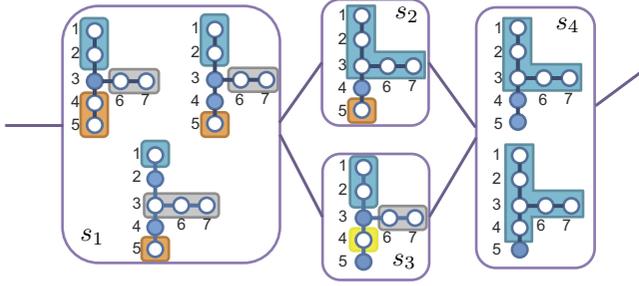


Figure 7: Simple configuration space of two pursuers is partitioned into abstract state space using equivalence relation.

Using the results of the CC function, we want to partition the configuration space into *abstract states* in a way that preserves the topology of the evader space, which is equivalent to the contamination status of each connected component remains unchanged. Using abstract state S_1 in Figure 7 as an example, $\langle 3, 3 \rangle \sim \langle 3, 4 \rangle$ and there exists a one-to-one mapping between $CC(\langle 3, 3 \rangle)$ and $CC(\langle 3, 4 \rangle)$ which preserves the contamination status of each connected component. On the other hand, the edge between two abstract states denotes the transition that does not preserve the topology of an evader space, which could then lead to the changes in contamination status of the evader space. For instance, the edge between S_1 and S_2 represents the transition between $\langle 3, 4 \rangle$ and $\langle 4, 4 \rangle$ which is resulted in two connected components of $CC(\langle 3, 4 \rangle)$ merging and could potentially changes their contamination statuses. We first introduce a relation between two adjacent configurations and then formally define an equivalence relation for partitioning the configuration space as follow.

Let $\mathbf{p} \rightarrow \mathbf{q}$ denotes two adjacent configurations $\mathbf{p}, \mathbf{q} \in P$ s.t. $(p^k, q^k) \in E, \forall k \in \{1, \dots, N\}$. Given two adjacent configurations we can define a relation, which we call *transition relation*, as follows.

Definition 6. (Transition Relation) Let $\mathbf{p}, \mathbf{q} \in P$ s.t. $\mathbf{p} \rightarrow \mathbf{q}$. The transition relation $\rho_{\mathbf{p}, \mathbf{q}}$ between connected components of \mathbf{p} and \mathbf{q} is defined as

$$(cc_i^{\mathbf{p}}, cc_j^{\mathbf{q}}) \in \rho_{\mathbf{p}, \mathbf{q}} \Leftrightarrow cc_i^{\mathbf{p}} \cap cc_j^{\mathbf{q}} \neq \emptyset,$$

where $cc_i^{\mathbf{p}} \in CC(\mathbf{p}), cc_j^{\mathbf{q}} \in CC(\mathbf{q})$.

Given the previous definition we can now formally introduce an equivalence relation between states, which we will use to define a state abstraction.

Definition 7. (Equivalence Relation) For all $\mathbf{r}, \mathbf{s} \in P$ we say that \mathbf{r} is equivalent to \mathbf{s} , or $\mathbf{r} \sim \mathbf{s}$, if and only if there exists a finite sequence $\{\mathbf{z}^i\}_0^T \in P^{T+1}$ with ρ such that

1. $\mathbf{z}^0 = \mathbf{r}$, and $\mathbf{z}^T = \mathbf{s}$;
2. $\mathbf{z}^i \rightarrow \mathbf{z}^{i+1}$, with $i \in \{0, \dots, T-1\}$;
3. $\rho_{\mathbf{z}^i, \mathbf{z}^{i+1}}$ is a bijection.

Hence, the abstract state space can be defined as $\mathcal{S} = P/\sim$, where each abstract state $s_i \in \mathcal{S}$ is a collection of equivalent configurations.

As a result, the contamination status of each connected component could only be changed upon transition between abstract states. Hence, we can synthesize a solution strategy on the abstract state space instead of synthesizing the strategy directly in the configuration space.

In next section, we will describe the algorithm to incrementally construct the abstract state space \mathcal{S} , the function mapping P to \mathcal{S} , denoted by (γ) , and the adjacency matrix of abstract states, denoted by \mathcal{M} .

3.2 Partition Algorithm

Algorithm 1 Partition algorithm

```

1:  $\mathcal{S} \leftarrow \emptyset, \mathcal{M} \leftarrow \emptyset$ 
2:  $Q.Insert(\mathbf{p}_I)$  for some arbitrary  $\mathbf{p}_I$ 
3: while  $Q \neq \emptyset$  do
4:    $\mathbf{p} \leftarrow Q.GetFirst()$ , mark  $\mathbf{p}$  as visited
5:    $\gamma(\mathbf{p}) \leftarrow null$ ,  $Adjacent_S \leftarrow \emptyset$ 
6:   for  $\mathbf{p}' \in Adjacent(\mathbf{p})$  do
7:     if  $\mathbf{p}'$  is visited then
8:       if  $CC(\mathbf{p}) \sim CC(\mathbf{p}')$  then
9:         if  $\gamma(\mathbf{p})$  is null then
10:            $\gamma(\mathbf{p}) \leftarrow \gamma(\mathbf{p}')$ 
11:         else
12:           Resolve conflict if needed
13:       end if
14:     else
15:        $Adjacent_S.Insert(\gamma(\mathbf{p}'))$ 
16:     end if
17:     else if  $\mathbf{p}'$  is unvisited then
18:        $Q.Insert(\mathbf{p}')$ , mark  $\mathbf{p}'$  as alive
19:     end if
20:   end for
21:   if  $\gamma(\mathbf{p})$  is null then
22:      $\mathcal{S}.Insert(Abstract(\mathbf{p}))$ ,  $\gamma(\mathbf{p}) \leftarrow Abstract(\mathbf{p})$ 
23:   end if
24:   for  $a \in Adjacent_S$  do
25:     Update  $\mathcal{M}(\gamma(\mathbf{p}), a)$ 
26:   end for
27: end while

```

Algorithm 1 outlines an incrementally construction of the abstract state space and other components required for synthesizing a strategy. The concept is to perform a forward search over the configuration space P beginning at an arbitrary state \mathbf{p}_I and partition the configuration states into the abstract state, $a \in \mathcal{S}$, based on the output of $CC(\mathbf{p})$.

Following standard forward search algorithm (line 2-6, 17-18), each state in P begins as *unvisited* and will be marked *alive* or *visited* upon inserting to or removing from Q respectively. The set of alive states is stored in the list Q and the search is completed when the list Q is empty. The function $Adjacent(\cdot)$ in line 6 returns the set of adjacent states by moving the pursuers along graph G . In this step, we will restrict the adjacent states to one pursuer movement only.

The partitioning occurs between line 7-15 and 21-25, where we compare the connected components of the current state to the visited adjacent states and either assign the current state to the new abstract state or append it to the existing one.

In general, comparing the connected component between two arbitrary configurations is nontrivial. Comparing those of the adjacent configurations is much simpler. In line 8, we compute the transition relation, $\rho_{\mathbf{p}, \mathbf{p}'}$, as defined in Definition 6 and check whether it is bijective. This transition relation is also used for updating the contaminated regions when transiting between abstract states.

If the graph consists of *cycles*, a conflict might occur when two similar configurations get assigned into two different abstract states. This will be resolved in line 12 where two abstract states will be combined.

Furthermore, the adjacency matrix, \mathcal{M} , is updated based on the connectivity of the corresponding configuration states. In line 15,

$Adjacent_A$ keeps track of the adjacent abstract states which will then update \mathcal{M} in line 25. The adjacency matrix also store the transition relation(s) between two abstract states and the corresponding configurations. The transition relation might not be unique if G consists of cycles.

As a result, the computational complexity of the partition algorithm is approximately $O(dN|V|^{N+1})$, which consists of $O(dN|V|^N)$ from the forward search algorithm over the configuration space of size $|V|^N$ where each configuration has $O(dN)$ adjacent states, d denotes the average degrees of the vertices, and $O(|V|)$ from comparison of evader space in line 8. Although the number might seem large, it is much smaller comparing to searching over original belief space because the partition algorithm is only exponential with respect to the number of pursuers, which is commonly known as *curse of dimensionality* in multi-robot motion planning problem. In the next section, we will explain how to use the output of the partition algorithm to synthesize the solution strategy.

4 Hierarchical algorithm

To synthesize the strategy using the abstraction framework, we first search for the strategy in the abstract state space and then refine the strategy into the configuration space. If the number of pursuers, N is given, planning in abstraction framework would either return the strategy or indicate that no solution exists for the given N . The search for strategy in the abstract state space can be done using existing techniques for graph-based searching such as Dijkstra’s algorithm. We will describe the abstract belief space for planning in the abstract state space in section 4.1, and then discuss the refinement step in section 4.2.

4.1 Planning in the abstract state space

The abstract belief state for the abstract state space, denoted by x_S , becomes a pair of the abstract state (s) and the list of contaminated regions (L), where each region represents a set of adjacent vertices of the evader space.

$$x_S = (s, L), \quad L = \{s^j\}.$$

The update step during planning will keep track of the contaminated regions using the information stored in the adjacency matrix \mathcal{M} . Since the transition relation ρ may not be unique, the update step with input s_k has to be called for each ρ stored in $\mathcal{M}(s_t, s_k)$.

$$\begin{aligned} Update(x_{S,t}, s_k, \rho) : \\ L_{t+1} &= \{s_k^j \mid \exists s_t^i \in L_t, (s_t^i, s_k^j) \in \rho\} \\ x_{S,t+1} &= (s_k, L_{t+1}) \end{aligned}$$

The solution strategy is a sequence of abstract belief states such that the list of contaminated regions eventually becomes empty, denoted by $\pi_S = (a_{i_1}, \{a_{i_1}^j\}), (a_{i_2}, \{a_{i_2}^j\}), \dots, (a_{i_k}, \emptyset)$. We will then describe how to map the solution strategy into the strategy on a graph with the refinement process on the following section.

4.2 Refinement

The information stored in \mathcal{M} provides the boundary configurations representing the transition between abstract states. Given the sequence of abstract states, the entering configuration might not be adjacent to the leaving one. For instance, in Figure 8, the incoming and

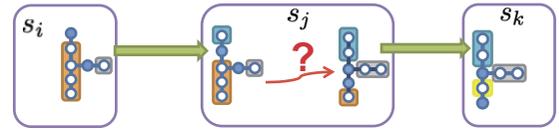


Figure 8: The transition between abstract states represents the action between boundary configurations, however, the incoming configuration need not be adjacent to the outgoing one. Hence, the refinement step is necessary to find the trajectories between them.

outgoing configuration of abstract state s_j are not adjacent. Thus, the refinement step is required to find the trajectory from the incoming to outgoing configuration such that all intermediate configurations are the member of s_j .

Using mapping function γ , one method for refinement is to perform the forward search inside each abstract state to find the trajectory from the incoming to outgoing configurations. However, this method might be inefficient since we already expand the full configuration space while executing partition algorithm.

An alternative method is to store information of the spanning trees of each abstract state during the partition algorithm and then search for the trajectory on the spanning trees during refinement. The downside of this method is that the result is usually suboptimal compared with one from forward search method.

Given the strategy $\pi_A = \{s_0, s_1, s_2, \dots, s_k\}$ where $\gamma(\mathbf{p}_I) = s_0$, the refinement then first searches for a trajectory from \mathbf{p}_I to the boundary configuration connecting to s_1 while remaining within s_0 . Then, we continue refine the solution inside s_1 from the incoming configuration from s_0 to the outgoing configuration connecting to s_2 while remaining within s_1 . The same process continues until we reach the final abstract state s_k .

4.3 Minimizing the number of pursuers

The full algorithm for synthesizing the solution strategy is given in algorithm 2 where the number of pursuers, N , required is unknown. Note that incrementing N by one at each iteration is more efficient than doing binary search because the computational complexity is exponential with respect to N .

Algorithm 2 Hierarchical Strategy Synthesis

- 1: Construct G of free space \mathcal{F} with sensor model $O(\cdot)$
 - 2: $N \leftarrow 1, \pi_S \leftarrow []$
 - 3: **while** π_S is empty **do**
 - 4: Partition G^N into abstract state space \mathcal{S}
 - 5: $\pi_S \leftarrow Planning(\mathcal{S}, \mathbf{p}_I)$
 - 6: Increment N
 - 7: **end while**
 - 8: $\pi \leftarrow Refine(\pi_S)$
-

5 Results

The construction of graph representation is implemented in MATLAB, whereas the remaining components are implemented in C++. We validate the proposed method in simulations with environments of varying topologies and using different number of pursuers as discussed in section 5.1. Then, we compare our results with the graph-based searching over the full belief space on simple environments in section 5.2.

5.1 Simulation Results

We evaluated the performance in simulation on environments with three different topologies as show in Figure 9. Each pursuer has a disc sensor footprint with a radius of one meter. Due to various structures of the environments, we represent their dimensions with the number of vertices in the graph representation. Figure 9(c) illustrates the graph representation of the testing environments and the sensor footprint of the pursuer. The number of pursuers required in each environment is computed by iterating from $N = 1$.

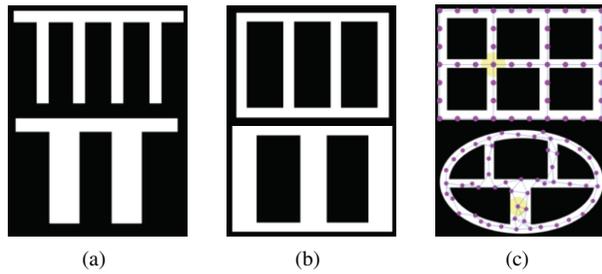


Figure 9: Testing Environments: (a) Tree Structures; (b) Ladder Structures; (c) Random Loops Structures with graph representation

5.1.1 Tree Structures

We evaluated on the tree structures with varying number k and width w of branches (vertical corridors). The graph representations contain 8 – 48 vertices. It requires 2 pursuers to clear the map for $w = 1$ and 3 pursuers to clear the map for $w = 2$. The execution times of each component are illustrated in Figure 10.

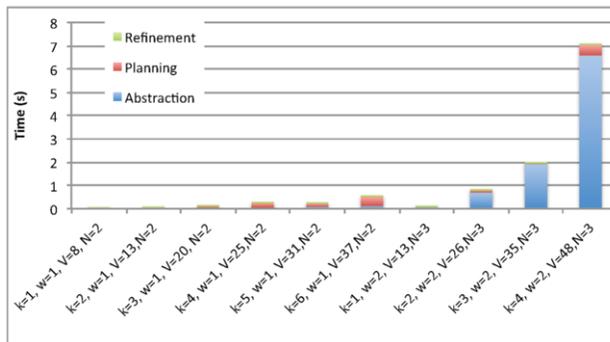


Figure 10: The execution time of the proposed method on tree structure with k branches of width w using N pursuers on graph with V vertices.

5.1.2 Ladder Structures

We evaluated on the ladder structures with varying number k and width, w , of steps (vertical corridors). The graph representations contain 30 – 52 vertices. For ladder with single loop $k = 2$, it requires 2 pursuers to clear the map with $w = 1$ and 3 pursuers to clear the map with $w = 2$. If the ladder with multiple loops $k > 2$, it requires 3 pursuers to clear the map with $w = 1$ and 4 pursuers to clear the map with $w = 2$. The execution times of each component are illustrated in Figure 11.

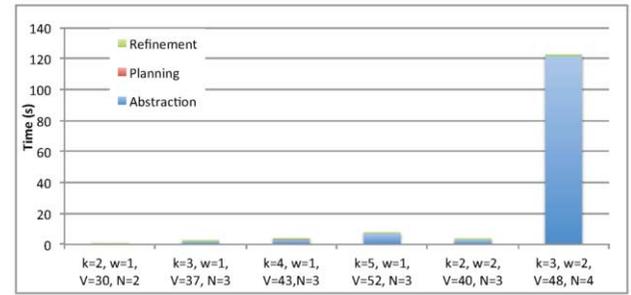


Figure 11: The execution time of proposed method on ladder structure with k steps of width w using N pursuers on graph with V vertices.

5.1.3 Random Loop Structures

We evaluated the proposed methods on two maps shown in 9(c) using 4 pursuers for the top one, which consists of 46 vertices, and 5 pursuers for the bottom map, which consists of 53 vertices. The total execution times are 105.59 and 4076.32 seconds, in which abstraction framework are accounted for 103.25 and 4074.44 seconds respectively. The intermediate steps during the clearing process are shown in Figure 12 and Figure 13.

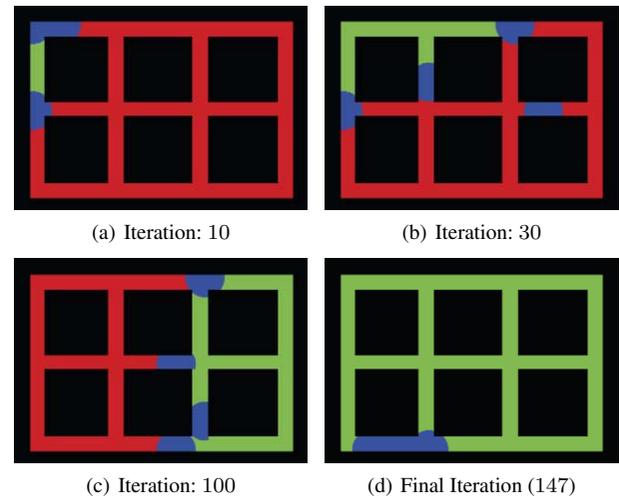


Figure 12: Snapshots of clearing process on 3×4 grid map with all narrow passages using 4 pursuers.

As explained in section 4.2, one the main disadvantages of refinement using spanning tree is the lengthy strategy as indicated by a high number of iteration in both examples. This strategy can be further improved; however, this is out the scope of this paper.

The simulation results show that the abstraction framework is responsible for the majority of computation time as N increases, which conforms with our analysis on computational complexity of the partition algorithm. Nevertheless, the abstraction framework only needs to be executed once for each given map with the same number of pursuers. It is invariant of the initial position and thus we can quickly synthesize the strategy again for different initial position. This can be useful if an error occurs while executing the solution strategy and the pursuers are deviated from the planned strategy.

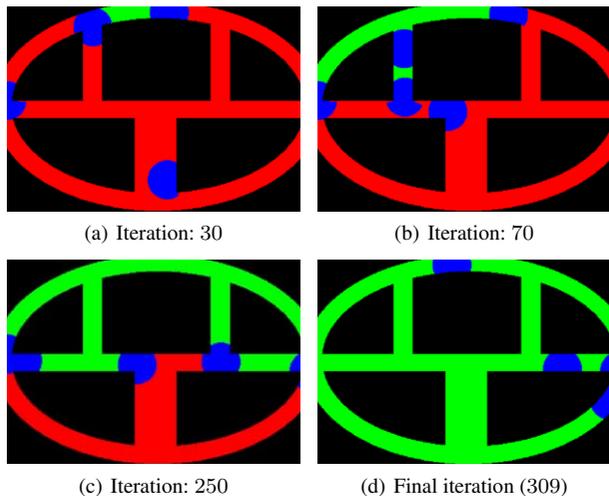


Figure 13: Snapshots of clearing process on curved hallway with vary passages size using 5 pursuers.

5.2 Comparison

We compare the results of our proposed algorithm with a baseline (brute force) planner which searches for a strategy on the full belief space, described in section 2.2. We used maps with tree and ladder structures. The baseline planner can only solve the maps containing up to 2 branches (13 vertices) for tree structure and the maps containing one loop (14 vertices).

Map	Our framework	Baseline planner
Tree with 1 branch, $V=8$	0.014	0.04
Tree with 2 branches, $V=13$	0.034	49.95
Ladder with 2 steps, $V=14$	0.018	1082.77

Table 1: Comparison of execution time (sec) between our framework and baseline (brute force) planner.

The execution time of the baseline planner grows exponentially as V increases. Additionally, the baseline planner suffers greatly from the topological invariant of the loop structure (such as a ladder) due to a large reachable states of the contamination space. On the other hand, our framework reduces the configuration space of 2 pursuers with one loop into only two abstract states; one for two pursuers being adjacent and other when they are separated.

6 Conclusion and Future Work

In this paper, we proposed an abstraction framework to solve a worst-case adversarial pursuit-evasion problem where multiple pursuers with limited-range sensor coverage are used to detect all possible mobile evaders. This method involves constructing the graph representation of an environment using the sensor model equipped on the pursuers, partitioning the configuration space of the pursuers over graph into an abstract state space, searching for a strategy in the abstract state space, and finally refine the strategy into the configuration space. We validate our proposed method by simulating environments with different topologies and compare the result with a brute force searching over the full belief space. Since the full belief space grows exponentially with N and the number of vertices in V , the brute force search can only solve PE problems on small maps ($|V| < 20$) for

$N = 2$. In contrast, our approach reduces the complexity into exponential of N with base $|V|$ and can solve the PE problems with a few hundred vertices for $N = 2$ and up to 50 vertices for $N = 5$.

Although the abstraction framework requires inspecting the full configuration space of N pursuers, this step needs to be done only once for a given map with the same number of pursuers. The output can be reused for different initial configurations. Furthermore, we observe that many maps with the similar structure yield the same abstraction. This opens up an interesting problem of how can we apply the result of one abstraction framework to the new maps with similar structure without recomputing it. Additionally, we are interested in converting an abstraction framework into an on-line algorithms so that we can concurrently synthesize for the solution strategy, which may avoid exploring the entire configuration space.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of ONR grant N00014-14-1-0510 and the United Technologies Research Center.

REFERENCES

- [1] Henry Adams and Gunnar Carlsson, ‘Evasion paths in mobile sensor networks’, *The International Journal of Robotics Research*, **34**(1), 90–104, (2015).
- [2] Frederic Bourgault, Tomonari Furukawa, and Hugh F Durrant-Whyte, ‘Coordinated decentralized search for a lost target in a bayesian world’, in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pp. 48–53. IEEE, (2003).
- [3] Frédéric Bourgault, Tomonari Furukawa, and Hugh F Durrant-Whyte, ‘Optimal search for a lost target in a bayesian world’, in *Field and service robotics*, pp. 209–222. Springer, (2003).
- [4] Brian P Gerkey, Sebastian Thrun, and Geoff Gordon, ‘Visibility-based pursuit-evasion with limited field of view’, *The International Journal of Robotics Research*, **25**(4), 299–315, (2006).
- [5] Leonidas J Guibas, Jean-Claude Latombe, Steven M LaValle, David Lin, and Rajeev Motwani, ‘A visibility-based pursuit-evasion problem’, *International Journal of Computational Geometry & Applications*, **9**(04n05), 471–493.
- [6] Joao P Hespanha, Hyoun Jin Kim, and Shankar Sastry, ‘Multiple-agent probabilistic pursuit-evasion games’, in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 3, pp. 2432–2437. IEEE, (1999).
- [7] Geoffrey Hollinger, Athanasios Kehagias, and Sanjiv Singh, ‘Gsst: anytime guaranteed search’, *Autonomous Robots*, **29**(1), 99–118, (2010).
- [8] Rufus Isaacs, *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*, Courier Corporation, 1999.
- [9] Volkan Isler, Sampath Kannan, and Sanjeev Khanna, ‘Randomized pursuit-evasion in a polygonal environment’, *Robotics, IEEE Transactions on*, **21**(5), 875–884, (2005).
- [10] Andreas Kolling and Stefano Carpin, ‘The graph-clear problem: definition, theoretical properties and its connections to multirobot aided surveillance’, in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 1003–1008. IEEE, (2007).
- [11] Steven M LaValle and John E Hinrichsen, ‘Visibility-based pursuit-evasion: The case of curved environments’, *Robotics and Automation, IEEE Transactions on*, **17**(2), 196–202, (2001).
- [12] Steven M LaValle, David Lin, Leonida J Guibas, Jean-Claude Latombe, and Rajeev Motwani, ‘Finding an unpredictable target in a workspace with obstacles’, in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 1, pp. 737–742. IEEE, (1997).
- [13] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee, ‘Planning under uncertainty for robotic tasks with mixed observability’, *The International Journal of Robotics Research*, **29**(8), 1053–1068, (2010).
- [14] Torrence D Parsons, ‘Pursuit-evasion in a graph’, in *Theory and applications of graphs*, 426–441, Springer, (1978).

- [15] El-Mane Wong, Frédéric Bourgault, and Tomonari Furukawa, 'Multi-vehicle bayesian search for multiple lost targets', in *Proceedings of the 2005 ieee international conference on robotics and automation*, pp. 3169–3174. IEEE, (2005).

Decidable Reasoning in a First-Order Logic of Limited Conditional Belief

Christoph Schwering¹ and Gerhard Lakemeyer²

Abstract. In a series of papers, Liu, Lakemeyer, and Levesque address the problem of decidable reasoning in expressive first-order knowledge bases. Here, we extend their ideas to accommodate *conditional beliefs*, as in “if she is Australian, then she presumably eats Kangaroo meat.” Perhaps the most prevalent semantics of a conditional belief is to evaluate the consequent in the most-plausible worlds consistent with the premise. In this paper, we devise a technique to approximate this notion of plausibility, and complement it with Liu, Lakemeyer, and Levesque’s weak inference. Based on these ideas, we develop a logic of *limited* conditional belief, and provide soundness, decidability, and (for the propositional case) tractability results.

1 INTRODUCTION

Taking into account different contingencies is elementary for humans. For example, when we expect a guest for dinner, we might believe her to be *not* Australian, but at the same time believe that if she *is* Australian, then she presumably eats Kangaroo meat – and we would plan the menu based on these beliefs. It is no surprise that the concept of *conditional belief* is a subject of KR research, for example in belief revision [13].

The goal of this paper is to devise a reasoning service that soundly decides whether a given conditional knowledge base entails some conditional belief. As with any knowledge-based system, this requires to trade off expressivity and/or reasoning power against computational feasibility. We will address this question by beginning with a fully fledged first-order logic of conditional belief, and then weaken its inference mechanism in order to achieve decidability, while retaining the expressivity of a first-order language as well as soundness wrt the fully fledged logic.

Our approach is based on Liu, Lakemeyer, and Levesque’s (henceforth LLL) work on *limited reasoning* [26, 19, 20, 21], where they address the problem of decidable reasoning in expressive first-order knowledge bases. They stratify belief in levels: level 0 only contains the agent’s explicit beliefs; every following level k adds the beliefs that can be inferred after k case splits, that is, by branching on the possible truth assignments of k literals.

Two key features distinguish limited reasoning from other approaches to decidable first-order reasoning such as description logics. Firstly, no limits are set with regard to first-order expressivity in the query. Naturally then, soundness and decidability come at the cost of completeness. Secondly, limited reasoning solves the problem of logical omniscience [11]. Omniscience means that an agent knows

all logical consequences of his knowledge, which is obviously not realistic. LLL avoid omniscience by limiting the number of allowed case splits; intuitively, this number represents the *effort* the reasoner may spend to prove a belief.

Surprisingly perhaps, generalizing LLL’s ideas to the case of *conditional* belief is far from trivial. The following running example shall illustrate this.

Example 1 Suppose we expect a guest for dinner. We don’t know much about her preferred diet yet, but we do have some (somewhat narrow-minded) beliefs:

- Most Australians are not Italians, and vice versa.
- Australians usually eat kangaroo meat.
- We believe our guest is Italian or a vegetarian, and if she is not Italian, she presumably is Australian.
- We know that kangaroo is meat, and that vegetarians do not eat meat.

Given this conditional knowledge base, do we expect our guest to be a vegetarian in case she is not Italian? Monotonic reasoning would suggest so: our belief of her being Italian or a vegetarian yields that she must be a vegetarian if not Italian. But then she also must be Australian, hence eat kangaroo, and thus be a non-vegetarian – that is, in a monotonic logic *everything* is believed if she is not Italian, because the beliefs are inconsistent with that contingency. Conditional beliefs do not trap into inconsistency that easily. They detect that the premise “not Italian” is inconsistent with the most-plausible scenarios, and therefore go on to look for *less-plausible* scenarios where the premise holds, and check the consequent in these scenarios. In our example, we hence believe that if the guest is not Italian, she presumably is an Australian kangaroo-eater, but not a vegetarian.

It is the consistency test that makes reasoning about conditional beliefs more involved than normal beliefs. To understand why, we need some details. An agent’s epistemic state with the contingencies he considers possible can be represented as a *system of spheres* [25, 9] as in Figure 1a (page 3). The innermost sphere contains the most-plausible scenarios, and every outer sphere contains additional less-plausible scenarios. A conditional “if ϕ , then presumably ψ ” is believed iff the material implication $\phi \supset \psi$ holds in the innermost sphere that is consistent with ϕ . LLL provide us with a limited semantics of first-order logic which is suitable for checking if $\phi \supset \psi$ holds in a given sphere. On the other hand, it is insufficient for the consistency check: as the semantics is incomplete wrt classical logic, inconsistencies might go undetected, and wrong beliefs could come out true. For that reason we devise an additional *complete but unsound* semantics complementary to LLL’s: the former can be used to

¹ RWTH Aachen University, Germany and The University of New South Wales, Australia, email: schwering@kbsg.rwth-aachen.de

² RWTH Aachen University, Germany, email: gerhard@kbsg.rwth-aachen.de

soundly determine consistency, and the latter for sound inferences. Together, they will allow us to develop a logic of limited conditional belief.

The rest of the paper is structured as follows. After a survey of related work in the next section, we proceed with the technical part. To start with, we introduce the logic \mathcal{BO} for reasoning about conditional belief; it will serve as a reference point for the following results. Section 4 presents a derivative of LLL's limited semantics of first-order logic, and introduces a novel unsound but complete semantics. The stage is then set for the paper's main contribution in Section 5: a logic of limited conditional belief called \mathcal{BOL} , which is sound wrt \mathcal{BO} and decidable for a large class of knowledge bases and sometimes even tractable. Then we conclude and discuss future work. While we illustrate all techniques using the above example, we only sketch most proofs for space reasons. The full proofs can be found in [33].

2 RELATED WORK

There are two principal directions to tackle the problem of decidable first-order reasoning: restricting the language or restricting the inference mechanism. The earliest classes of syntactic fragments of first-order logic are prefix-vocabulary classes of formulas in prenex normal form whose symbols are taken from a certain vocabulary and whose leading quantifier prefix adheres to a specific form. Today, it is known which prefix-vocabulary classes are decidable and which are not [4]. Modern work on decidable fragments concerns bounded variable logics such as the two-variable fragment [37, 28], which connects first-order logic with propositional modal logic [8] and the prototypical description logic \mathcal{ALC} [29], which in turn has also been the basis for epistemic description logics [2].

Unlike these syntactic approaches, the direction we take here is based on restricting the inference mechanism. In other words, instead of exchanging expressivity for decidability as do syntactic fragments of first-order logic and description logics, we aim to trade off completeness against decidability. The standard tool to give semantics to knowledge and belief are Hintikka-style possible-worlds semantics [10]. Typically they imply omniscience [11], which brings along undecidability in the first-order case and intractability in the propositional case. Approaches to solve the omniscience problem include rather syntactic ones [15, 38, 6] as well as semantic approaches [22, 30, 16, 17, 5] based on tautological entailment [3, 1]. The former have the drawback of being very fine-grained and providing only little guidance as to which beliefs to include and which to leave out. The semantic approaches based on three- or four-valued semantics, by contrast, are much closer to the classical possible-worlds semantics, but miss out on many seemingly trivial inferences, such as simple unit propagation.

LLL's work on limited belief [26, 19, 20, 21] steers a middle course between these semantic and syntactic approaches. As we shall see, their semantics has a somewhat syntactic flavour, yet it is perspicuously defined and motivated. Klassen et al. [14] recently proposed a neighbourhood semantics that avoids the syntactic flavour; however, it is restricted to the propositional case. The closest relative of our approach among LLL's work is [20].

Although the omniscience problem and logics of limited belief have been around for over thirty years, the present paper is, as far as we know, the first to address the problem in the context of conditional belief. The underlying logic of conditional belief used as a reference for the limited logic developed in this paper is an extension of Levesque's logic of only-knowing [23, 24] to the case of conditional belief. Its semantics is defined in terms of plausibility-ranked

possible worlds.

Lewis [25] was the first to pick up the concept of possible worlds to semantically characterize conditionals; Grove [9] adopted this view and popularized it for belief change. The idea is to sequentially nest sets of possible worlds to obtain a so-called system of spheres. Intuitively, a system of spheres induces a plausibility ranking on the worlds: the most plausible worlds are contained in the innermost set, the second-most plausible worlds in the next set, and so on. Another popular and equivalent technique are total preorders [12].

Unlike material implications (but like counterfactuals), conditional belief is non-monotonic in the sense that strengthening the antecedent is not valid. The closest relative among the non-monotonic-reasoning approaches is Pearl's System Z [31]: it can be shown that conditional belief and only-believing subsume the non-monotonic 1-entailment in System Z [33].

As we saw already, evaluating a conditional in a system of spheres requires a consistency check. A sound consistency check, as necessitated for a sound semantics of conditionals, in turn requires a complete semantics. Relatively little work has been done in this area yet. Schaerf and Cadoli [32] address both unsound and incomplete inference, but they apply propositional techniques to restricted fragments of first-order logic. Recently, other methods of unsound reasoning have been investigated [27, 7]. However, as these approaches are not based primarily on subsumption and unit propagation, they do not fit well with LLL's techniques. We therefore propose a new complete semantics to complement LLL's sound inference.

3 CONDITIONAL BELIEF IN \mathcal{BO}

This section introduces the first-order logic of conditional belief \mathcal{BO} [34]. It features two modal operators to represent beliefs, whose semantics is defined through a system of spheres, that is, a sequence of nested sets of possible worlds.

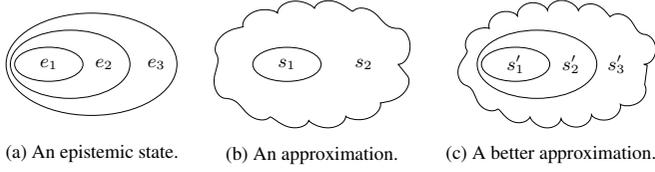
The language \mathcal{BO} is defined as follows. The set of *terms* is the least set that contains all first-order variables x and all (*standard*) *names* $n \in \mathcal{N} = \{\#1, \#2, \dots\}$. The set of *formulas* is the least set of expressions such that

- $P(t_1, \dots, t_j), (t_1 = t_2), (\alpha \vee \beta), \neg\alpha, \exists x\alpha,$
- $\mathbf{B}(\phi_1 \Rightarrow \psi_1), \mathbf{O}\{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$

are formulas, where P is a predicate symbol other than $=$, t_i are terms, α and β are formulas, and ϕ_i and ψ_i are *objective* formulas, that is, ϕ_i and ψ_i mention no \mathbf{B} or \mathbf{O} operators. $P(t_1, \dots, t_j)$ and $(t_1 = t_2)$ are called (*non*-)equality atoms, respectively. A *literal* is an atom a or its negation $\neg a$. The complement of a literal ℓ is written $\bar{\ell}$. A formula is *ground* when it contains no variables. A *sentence* is a formula without free variables. We let $\wedge, \vee, \supset, \equiv$ be the usual abbreviations, \top stand for $\exists x(x = x)$, and \perp abbreviate $\neg\top$.

Standard names are special constants that represent all the individuals in the universe. They allow for a simpler semantics than the classical Tarskian model theory; in particular, quantification can be handled by simply substituting standard names. The formula $\mathbf{B}(\phi \Rightarrow \psi)$ intuitively expresses a conditional belief, namely the belief that if ϕ is true, then presumably ψ is also true. The formula $\mathbf{O}\{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$ goes one step further and says that the conditional beliefs $\mathbf{B}(\phi_i \Rightarrow \psi_i)$ are *all* that is believed; everything that is not a consequence of these conditional beliefs is implicitly not believed. The concept is called *only-believing*; it generalizes Levesque's only-knowing [23, 24] to the case of conditional beliefs.

Before we give a semantics to this language, let us see how Example 1 can be expressed in \mathcal{BO} .



(a) An epistemic state. (b) An approximation. (c) A better approximation.

Figure 1. An epistemic state and two approximations. Each ellipse represents the scenarios considered possible in that sphere.

Example 2 Let A , I , V represent that the guest is Australian, Italian, a vegetarian, respectively; $E(x)$ that x is among her preferred diet; $M(x)$ that x is meat; and roo be a standard name representing kangaroo meat. Then all we believe is

- $A \Rightarrow \neg I$ and $I \Rightarrow \neg A$;
 - $A \Rightarrow E(\text{roo})$;
 - $\top \Rightarrow I \vee V$ and $\neg I \Rightarrow A$;
 - $\neg M(\text{roo}) \Rightarrow \perp$ and $\neg \forall x (V \wedge M(x) \supset \neg E(x)) \Rightarrow \perp$.
- } = Γ

($\neg \phi \Rightarrow \perp$ expresses indefeasible knowledge of ϕ .) The question “if she is not Italian, is she presumably not a vegetarian?” then boils down to whether $\text{O}\Gamma$ entails $\mathbf{B}(\neg I \Rightarrow \neg V)$.

To investigate such problems, we define a possible-worlds semantics for \mathcal{BC} .

Definition 3 A *world* is a set of ground non-equality atoms. An *epistemic state* \vec{e} is an infinite sequence of sets of worlds e_1, e_2, \dots such that $e_1 \subseteq e_2 \subseteq \dots$ and for some $p \in \{1, 2, \dots\}$, $e_p = e_{p+1} = \dots$

Intuitively, a world is a truth assignment of the ground non-equality atoms, that is, of all $P(n_1, \dots, n_j)$ where the n_i are standard names. An epistemic state stratifies sets of such worlds with the subset relation. The intuition behind an epistemic state \vec{e} is to model a system of spheres as discussed in the introduction: e_1 contains the most-plausible worlds, e_2 adds the second-most-plausible worlds, and so on. Note that in any epistemic state \vec{e} only finitely many different sets of worlds are allowed, since the definition requires that $e_p = e_{p+1} = \dots$ for some p . (Just as well we could have defined an epistemic state to be a non-empty finite sequence of supersets; the present definition however is often easier to work with.)

Given an epistemic state \vec{e} and a world w , we can define truth of a sentence α in \mathcal{BC} , written $\vec{e}, w \models \alpha$. We let α_n^x denote the result of substituting all free occurrences of x by n . The objective part of the semantics is defined inductively as follows:

1. $\vec{e}, w \models P(n_1, \dots, n_j)$ iff $P(n_1, \dots, n_j) \in w$;
2. $\vec{e}, w \models (n_1 = n_2)$ iff n_1 and n_2 are identical names;
3. $\vec{e}, w \models (\alpha \vee \beta)$ iff $\vec{e}, w \models \alpha$ or $\vec{e}, w \models \beta$;
4. $\vec{e}, w \models \neg \alpha$ iff $\vec{e}, w \not\models \alpha$;
5. $\vec{e}, w \models \exists x \alpha$ iff $\vec{e}, w \models \alpha_n^x$ for some $n \in \mathcal{N}$.

To define the semantics of belief, it is convenient to define the *plausibility* $[\vec{e} | \phi]$ of an objective sentence ϕ in \vec{e} as the index of the first sphere consistent with ϕ :

$$[\vec{e} | \phi] = \min\{p \mid p = \infty \text{ or } \vec{e}, w \models \phi \text{ for some } w \in e_p\},$$

where $\infty \notin \{1, 2, \dots\}$ represents an “undefined” plausibility with the understanding that $p + \infty = \infty$ and $p < \infty$ for all $p \in \{1, 2, \dots\}$. Then the semantics of beliefs is as follows:

6. $\vec{e}, w \models \mathbf{B}(\phi \Rightarrow \psi)$ iff for all $p \in \{1, 2, \dots\}$, if $p \leq [\vec{e} | \phi]$ and $w' \in e_p$, then $\vec{e}, w' \models (\phi \supset \psi)$;
7. $\vec{e}, w \models \mathbf{O}\{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$ iff for all $p \in \{1, 2, \dots\}$, $w' \in e_p$ iff $\vec{e}, w' \models \bigwedge_{i: [\vec{e} | \phi_i] \geq p} (\phi_i \supset \psi_i)$.

Rules 1–5 are straightforward. Notable is perhaps that quantification can be handled substitutionally thanks to standard names. The most interesting rules are the ones for belief, of course. According to Rule 6, $\mathbf{B}(\phi \Rightarrow \psi)$ holds iff $\phi \supset \psi$ is true in the innermost sphere consistent with ϕ . Only-believing $\mathbf{O}\{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$ has the effect of $\mathbf{B}(\phi_i \Rightarrow \psi_i)$ plus maximizing every sphere: it requires the sphere e_p to contain *all* worlds that satisfy all $(\phi_i \supset \psi_i)$ for which $[\vec{e} | \phi_i] \geq p$.

When $\vec{e}, w \models \alpha$ for all \vec{e} (or w), we allow ourselves to omit \vec{e} (or w , respectively). As usual, we use the symbol \models also to denote entailment. In particular, $\text{O}\Gamma \models \mathbf{B}(\phi \Rightarrow \psi)$ is to say that for all \vec{e} , $\vec{e} \models \text{O}\Gamma$, then $\vec{e} \models \mathbf{B}(\phi \Rightarrow \psi)$.

The fundamental property of only-believing is that it uniquely determines the epistemic state.

Theorem 4 *There is a unique epistemic state \vec{e} such that $\vec{e} \models \mathbf{O}\{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$.*

In fact, \vec{e} can be generated inductively: when spheres e_1, \dots, e_{p-1} are determined, we can decide whether $[\vec{e} | \phi_i] \geq p$ for every i , and thus determine the next sphere e_p using the right-hand side of Rule 7. A proof of the theorem and this construction can be found in [34]; here, we illustrate the process by continuing Example 2 instead.

Example 5 The first sphere e_1 of \vec{e} such that $\vec{e} \models \text{O}\Gamma$ contains all worlds that satisfy all materialized conditionals from Γ :

$$e_1 = \{w \mid w \models (\neg A \vee \neg I) \wedge (\neg A \vee E(\text{roo})) \wedge (I \vee V) \wedge (I \vee A) \wedge \mu\}$$

where $\mu = M(\text{roo}) \wedge \forall x (V \wedge M(x) \supset \neg E(x))$ represents our knowledge about meat and vegetarians.

For the next sphere, we need to figure out the plausibilities $[\vec{e} | \phi]$ for the conditionals $\phi \Rightarrow \psi \in \Gamma$. To begin with, we need to answer if $[\vec{e} | A] \geq 2$, that is, if e_1 is inconsistent with A . To this end, we can split on V : from V we obtain $\neg E(\text{roo})$ (by μ) and thus $\neg A$; on the other hand, from $\neg V$ we infer I and thus $\neg A$; so indeed e_1 is inconsistent with A , that is, $[\vec{e} | A] \geq 2$. By the same argument, $[\vec{e} | \neg I] \geq 2$. It is moreover easy to see that e_1 is consistent and thus $[\vec{e} | I] = [\vec{e} | \top] = 1$. Hence the conditionals $A \Rightarrow \neg I$, $A \Rightarrow E(\text{roo})$, $\neg I \Rightarrow A$, plus the knowledge about meat and vegetarians determine the second sphere:

$$e_2 = \{w \mid w \models (\neg A \vee \neg I) \wedge (\neg A \vee E(\text{roo})) \wedge (I \vee A) \wedge \mu\}.$$

Again we need to check which premises are consistent with e_2 , and only the remaining conditionals determine the next sphere e_3 . It is easy to see that $[\vec{e} | A] = [\vec{e} | \neg I] = 2$, so for the third and last sphere:

$$e_3 = \{w \mid w \models \mu\}.$$

Since \vec{e} is the unique model of $\text{O}\Gamma$ in \mathcal{BC} , it determines our beliefs. For example, $\text{O}\Gamma \models \mathbf{B}(\neg I \Rightarrow \neg V)$ since $[\vec{e} | \neg I] = 2$ and $w \models \neg I \supset \neg V$ for all $w \in e_2$.

Now, how could a limited, decidable version of \mathcal{BC} look like? As for $\mathbf{B}(\phi \Rightarrow \psi)$, we sketched the idea in the introduction already: approximate the plausibility of ϕ from above, and then use sound inference to check whether that sphere satisfies $\phi \supset \psi$. Even if the

\models	satisfaction and entailment in \mathcal{BO}
\approx	satisfaction in sound first-order semantics
\approx	satisfaction in complete first-order semantics
\approx	satisfaction and entailment in \mathcal{BOC}

Table 1. The zoo of turnstile symbols used in the paper.

plausibility approximation leads to a too-far-out sphere, this is sound because what can be inferred from an outer sphere can also be inferred from any inner sphere. But another problem is how to approximate the model of $\mathbf{O}\{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$. Example 5 shows that determining this epistemic state \vec{e} is not trivial, as reasoning is necessary to figure out which beliefs are more plausible than others, that is, which plausibilities are $\geq p$ to determine the p th sphere. Our approximation of \vec{e} will be based on a lower and an upper bound of the plausibilities of ϕ_i . As long as both bounds agree for every i on whether the plausibility of ϕ_i is $\geq p$, we can faithfully represent the p th sphere in the approximation. Once the bounds are inconsistent, though, it is not clear which conditionals shall determine the p th sphere of the approximation, and hence we skip to the final sphere, which represents at least those scenarios contained in the last sphere of \vec{e} . Two such approximated epistemic states are depicted in Figure 1: in Figure 1b the bounds are inconsistent already for the second sphere; Figure 1c faithfully represents the first two spheres, but is pessimistic about the outermost ones, that is, considers too many scenarios. It is important that the last sphere of the approximation must not be optimistic, for otherwise it might satisfy formulas that the last sphere of \vec{e} does not.

4 LIMITED OBJECTIVE REASONING

Here we introduce a sound but incomplete and another complete but unsound semantics for objective formulas. As argued before, they will lay the groundwork for \mathcal{BOC} , the limited version of \mathcal{BO} . Before we elaborate on what is meant by soundness and completeness, we define some fundamentals.

Definition 6 A *clause* is a set of literals $[\ell_1, \dots, \ell_k]$ (we use square brackets to ease readability). The empty clause is written as $[\]$. Every non-empty clause corresponds to the disjunction $(\ell_1 \vee \dots \vee \ell_k)$ (with arbitrary brackets and order). A *setup* is a set of ground clauses.

As in [26, 19, 20, 21], setups are the primitives of our limited semantics. Just like a set of possible worlds, a setup represents possibly incomplete or disjunctive information. The semantics \approx and \approx are sound and complete, respectively, in the following sense:

- whenever a setup s satisfies ϕ in the *sound* semantics \approx , s classically entails ϕ , that is, every world that satisfies all $c \in s$ also satisfies ϕ in unlimited first-order logic;
- whenever a setup s classically entails ϕ , s satisfies ϕ in the *complete* semantics \approx .

Inference in both semantics is based on unit propagation and subsumption. For example, if $[A, I]$ and $[\neg I]$ are in the setup, then $[A]$ is inferred by unit propagation, and $[A, V]$ follows by subsumption.

Definition 7 For a setup s , we write s^- to remove all subsumed clauses, s^+ to add all subsumed clauses, and $\text{UP}(s)$ to close s to-

gether with all valid equality literals under unit propagation:

$$\begin{aligned}
s^- &= \{c \in s \mid \text{for all } c' \subsetneq c, c' \notin s\}; \\
s^+ &= \{c \mid \text{for some } c' \subseteq c, c' \in s\}; \\
\text{EQ} &= \{[(n = n)], [(n \neq n')]\mid \text{distinct } n, n' \in \mathcal{N}\}; \\
\text{UP}(s) &= \text{closure of } \text{EQ} \cup s \text{ under unit propagation.}
\end{aligned}$$

We also write $\text{UP}^-(s)$ for $\text{UP}(s)^-$, and $\text{UP}^+(s)$ for $\text{UP}(s)^+$.

The following lemma states the equivalence of $s, s^-, s^+, \text{UP}(s)$.

Lemma 8 For any world w and setup s ,

- $w \models c$ for all $c \in s$ iff
- $w \models c$ for all $c \in s^-$ iff
- $w \models c$ for all $c \in s^+$ iff
- $w \models c$ for all $c \in \text{UP}(s)$.

Proof. Showing the equivalence of (i)–(iii) is straightforward. Here we only show that (i) iff (iv). The if direction is immediate. For the converse, suppose (i) holds and let $c \in \text{UP}(s)$. The proof is by induction on the length of the derivation of c . For the base case let $c \in \text{EQ} \cup s$. If $c \in \text{EQ}$, then clearly $w \models c$. If $c \in s$, then $w \models c$ by assumption. For the induction step suppose $c \in \text{UP}(s)$ is the resolvent of $c \cup [\ell], [\bar{\ell}] \in \text{UP}(s)$. By induction, $w \models c \vee \ell$ and $w \models \bar{\ell}$. Thus $w \models c$. \square

4.1 Sound but incomplete semantics

We are now ready to define the first satisfaction relation: $s, k \approx \phi$ for a setup $s, k \in \{0, 1, 2, \dots\}$, and an objective sentence ϕ . Intuitively, k indicates how much *effort* is put into proving that ϕ is true in s . The idea is that at level $k = 0$, only obvious inferences can be made, that is, only what is subsumed by the setup s comes out true. At every higher split level $k > 0$, the reasoner may pick a literal ℓ and consider the cases where the unit clauses $[\ell]$ and $[\bar{\ell}]$ are added to the setup. When ℓ is chosen smartly, this may set off unit propagation and lead to new inferences.

We define the sound semantics as follows:

- $s, k + 1 \approx \phi$ iff $s \cup \{[\ell]\}, k \approx \phi$ and $s \cup \{[\bar{\ell}]\}, k \approx \phi$ for some ground literal ℓ ;
- if c is a clause:
 $s, 0 \approx c$ iff $c \in \text{UP}^+(s)$;
- if $(\phi \vee \psi)$ is not a clause:
 $s, 0 \approx (\phi \vee \psi)$ iff $s, 0 \approx \phi$ or $s, 0 \approx \psi$;
- $s, 0 \approx \neg(\phi \vee \psi)$ iff $s, 0 \approx \neg\phi$ and $s, 0 \approx \neg\psi$;
- $s, 0 \approx \neg\neg\phi$ iff $s, 0 \approx \phi$;
- $s, 0 \approx \exists x \phi$ iff $s, 0 \approx \phi_n^x$ for some $n \in \mathcal{N}$;
- $s, 0 \approx \neg\exists x \phi$ iff $s, 0 \approx \neg\phi_n^x$ for all $n \in \mathcal{N}$.

Let us first illustrate how the definition works by way of our running example.

Example 9 Let $s_\mu = \{[M(\text{roo})], [\neg M(n), \neg E(n), \neg V] \mid n \in \mathcal{N}\}$ and $s_1 = \{[\neg A, \neg I], [\neg A, E(\text{roo})], [I, V], [I, A]\} \cup s_\mu$. This setup corresponds to the first sphere e_1 from Example 5. There we argued that it is inconsistent with A . To obtain the same result in this limited semantics, that is, $s_1, k \approx \neg A$, one split is needed, that is, $k \geq 1$: clearly, $[\neg A] \notin \text{UP}^+(s_1)$; but adding $[V]$ to s_1 triggers unit propagation that first yields $[\neg M(\text{roo}), \neg E(\text{roo})]$, then $[\neg E(\text{roo})]$, and

then $[\neg A]$; on the other hand, adding $[\neg V]$ yields $[I]$ and then again $[\neg A]$. Hence, $s_1, k \models \neg A$ iff $k \geq 1$. Analogously we can argue that $s_1, k \models I$ iff $k \geq 1$.

The following theorem establishes the aforementioned soundness of \models wrt classical logic. We write $s \models \phi$ to say that $w \models \phi$ for all w with $w \models c$ for all $c \in s$.

Theorem 10 *If $s, k \models \phi$, then $s \models \phi$.*

Proof. By induction on k . We show the base case $k = 0$ by subinduction on $|\phi|$. For any clause, $s, 0 \models c$ iff $c \in \text{UP}^+(s)$ only if $\text{UP}^+(s) \models c$ iff (by Lemma 8) $s \models c$. The other subinduction cases are trivial; for example, for an existential, $s, 0 \models \exists x \phi$ iff $s, 0 \models \phi_n^x$ for some $n \in \mathcal{N}$ only if (by subinduction) $s \models \phi_n^x$ for some $n \in \mathcal{N}$ only if $s \models \exists x \phi$.

For the main induction step suppose the lemma holds for k and that $s, k + 1 \models \phi$. Suppose $w \models c$ for all $c \in s$. By the Rule 1, $s \cup \{[\ell]\}, k \models \phi$ and $s \cup \{[\bar{\ell}]\}, k \models \phi$ for some ℓ . By induction, $s \cup \{[\ell]\} \models \phi$ and $s \cup \{[\bar{\ell}]\} \models \phi$. Therefore, since either $w \models \ell$ or $w \models \bar{\ell}$, we have $w \models \phi$. Hence $s \models \phi$. \square

Another interesting property is the following so-called eventual completeness for propositional formulas.

Theorem 11 *Let s be finite and ϕ be propositional. Then $s, k \models \phi$ for some $k \in \{0, 1, 2, \dots\}$ iff $s \models \phi$.*

Proof sketch. The only-if direction follows from Theorem 10. Conversely, let k be at least the number of atoms in s and ϕ . Then we can split them all, which corresponds to testing all truth assignments for these atoms. \square

We remark that \models is a slightly restricted version of the semantics in [20] to ease the presentation. The main cost of our simplification is that we lose eventual completeness for formulas $\forall \vec{x} \phi$ where ϕ is quantifier-free.

4.2 Complete but unsound semantics

Next, we turn to the complete but unsound semantics \models . In the complete semantics, it is often more intuitive to consider the task of *disproving* that s satisfies ϕ , that is, $s, l \not\models \phi$ where $l \in \{0, 1, 2, \dots\}$. Here l specifies the reasoning effort similar to the split levels k before. In $s, k \models \phi$ one (roughly) shows that for some atoms, ϕ obviously comes out true in s under any truth assignment of these atoms (where “obvious” means after unit propagation and subsumption). By contrast, the objective for $s, l \not\models \phi$ is to show that s can be augmented with l literals so the resulting setup obviously disproves ϕ .

In particular, this requires to detect whether the setup *might* be inconsistent, because only a consistent setup can disprove ϕ . For that, we use a very simple heuristic: whenever the setup mentions some literal both positively and negatively after removing all subsumed clauses, it is deemed possibly-inconsistent. While this heuristic is of course not sophisticated, the idea is to compensate for its naivete by increasing l , that is, by more reasoning effort.

Definition 12 We write $\text{XP}(s)$ to close the set of all literals that occur in $\text{UP}^-(s)$ under unit propagation. $\text{gnd}(c)$ for the set of ground instances of c , and $s \otimes \ell$ to augment s with all ground instances of $[\ell]$ which are not obviously inconsistent with s :

$$\begin{aligned} \text{XP}(s) &= \text{UP}(\{[\ell] \mid \ell \in c \text{ for some } c \in \text{UP}^-(s)\}); \\ \text{gnd}(c) &= \{c_n^{\vec{x}} \mid \vec{x} \text{ are the free variables of } c, n_i \in \mathcal{N}\}; \\ s \otimes \ell &= s \cup \{[\ell_n^{\vec{x}}] \in \text{gnd}([\ell]) \mid [\bar{\ell}_n^{\vec{x}}] \notin \text{UP}^+(s)\}. \end{aligned}$$

The rationale behind $s \otimes \ell$ is that often a setup may contain infinitely many instances of some clause, and we want to trigger unit propagation for all of them. For example, when $s = \{[\neg P(\#1)], [\neg P(n), Q(n)] \mid n \in \mathcal{N}\}$, then $s \otimes P(x)$ augments s with the instances $P(n)$ for all $n \neq \#1$. With unit propagation we can then infer $Q(n)$ for all $n \neq \#1$, but we avoid the empty clause.

$\text{XP}(s)$ simply serves our simple heuristic to check whether a setup might be inconsistent: it takes all literals from $\text{UP}^-(s)$ and closes them under unit resolution. The next lemma is therefore no surprise.

Lemma 13 *If $[\] \notin \text{XP}(s)$, then for some w , for all $c \in s$, $w \models c$.*

Proof. Let $[\] \notin \text{XP}(s)$ and let $w \models \ell$ iff $[\ell] \in \text{XP}(s)$, which exists as $[\ell] \notin \text{XP}(s)$ or $[\bar{\ell}] \notin \text{XP}(s)$. By subsumption, $w \models c$ for all $c \in \text{UP}^-(s)$. By Lemma 8, $w \models c$ for all $c \in s$. \square

For a setup s , effort $l \in \{0, 1, 2, \dots\}$, and an objective sentence ϕ , the complete satisfaction relation $s, l \models \phi$ is defined inductively:

1. $s, l + 1 \models \phi$ iff $s \otimes \ell, l \models \phi$ for all literals ℓ ;
2. if c is a clause:
 $s, 0 \models \neg c$ iff $[\] \in \text{XP}(s)$ or $c \notin \text{UP}^+(s)$;
3. $s, 0 \models (\phi \vee \psi)$ iff $s, 0 \models \phi$ or $s, 0 \models \psi$;
4. if $(\phi \vee \psi)$ is not a clause:
 $s, 0 \models \neg(\phi \vee \psi)$ iff $s, 0 \models \neg\phi$ and $s, 0 \models \neg\psi$;
5. $s, 0 \models \neg\neg\phi$ iff $s, 0 \models \phi$;
6. $s, 0 \models \exists x \phi$ iff $s, 0 \models \phi_n^x$ for some $n \in \mathcal{N}$;
7. $s, 0 \models \neg\exists x \phi$ iff $s, 0 \models \neg\phi_n^x$ for all $n \in \mathcal{N}$.

The main differences between \models and \models are Rules 1 and 2. It may be more intuitive to read the definition of \models from the perspective disproving. According to Rule 1 $s, l + 1 \not\models \phi$ means that we may pick some literal ℓ and show $s \otimes \ell, l \not\models \phi$. And Rule 2 says that $s, 0 \not\models \neg c$ when the setup is certainly consistent (since $[\] \notin \text{XP}(s)$) but satisfies c (since $c \in \text{UP}(s)$). We illustrate this with our example.

Example 14 Consider s_1 from Example 9, and let us see whether it is consistent with I , that is, $s_1, l \models \neg I$ for certain l . For $l = 0$, note that $\text{UP}^-(s_1)$ mentions I and $\neg I$ in clauses, so $[\] \in \text{XP}(s_1)$, and thus $s_1, 0 \models \neg I$. For $l \geq 1$, however, we are allowed to add some literal to s_1 so as to build a countermodel that clearly disproves $\neg I$. Indeed, adding, for example, $[\neg A]$ does the job: $\text{UP}^-(s_1 \otimes \neg A) = \{[\neg A], [I]\} \cup \{[M(\text{roo})], [\neg E(\text{roo}), \neg V], [\neg M(n), \neg E(n), \neg V] \mid n \in \mathcal{N} \setminus \{\text{roo}\}\} \cup \text{EQ}$ is obviously consistent, that is, $[\] \notin \text{XP}(s_1)$, and moreover $[I] \in \text{UP}^+(s_1 \otimes \neg A)$. So by adding $\neg A$, we have shown that s_1 can falsify $\neg I$. Thus, $s_1, l \not\models \neg I$ iff $l \geq 1$.

The following theorem is the completeness result for \models .

Theorem 15 *If $s \models \phi$, then $s, l \models \phi$.*

Proof. By contraposition and by induction on l . For the base case let $l = 0$ and suppose $s, 0 \not\models \phi$. Then clearly $[\] \notin \text{XP}(s)$, so by Lemma 13, there is a w such that $w \models c$ for all $c \in s$ (*). We show that $w \not\models \phi$ by subinduction on $|\phi|$. For any negated clause, $s, 0 \not\models \neg c$ iff $[\] \notin \text{XP}(s)$ and $c \in \text{UP}^+(s)$ only if (by (*) and Lemma 8) $w \models c$ iff $w \not\models \neg c$. Very similarly, for any literal, $s, 0 \not\models \ell$ iff $s, 0 \not\models \neg \bar{\ell}$ only if (by the same argument as for negated clauses) $w \not\models \bar{\ell}$ iff $w \not\models \ell$. We omit the other cases; they are straightforward.

For the main induction step suppose the lemma holds for l and that $s, l + 1 \not\models \phi$. Then $s \otimes \ell, l \not\models \phi$ for some ℓ . By induction, $s \otimes \ell \not\models \phi$. By monotonicity, $s \not\models \phi$. \square

Just like \models is eventually complete for the propositional case, \models is eventually sound, that is, all invalid inferences can be detected for large enough l .

Theorem 16 *Let s be finite and ϕ be propositional. Then $s, l \not\models \phi$ for some $l \in \{0, 1, 2, \dots\}$ iff $s \not\models \phi$.*

Proof sketch. The only-if direction follows from Theorem 15. Conversely, when l is at least the number of atoms in s and ϕ , they can all be set to the same value as in a world that satisfies s but not ϕ . \square

5 LIMITED CONDITIONAL BELIEF IN \mathcal{BOC}

We are finally ready for \mathcal{BOC} , the logic of limited conditional belief. The language is the same as for \mathcal{BO} , except that the belief operators \mathbf{B}_k^l and \mathbf{O}_k^l are now decorated with $k, l \in \{0, 1, 2, \dots\}$ to indicate the reasoning effort, and for simplicity we disallow predicates outside of belief modalities.

Definition 17 A *limited epistemic state* \vec{s} is an infinite sequence of setups s_1, s_2, \dots such that $\text{UP}^+(s_1) \supseteq \text{UP}^+(s_2) \supseteq \dots$ and for some $p \in \{1, 2, \dots\}$, $\text{UP}^+(s_p) = \text{UP}^+(s_{p+1}) = \dots$

The idea behind limited epistemic states is the same as for unlimited epistemic states, except that in the limited case every sphere is represented as a setup instead of a set of worlds.

Recall that the plausibility of a formula is the index of the first sphere consistent with that formula. With the limited satisfaction relations from the previous section, we can approximate this notion of plausibility from below and above:

$$\begin{aligned} [\vec{s}, k \dot{\phi}] &= \min\{p \mid p = \infty \text{ or } s_p, k \not\models \neg\phi\}; \\ [\vec{s}, l \phi] &= \min\{p \mid p = \infty \text{ or } s_p, l \not\models \neg\phi\}. \end{aligned}$$

It is easy to see that increasing the effort does not impair the quality of these the approximations, as stated in the next lemma.

Lemma 18 $[\vec{s}, k \dot{\phi}] \leq [\vec{s}, k + 1 \dot{\phi}] \leq [\vec{s}, l + 1 \phi] \leq [\vec{s}, l \phi]$.

Proof. For the first inequality, it is easy to see that $s_p, k + 1 \not\models \neg\phi$ implies $s_p, k \not\models \neg\phi$. Similarly for the third inequality, $s_p, l \not\models \neg\phi$ implies $s_p, l + 1 \not\models \neg\phi$. For the remaining one, if $s_p, l + 1 \not\models \neg\phi$, then $s_p \not\models \neg\phi$ by Theorem 15, and so $s, k + 1 \not\models \neg\phi$ by Theorem 10. \square

These approximations are key to the semantics of \mathcal{BOC} . Recall that in \mathcal{BO} the semantics of conditional belief is that the plausibility of the antecedent denotes the sphere in which the material implication of antecedent and consequent should be evaluated. And for only-believing in \mathcal{BO} the p th sphere of the epistemic state is determined by those conditionals whose antecedent have a plausibility $\geq p$.

In limited reasoning, we only have the approximate plausibilities. For limited conditional belief $\mathbf{B}_k^l(\phi \Rightarrow \psi)$ the idea is therefore to approximate the plausibility of the ϕ from above. And for limited only-believing $\mathbf{O}_k^l\{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$ we shall build up the corresponding limited epistemic state only as long as the approximations from below and above are consistent: we say a limited epistemic state \vec{s} is *l_k -plausibility-consistent at $p \in \{1, 2, \dots\}$* iff for all $i \in \{1, \dots, m\}$, $[\vec{s}, k \dot{\phi}_i] \geq p$ iff $[\vec{s}, l \phi_i] \geq p$.

Moreover, in analogy to how only-believing in \mathcal{BO} maximizes every set of world of the epistemic state, we here need to minimize the setups in order to maximize the agent's non-beliefs. We say a setup s is *minimal* wrt $s, k \not\models \phi$ iff $s, k \not\models \phi$ and there is no s' such that $\text{UP}^+(s') \subsetneq \text{UP}^+(s)$ and $s', k \not\models \phi$. Finally, we let $\text{NF}[v]$ stand for the prenex negation normal form of v .

Truth of a sentence α in a limited epistemic state \vec{s} , written $\vec{s} \models \alpha$, is now defined inductively:

1. $\vec{s} \models (\alpha \vee \beta)$ iff $\vec{s} \models \alpha$ or $\vec{s} \models \beta$;
2. $\vec{s} \models \neg\alpha$ iff $\vec{s} \not\models \alpha$;

3. $\vec{s} \models \exists x \alpha$ iff $\vec{s} \models \alpha_n^x$ for some name n ;
4. $\vec{s} \models \mathbf{B}_k^l(\phi \Rightarrow \psi)$ iff for all $p \in \{1, 2, \dots\}$, if $p \leq [\vec{s}, l \dot{\phi}]$, then $s_p, k \not\models (\phi \supset \psi)$;
5. $\vec{s} \models \mathbf{O}_k^l\{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$ iff for some limited epistemic state \vec{s}' , and for all $p \in \{1, 2, \dots\}$,
 - s'_p is minimal wrt $s'_p, 0 \not\models \bigwedge_{i: [\vec{s}', k \dot{\phi}_i] \geq p} \text{NF}[(\phi_i \supset \psi_i)]$;
 - $s_p = \begin{cases} s'_p & \text{if } \vec{s}' \text{ is } l_k\text{-plausibility-consistent at } 1, \dots, p; \\ s'_{p^\circ} & \text{otherwise;} \end{cases}$
 - where p° is such that $\text{UP}^+(s_{p^\circ}) = \text{UP}^+(s_{p'})$ for all $p' \geq p^\circ$.

As usual and as in \mathcal{BO} , the symbol \models is also used to denote entailment.

Rule 4 approximates the plausibility of ϕ from above, which avoids too-plausible spheres inconsistent with ϕ , and then applies sound inference. That way, \mathbf{B}_k^l is a conservative variant of \mathcal{BO} 's conditional belief operator.

The same spirit is behind Rule 5. The intuition is to build up the system of spheres as long as the lower and upper bound of all plausibilities are consistent. Once they are not, it is unclear how the next sphere should look like, so we skip to the "last" one, s'_{p° . That last sphere is determined by conditionals which (mutually) contradict their premises, so there is no scenario where any of them could be true. The parameters k and l determine how much effort is put into checking the plausibility-consistency. Note that there may be conditionals $\phi_i \Rightarrow \psi_i$ with unsatisfiable antecedents which do not occur in the last sphere. This is because we only take those conditionals whose antecedents can be proved unsatisfiable by sound reasoning (with effort k). If we used complete instead of sound reasoning here, the outermost sphere could end up being too strong and then yield false beliefs. Figure 1 illustrates such approximations.

Before we illustrate how the semantics works, we show a unique-model property for a certain type of knowledge base.

5.1 Proper⁺ knowledge bases

The class of knowledge bases we are chiefly interested in is called proper⁺ [18]. Essentially, it requires clausal form and disallows existential quantifiers.

Definition 19 A sentence π is *proper⁺* when π is of the form $\bigwedge_j \forall \vec{x} c_j$ for clauses c_j . Then we let $\text{gnd}(\pi) = \bigcup_j \text{gnd}(c_j)$. A set of conditionals $\Gamma = \{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$ is proper⁺ when $\bigwedge_{1 \leq i \leq m} \text{NF}[(\phi_i \supset \psi_i)]$ is proper⁺.

Bringing $(\phi_i \supset \psi_i)$ into prenex negation normal form has the benefit many conditionals are proper⁺ which otherwise wouldn't. For example, $(P \wedge Q \supset R)$, which is just an abbreviation for $\neg\neg(\neg P \vee \neg Q) \vee R$ is not proper⁺, but eliminating the double negation does the job already. Incidentally, this is also the reason why NF occurs in Rule 5.

For the remainder of this paper we let π and Γ be proper⁺. Proper⁺ knowledge bases have been shown to have attractive properties for limited belief [26, 19, 20, 21], and as we shall see these qualities also hold for conditional belief. Above all, the unique-model property from Theorem 4 carries over to limited belief.

Theorem 20 *There is a unique (modulo UP^+) limited epistemic state \vec{s} such that $\vec{s} \models \mathbf{O}_k^l \Gamma$, that is, for all \vec{s}' such that $\vec{s}' \models \mathbf{O}_k^l \Gamma$ and for all $p \in \{1, 2, \dots\}$, $\text{UP}^+(s_p) = \text{UP}^+(s'_p)$.*

Proof sketch. The crucial lemma is that for every proper⁺ π , s is minimal wrt $s, 0 \not\models \pi$ iff $\text{UP}^+(s) = \text{UP}^+(\text{gnd}(\pi))$, proven in [19].

With that result, the theorem can be shown by the same argument used to prove the unique-model property in \mathcal{BO} [34]. \square

Finally, here is the kangaroo example with limited belief.

Example 21 Note that Γ from Example 2 is proper⁺. Let $k = 1$ and $l = 1$, and let s_1 and s_μ be as in Example 9. Then s_1 is the first sphere of $\vec{s} \approx \mathbf{O}_k^l \Gamma$. To determine the next sphere, we first need to see whether \vec{s} is l -plausibility-consistent at 2, that is, $[\vec{s}, k \dot{\phi}] \geq 2$ iff $[\vec{s}, l \dot{\phi}] \geq 2$ for all $\phi \Rightarrow \psi \in \Gamma$. We can reuse our results from Examples 9 and 14. For example, we have shown in Example 14 that $s_1, l \not\approx \neg I$, so we have $[\vec{s}, l \dot{I}] = 1$. Similarly, in Example 9 we have shown $s_1, k \not\approx \neg A$, so $[\vec{s}, k \dot{A}] \geq 2$. That way and with Lemma 18, we obtain

- $[\vec{s}, k \dot{I}] = 1$ and $[\vec{s}, l \dot{I}] = 1$;
- $[\vec{s}, k \dot{A}] \geq 2$ and $[\vec{s}, l \dot{A}] \geq 2$;
- $[\vec{s}, k \dot{\top}] = 1$ and $[\vec{s}, l \dot{\top}] = 1$;
- $[\vec{s}, k \dot{\neg I}] \geq 2$ and $[\vec{s}, l \dot{\neg I}] \geq 2$.

The plausibilities of the last two conditionals in Example 2 are omitted, as they are vacuously ∞ . Hence, \vec{s} is l -plausibility-consistent at 2. The conditionals with plausibility ≥ 2 determine the second sphere, so we obtain

$$\text{UP}^+(s_2) = \text{UP}^+(\{[\neg A, \neg I], [\neg A, E(\text{roo})], [I, A]\} \cup s_\mu).$$

It is easy to see that $[\vec{s}, k \dot{A}] = [\vec{s}, k \dot{\neg I}] = 2$. Moreover $[\vec{s}, l \dot{A}] = [\vec{s}, l \dot{\neg I}] = 2$ can be shown by adding A to the setup. So for the final sphere s_3 we have

$$\text{UP}^+(s_3) = \text{UP}^+(s_\mu).$$

By Theorem 20, \vec{s} is the unique model of $\mathbf{O}_k^l \Gamma$, so we can now prove $\mathbf{O}_k^l \Gamma \approx \mathbf{B}_{k'}^{l'}(\neg I \Rightarrow \neg V)$ for $k' = 1, l' = 1$: since $[\vec{s}, l' \dot{\phi}] = 2$, we only need to show $s_2, k' \not\approx I \vee \neg V$, which is easy by splitting I .

Note that for $k = 0$ or $l = 0$, the model of $\mathbf{O}_k^l \Gamma$ would have consisted of s_1 followed immediately by s_μ , because of l -plausibility-inconsistency at 2. In this case, no k' or l' would have been large enough to show $\mathbf{B}_{k'}^{l'}(\phi \Rightarrow \psi)$.

In this example, we let $k = l = k' = l' = 1$. It is easy to see that the entailment in fact holds for arbitrary $k \geq 1, l \geq 1, k' \geq 1, l' \geq 1$. It is no surprise that increasing the effort retains the beliefs in general, that is, effort behaves monotonically.

Theorem 22 Suppose $\mathbf{O}_k^l \Gamma \approx \mathbf{B}_{k'}^{l'}(\phi \Rightarrow \psi)$.

Then $\mathbf{O}_{\tilde{k}}^{\tilde{l}} \Gamma \approx \mathbf{B}_{\tilde{k}'}^{\tilde{l}'}(\phi \Rightarrow \psi)$ for all $\tilde{k} \geq k, \tilde{l} \geq l, \tilde{k}' \geq k', \tilde{l}' \geq l'$.

Proof sketch. Suppose $\vec{e} \models \mathbf{O}\Gamma$, $\vec{s} \approx \mathbf{O}_k^l \Gamma$, and $\vec{s}' \approx \mathbf{O}_{\tilde{k}}^{\tilde{l}} \Gamma$. The key argument is that \vec{s}' is at least as faithful to \vec{e} as \vec{s} is (cf. Figure 1). This is proven by inductions on \tilde{k} and \tilde{l} using Lemma 18. It is then easy to see that \vec{s}' entails at least the beliefs that \vec{s} does. Again using Lemma 18, we can then show that beliefs proved with effort k', l' can also be proved for \tilde{k}', \tilde{l}' . \square

More important perhaps is the question whether \mathcal{BOC} is sound wrt its archetype \mathcal{BO} . Indeed this is the case for belief implications with proper⁺ knowledge bases, as expressed by the following theorem.

Theorem 23 If $\mathbf{O}_k^l \Gamma \approx \mathbf{B}_{k'}^{l'}(\phi \Rightarrow \psi)$, then $\mathbf{O}\Gamma \models \mathbf{B}(\phi \Rightarrow \psi)$.

Proof sketch. By Theorem 20, there is a unique (modulo UP^+) \vec{s} such that $\vec{s} \approx \mathbf{O}_k^l \Gamma$. All spheres but its last one faithfully match the corresponding spheres of the unique \vec{e} such that $\vec{e} \models \mathbf{O}\Gamma$, and the final sphere of \vec{s} is weaker than the last sphere of \vec{e} (cf. Figure 1), so everything that can be inferred from a sphere of \vec{s} by sound inference

can also be inferred from \vec{e} . It is then easy to show that $[\vec{e} | \phi] \leq [\vec{s}, l' \dot{\phi}]$. Since sound inference is applied to prove $(\phi \supset \psi)$, the claim follows. \square

5.2 Decidability of belief implications

Finally, we investigate computational questions of belief implications $\mathbf{O}_k^l \Gamma \approx \mathbf{B}_{k'}^{l'}(\phi \Rightarrow \psi)$. We shall see that the problem is decidable for proper⁺ knowledge bases, and in the propositional case even tractable for fixed effort.

The fundamental idea behind the decidability result is that standard names that do not occur in the knowledge base or query cannot be distinguished. Hence we only need to consider a finite number of them: those from the knowledge base and query, plus a few more (their number is bounded by the number of quantifiers and maximum arity in the knowledge base and query). We first present decision procedures \mathbf{S} and \mathbf{C} for \approx and \approx , and finally the procedure \mathbf{B} for belief implications.

Again we let π and Γ be proper⁺.

Definition 24 We let $N(\pi, \phi, j)$ contain all names that occur in the formulas π or ϕ plus $(j + 1) \cdot \max\{|\pi|_w, |\phi|_w\}$ additional names, where $|v|_w$ is the maximum of the largest number of free variables in any subformula of v and the highest arity in v . For any set of names N , we let $\text{gnd}_N(\phi)$ and $s \otimes_N \ell$ be as $\text{gnd}(\phi)$ and $s \otimes \ell$ except that the grounding is restricted to the names in N .

As sketched above, to decide $\text{gnd}(\pi), k \approx \phi$ it suffices to consider only names from $N(\pi, \phi, k)$ for grounding, quantification, and splitting, and it is moreover easy to see that only literals whose symbols occur in π or ϕ need to be considered. These ideas lead to the procedure $\mathbf{S}[N, s, k, \phi] \in \{0, 1\}$ with the following inductive definition:

- $\mathbf{S}[N, s, k + 1, \phi] = 1$ iff $\mathbf{S}[N, s \cup \{\{\ell\}, k, \phi\}] = \mathbf{S}[N, s \cup \{\{\bar{\ell}\}, k, \phi\}] = 1$ for some ground literal ℓ whose symbol occurs in s or ϕ and whose names are from N ;
- if c is a clause: $\mathbf{S}[N, s, 0, c] = 1$ iff $c \in \text{UP}^+(s)$;
- if $(\phi \vee \psi)$ is not a clause: $\mathbf{S}[N, s, 0, (\phi \vee \psi)] = \max\{\mathbf{S}[N, s, 0, \phi], \mathbf{S}[N, s, 0, \psi]\}$;
- $\mathbf{S}[N, s, 0, \neg(\phi \vee \psi)] = \min\{\mathbf{S}[N, s, 0, \neg\phi], \mathbf{S}[N, s, 0, \neg\psi]\}$;
- $\mathbf{S}[N, s, 0, \neg\neg\phi] = \mathbf{S}[N, s, 0, \phi]$;
- $\mathbf{S}[N, s, 0, \exists x \phi] = \max\{\mathbf{S}[N, s, 0, \phi_n^x] \mid n \in N\}$;
- $\mathbf{S}[N, s, 0, \neg\exists x \phi] = \min\{\mathbf{S}[N, s, 0, \neg\phi_n^x] \mid n \in N\}$.

The following theorem says that \mathbf{S} is a decision procedure for $\text{gnd}(\pi), k \approx \phi$.

Theorem 25 $\text{gnd}(\pi), k \approx \phi$ iff $\mathbf{S}[N, \text{gnd}_N(\pi), k, \phi] = 1$ where $N = N(\pi, \phi, k)$.

Proof sketch. The key tool to show the theorem are bijections between standard names that leave the names from π and ϕ unchanged but possibly swap any other names. For any clause c that mentions no more than $\max\{|\pi|_w, |\phi|_w\}$ names that do not occur in π or ϕ , and a bijection $*$ that swaps these names with the additional names in N , it can be shown that $c \in \text{UP}^+(\text{gnd}(\pi))$ iff $c^* \in \text{UP}^+(\text{gnd}_N(\pi))$; this basically allows us to restrict grounding of π to N . Similarly, it can be shown that quantification and splitting can be restricted to names from N . Finally it is intuitively immediate that splitting only literals whose symbols occur in π or ϕ can generate new inferences. \square

Corollary 26 $\text{gnd}(\pi), k \models \phi$ is decidable. In the propositional case, the time complexity is $\mathcal{O}((|\pi| + k)^{k+1} \cdot |\phi|^{k+1} \cdot 2^k)$.

In a very similar fashion we can design a decision procedure for $\text{gnd}(\pi), l \models \phi$. For analogous reasons as in \models , it suffices to consider only names from $N(\pi, \phi, l)$ and to augment the setup only with literals whose symbols occur in π or ϕ . The resulting procedure $C[N, s, l, \phi] \in \{0, 1\}$ is defined inductively as follows:

- $C[N, s, l+1, \phi] = 1$ iff $C[N, s \otimes_N \ell, l, \phi] = 1$ for all (including non-ground) literals ℓ whose symbols occur in s or ϕ and whose names are from N ;
- if ℓ is a positive literal:
 $C[N, s, 0, \ell] = C[N, s, 0, \neg\bar{\ell}]$;
- if c is a clause:
 $C[N, s, 0, \neg c] = 1$ iff $[\] \in \text{XP}(s)$ or $c \notin \text{UP}^+(s)$;
- $C[N, s, 0, (\phi \vee \psi)] = \max\{C[N, s, 0, \phi], C[N, s, 0, \psi]\}$;
- if $(\phi \vee \psi)$ is not a clause:
 $C[N, s, 0, \neg(\phi \vee \psi)] = \min\{C[N, s, 0, \neg\phi], C[N, s, 0, \neg\psi]\}$;
- if $\neg\phi$ is not a clause:
 $C[N, s, 0, \neg\neg\phi] = C[N, s, 0, \phi]$;
- $C[N, s, 0, \exists x \phi] = \max\{C[N, s, 0, \phi_n^x] \mid n \in N\}$;
- $C[N, s, 0, \neg\exists x \phi] = \min\{C[N, s, 0, \neg\phi_n^x] \mid n \in N\}$.

For similar reasons as for \mathbb{S} and \models , we can show that C is a decision procedure for $\text{gnd}(\pi), l \models \phi$.

Theorem 27 $\text{gnd}(\pi), l \models \phi$ iff $C[N, \text{gnd}_N(\pi), l, \phi] = 1$ where $N = N(\pi, \phi, l)$.

Corollary 28 $\text{gnd}(\pi), l \models \phi$ are decidable. In the propositional case, the time complexity is $\mathcal{O}((|\pi| + l)^{l+1} \cdot |\phi|^{l+1})$.

So far we have established that reasoning in \models and \models is decidable for proper⁺ knowledge bases, and that it is tractable for given effort in the propositional case.

With these decision procedures for the objective limited semantics, it is easy to translate \mathcal{BOL} 's semantic rules for conditional belief and only-believing to a decision procedure for limited belief implications $\mathbf{O}_k^l \Gamma \approx \mathbf{B}_{k'}^l(\phi \Rightarrow \psi)$ for proper⁺ $\Gamma = \{\phi_1 \Rightarrow \psi_1, \dots, \phi_m \Rightarrow \psi_m\}$. The steps of the procedure $\mathbf{B}[N, k, l, k', l', \Gamma, \phi, \psi] \in \{0, 1\}$ are as follows:

- let s'_1, \dots, s'_{m+1} be such that
$$s'_p = \text{gnd}_N(\bigwedge_{i:p=1 \text{ or } S[N, s'_{p-1}, k, \neg\phi_i]=1} \text{NF}[(\phi_i \supset \psi_i)]);$$
- let $p^* = \max\{p \in \{1, \dots, m\} \mid \text{for some } i \in \{1, \dots, m\}, \max\{S[N, s'_p, k, \neg\phi_i] \mid p' < p\} = \max\{C[N, s'_p, l, \neg\phi_i] \mid p' < p\}\}$;
- let $(s_1, \dots, s_{p^*}, s_{p^*+1}) = (s'_1, \dots, s'_{p^*}, s'_{m+1})$;
- let $p^* = \min\{p \mid C[N, l', s_p, \neg\phi] = 0 \text{ or } p = p^* + 1\}$;
- return $S[N, k', s_{p^*}, (\phi \supset \psi)]$.

Theorem 29 $\mathbf{O}_k^l \Gamma \approx \mathbf{B}_{k'}^l(\phi \Rightarrow \psi)$ iff $\mathbf{B}[N, k, l, k', l', \Gamma, \phi, \psi] = 1$ where $N = N(\bigwedge_i \text{NF}[(\phi_i \supset \psi_i)], (\phi \supset \psi), \max\{k, l, k', l'\})$.

Proof sketch. The setups s'_1, \dots, s'_{m+1} in \mathbf{B} correspond to \vec{s}' in Rule 5 of \mathcal{BOL} 's semantics, for it is sufficient to consider only $m+1$ many setups, since the number of different setups in \vec{s}' can be shown to be bounded by $m+1$. Then s'_{p^*} is the last sphere that is l_k -plausibility-consistent wrt Γ , and therefore s_1, \dots, s_{p^*+1} in \mathbf{B} correspond to \vec{s} in Rule 5. Finally p^* is denotes the approximated plausibility of ϕ , and the last line evaluates $(\phi \supset \psi)$ in that sphere. \square

Corollary 30 $\mathbf{O}_k^l \Gamma \approx \mathbf{B}_{k'}^l(\phi \Rightarrow \psi)$ is decidable. For propositional Γ and ϕ , the time complexity is $\mathcal{O}(m^2 \cdot (|\Gamma| + j)^{2 \cdot (j+1)} + m \cdot (|\Gamma| + j')^{j+1} \cdot |(\phi \supset \psi)|^{j'+1})$, where $j = \max\{k, l\}$, $j' = \max\{k', l'\}$, and $|\Gamma| = |\bigwedge_i (\phi_i \supset \psi_i)|$.

Note that the time complexity is exponential only in the effort parameters. For fixed effort, propositional limited belief is hence tractable.

6 CONCLUSION

This paper introduces a logic of limited conditional belief. It is shown that reasoning in proper⁺ knowledge bases is decidable, and even tractable in the propositional case. This is achieved by limiting the effort to be spend on the reasoning task, thereby avoiding logical omniscience while retaining the first-order expressivity in the query. Generalizing LLL's framework of limited reasoning to conditional belief turned out to be surprisingly complicated. This is chiefly due to the prominent role of plausibilities in conditional belief.

Semantically a conditional knowledge base can be uniquely represented by a system of spheres; inference then boils down to model-checking. However, as it is undecidable in general which system of spheres corresponds to the knowledge base, in limited reasoning the best we can do is work with an approximation of the system of spheres. By approximating the plausibilities of formulas from below and above, we came up with an approximative system whose first spheres faithfully represent the unlimited spheres, and whose last sphere conservatively approximates the outermost sphere of the unlimited system. Given such an approximative system of spheres for a conditional knowledge base, a limited conditional belief is evaluated by approximating the plausibility of its antecedent from above to select a sphere, and then applying sound inference in that sphere.

We see several interesting avenues of future work. First and foremost, we are currently working on an implementation of the reasoning service described here, enriched with functions in the spirit of [21]. The treatment of functions from [21] seamlessly carries over to our logic; we skipped it here for simplicity. What makes functions very attractive is, among other things, that they allow to express existentials in the knowledge base by way of Skolemization. With that implementation, we plan to explore the practical utility of limited reasoning (for example, for high-level control in robots) in general and conditional belief in particular. Then the question will arise which effort parameters to choose. A simple approach may be to iteratively increase the effort with a situation-dependent timeout after which the search procedure is aborted. Even when the expected result could not be proved within the timeout, the tried effort parameters will give the system designer insight about why his program or robot behaved the way it did.

We also plan to investigate notions of limited belief revision. The problem with many belief revision operators is that they bring along exponential growth in the number of iterated revisions. We hope to alleviate this with a limited revision operator where the revised epistemic state is an approximative structure, similar as in limited only-believing. Such a notion of limited revision could be used to devise a limited variant of the situation calculus in the style of [20] but extended to defeasible beliefs and imperfect sensing [35, 36].

ACKNOWLEDGEMENTS

This work was supported by the German National Science Foundation (DFG) research unit FOR 1513.

References

- [1] Alan R. Anderson and Nuel D. Belnap, *Entailment: The Logic of Relevance and Necessity*, Princeton University Press, 1975.
- [2] Franz Baader, Ralf Küsters, and Frank Wolter, 'Extensions to description logics', in *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, pp. 219–261, (2003).
- [3] Nuel D. Belnap, 'A useful four-valued logic', in *Modern uses of multiple-valued logic*, 5–37, Springer, (1977).
- [4] Egon Börger, Erich Grädel, and Yuri Gurevich, *The Classical Decision Problem*, Springer, 1997.
- [5] James P. Delgrande, 'A framework for logics of explicit belief', *Computational Intelligence*, **11**(1), 47–88, (1995).
- [6] Ronald Fagin and Joseph Y. Halpern, 'Belief, awareness, and limited reasoning', *Artificial Intelligence*, **34**(1), 39–76, (1987).
- [7] Marcelo Finger and Renata Wassermann, 'Anytime approximations of classical logic from above', *Journal of Logic and Computation*, **17**(1), 53–82, (2007).
- [8] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi, 'On the decision problem for two-variable first-order logic', *Bulletin of symbolic logic*, **3**(01), 53–69, (1997).
- [9] Adam Grove, 'Two modellings for theory change', *Journal of Philosophical Logic*, **17**(2), 157–170, (1988).
- [10] Jaakko Hintikka, *Knowledge and Belief: An Introduction to the Logic of the Two Notions*, Cornell University Press, 1962.
- [11] Jaakko Hintikka, 'Impossible possible worlds vindicated', *Journal of Philosophical Logic*, **4**(4), 475–484, (1975).
- [12] Hirofumi Katsuno and Alberto O. Mendelzon, 'Propositional knowledge base revision and minimal change', *Artificial Intelligence*, **52**(3), 263–294, (1991).
- [13] Gabriele Kern-Isberner, *Conditionals in Nonmonotonic Reasoning and Belief Revision*, Springer, 2001.
- [14] Toryn Q. Klassen, Sheila A. McIlraith, and Hector J. Levesque, 'Towards tractable inference for resource-bounded agents', in *Proceedings of the Twelfth International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense)*, (2015).
- [15] Kurt Konolige, 'A deduction model of belief', *Research notes in Artificial Intelligence*, (1986).
- [16] Gerhard Lakemeyer, 'Limited reasoning in first-order knowledge bases', *Artificial Intelligence*, **71**(2), 213–255, (1994).
- [17] Gerhard Lakemeyer, 'Limited reasoning in first-order knowledge bases with full introspection', *Artificial Intelligence*, **84**(1), 209–255, (1996).
- [18] Gerhard Lakemeyer and Hector J. Levesque, 'Evaluation-based reasoning with disjunctive information in first-order knowledge bases', in *Proceedings of the Eighth Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 73–81, (2002).
- [19] Gerhard Lakemeyer and Hector J. Levesque, 'Decidable reasoning in a logic of limited belief with introspection and unknown individuals', in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, (2013).
- [20] Gerhard Lakemeyer and Hector J. Levesque, 'Decidable reasoning in a fragment of the epistemic situation calculus', in *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 468–477, (2014).
- [21] Gerhard Lakemeyer and Hector J. Levesque, 'Decidable reasoning in a logic of limited belief with function symbols', in *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, (2016). To appear.
- [22] Hector J. Levesque, 'A logic of implicit and explicit belief', in *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI)*, pp. 198–202, (1984).
- [23] Hector J. Levesque, 'All I know: a study in autoepistemic logic', *Artificial Intelligence*, **42**(2), 263–309, (1990).
- [24] Hector J. Levesque and Gerhard Lakemeyer, *The Logic of Knowledge Bases*, MIT Press, 2001.
- [25] David Lewis, *Counterfactuals*, John Wiley & Sons, 1973.
- [26] Yongmei Liu, Gerhard Lakemeyer, and Hector J. Levesque, 'A logic of limited belief for reasoning with disjunctive information', in *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, (2004).
- [27] Christopher Lynch, 'Unsound theorem proving', in *Proceedings of the Workshop on Disproving: Non-Theorems, Non-Validity, Non-Provability*, pp. 1–12, (2004).
- [28] Michael Mortimer, 'On languages with two variables', *Mathematical Logic Quarterly*, **21**(1), 135–140, (1975).
- [29] Daniele Nardi and Ronald J. Brachman, 'An introduction to description logics', in *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, pp. 5–44, (2003).
- [30] Peter F. Patel-Schneider, 'A decidable first-order logic for knowledge representation', *Journal of automated reasoning*, **6**(4), 361–388, (1990).
- [31] Judea Pearl, 'System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning', in *Proceedings of the Third Conference on Theoretical Aspects of Reasoning about Knowledge (TARK)*, pp. 121–135, (1990).
- [32] Marco Schaerf and Marco Cadoli, 'Tractable reasoning via approximation', *Artificial Intelligence*, **74**(2), 249–310, (1995).
- [33] Christoph Schwering, *Conditional Beliefs in Action*, Ph.D. dissertation, RWTH Aachen University, 2016.
- [34] Christoph Schwering and Gerhard Lakemeyer, 'A semantic account of iterated belief revision in the situation calculus', in *Proceedings of the Twenty-First European Conference on Artificial Intelligence (ECAI)*, pp. 801–806, (2014).
- [35] Christoph Schwering, Gerhard Lakemeyer, and Maurice Pagnucco, 'Belief revision and progression of knowledge bases in the epistemic situation calculus', in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3214–3220, (2015).
- [36] Christoph Schwering, Gerhard Lakemeyer, and Maurice Pagnucco, 'Belief revision and projection in the epistemic situation calculus'. Forthcoming, 2016.
- [37] Dana Scott, 'A decision method for validity of sentences in two variables', *Journal of Symbolic Logic*, **27**(377), 74, (1962).
- [38] Moshe Y. Vardi, 'On epistemic logic and logical omniscience', in *Proceedings of the First Conference on Theoretical Aspects of Reasoning about Knowledge (TARK)*, pp. 293–305, (1986).

A Temporal-Causal Modelling Approach to Integrated Contagion and Network Change in Social Networks

Romy Blankendaal, Sarah Parinussa and Jan Treur¹

Abstract. This paper introduces an integrated adaptive temporal-causal network model for dynamics in networks of social interactions addressing contagion between states, and changing connections within these social networks by two principles: the homophily principle and the more-becomes-more principle. The model has been evaluated in three different manners: by simulation experiments, by verification based on mathematical analysis, and by validation against an empirical data set.

1 INTRODUCTION

In today's world being successful and popular is mostly influenced by your social capabilities and how you interact with the people you know and work with. These social interactions are heavily investigated over the last few decades, in which analysis and prediction of the behaviour of humans in social situations plays a major role. The area of Social Networks has already a longer tradition, starting in the Social Sciences over 40 years ago. More recently, it has gradually developed in other disciplines as well; see, for example [4, 7, 23]. This development also involves computational methods to analyse and simulate networks both from the perspective of network structure and of dynamics.

Two main types of dynamics in relation to networks can be distinguished: dynamics *within* a given network structure (e.g., social contagion), and dynamics *of* a network (evolving networks). In the former case the network stays the same, but *states* (nodes) in the network change their *level* over time. In the latter case the network *connections* change, for example, their *weights* may increase or decrease. An example of this, also used in Section 6, is a network of adolescents followed over the years with the individuals' opinions about alcohol drinking as states and their friendships as connections. These states and connections both change over time. In many cases the two types of dynamics are addressed computationally as separate phenomena. This paper will address both types of dynamics and their interactions in an integrated manner.

The computational modelling approach used is the Network-Oriented Modelling approach based on temporal-causal networks described in [21, 22]; see also the ECAI'16 tutorial on Network-Oriented Modelling. This approach is a generic, dynamic AI modelling approach based on networks of causal relations but it differs from most other causal approaches (e.g., [17]) in that it incorporates a continuous time dimension to model dynamics in an adaptive manner, both of the states and of the network itself. This

temporal dimension enables causal reasoning and simulation for cyclic and adaptive causal networks, such as networks for connected mental or brain states, or for social interactions, or both. The modelling approach can incorporate ingredients that are sometimes used in specific types of (continuous time, recurrent) neural network models, and ingredients that are often used in probabilistic or possibilistic modelling. It is more generic than such methods in the sense that a much wider variety of modelling elements are provided, enabling the modelling of many types of dynamical systems, as described in [21, 22].

For the dynamics within a network an existing approach to social contagion will be adopted. For the dynamics of the network, two different principles are considered and integrated, namely the *homophily* principle and the *more becomes more* principle. The main objective of this paper is to explore how combining of these three models for social contagion and network evolution can be used to analyse and predict human behaviour in social situations. The *contagion* principle indicates that the more a person interacts with someone else, the more their opinions or beliefs or emotions or other states will converge; e.g., [5]. The *homophily* principle in a sense indicates the converse of this: the similarity of states such as opinions or beliefs or emotions of persons affects the strength of the connection between them; e.g., [6, 12, 13, 19]. The *more becomes more principle* describes the phenomenon that whenever someone is popular (in the sense of the strength of connections), he or she will become more popular over time because it is believed that this person is worth relating to; e.g., [2, 15].

In this paper, in Section 2 the Network-Oriented Modelling approach based on temporal-causal modelling networks used is briefly described. Section 3 introduces the integrated network model, and in Section 4 simulation experiments are discussed. Section 5 addresses verification of the model by mathematical analysis of equilibria. In Section 6 validation of the model is addressed based on empirical data from [8]. Finally, a discussion is provided in Section 7.

2 TEMPORAL-CAUSAL NETWORK MODELS

Causal modelling, causal reasoning and causal simulation have a long tradition in AI; e.g., [10, 11, 17]. One of the challenges has been that causal modelling involving cyclic graphs is difficult; therefore, many approaches limit themselves to Directed Acyclic Graphs (DAG's). The Network-Oriented Modelling approach based on temporal-causal networks described in [21, 22] can be viewed as part of this tradition. The computational model presented here has been designed using this network modelling approach. It is a widely usable generic dynamic AI modelling approach that

¹ Computer Science Department, VU University Amsterdam, The Netherlands, romy.blankendaal@gmail.com, sparinussa@gmail.com, j.treur@vu.nl

distinguishes itself by incorporating a dynamic and adaptive perspective, both on states and causal relations. This dynamic perspective takes the form of an added continuous time dimension, enabling modelling of cyclic and adaptive networks, and also of timing of causal effects. Due to this, causal reasoning and simulation is possible for adaptive networks that inherently contain cycles, such as adaptive networks for connected mental or brain states, or for social interaction. From the technical point of view, it has some ingredients in common with specific types of (continuous time, recurrent) neural network models, but is more generic in the sense that a much wider variety of modelling elements can be used, as will also be shown by the integrated model presented here. In [21, 22] a more detailed description of this Network-Oriented Modelling approach can be found.

According to the adopted modelling approach, a model is designed at a conceptual level, for example, in the form of a graphical conceptual representation or a conceptual matrix representation. A graphical conceptual representation displays nodes for *states* and arrows for *connections* indicating causal impacts from one state to another, and includes some additional information in the form of:

- for each connection from a state X to a state Y a *connection weight* $\omega_{X,Y}$ (for the strength of the impact of X on Y)
- for each state Y a *speed factor* η_Y (for the timing of the effect of the impact)
- for each state Y the type of *combination function* $\mathbf{c}_Y(\dots)$ used (to aggregate multiple impacts on a state)

To choose combination functions, a number of standard options is available, varying from linear functions or logistic functions, to product or max and min-based functions as often used in probabilistic and possibilistic approaches; e.g., [21]. The conceptual representation of a model can be transformed in a systematic or even automated manner into a numerical representation of the model as follows [21]; here the variable t indicates a time point; it varies over the real numbers.

From conceptual to numerical representation

- at each time point t each state Y in the model has a real number value (usually in the interval $[0, 1]$), denoted by $Y(t)$
- at each time point t each state X connected to state Y has an impact on Y defined as $\mathbf{impact}_{X,Y}(t) = \omega_{X,Y} X(t)$ where $\omega_{X,Y}$ is the weight of the connection from X to Y
- The *aggregated impact* of multiple states X_i on Y at t is determined using a *combination function* $\mathbf{c}_Y(\dots)$:

$$\begin{aligned} \mathbf{aggimpact}_Y(t) &= \mathbf{c}_Y(\mathbf{impact}_{X_1,Y}(t), \dots, \mathbf{impact}_{X_k,Y}(t)) \\ &= \mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) \end{aligned}$$

where X_i are the states with connections to state Y

- The effect of $\mathbf{aggimpact}_Y(t)$ on Y is exerted over time gradually, depending on speed factor η_Y :

$$\begin{aligned} Y(t+\Delta t) &= Y(t) + \eta_Y [\mathbf{aggimpact}_Y(t) - Y(t)] \Delta t \\ \mathbf{d}Y(t)/\mathbf{d}t &= \eta_Y [\mathbf{aggimpact}_Y(t) - Y(t)] \end{aligned}$$

- This provides a *difference* and *differential equation* for Y :

$$\begin{aligned} Y(t+\Delta t) &= Y(t) + \eta_Y [\mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) - Y(t)] \Delta t \\ \mathbf{d}Y(t)/\mathbf{d}t &= \eta_Y [\mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) - Y(t)] \end{aligned}$$

These numerical representations can be used for mathematical and computational analysis and simulation.

In cases in which connection weights $\omega_{X,Y}$ are dynamic, they are also considered as states. This means that in graphical conceptual representations connection weights which usually are depicted as labels for arrows, can also be handled as nodes: also arrows can occur from and to them, as shown, for example in Fig.

1 and Fig. 2. For numerical representations dynamic connection weights also get a time argument: $\omega_{X,Y}(t)$. So the difference and differential equation for a state Y becomes:

$$\begin{aligned} Y(t+\Delta t) &= Y(t) + \eta_Y [\mathbf{c}_Y(\omega_{X_1,Y}(t)X_1(t), \dots, \omega_{X_k,Y}(t)X_k(t)) - Y(t)] \Delta t \\ \mathbf{d}Y(t)/\mathbf{d}t &= \eta_Y [\mathbf{c}_Y(\omega_{X_1,Y}(t)X_1(t), \dots, \omega_{X_k,Y}(t)X_k(t)) - Y(t)] \end{aligned}$$

Moreover, as they are considered states themselves, to model their dynamics, the dynamic connection weights will also be described by a difference or differential equation, which also can be based on a combination function and speed factor as above, and even on weights of connections to or from these connection weights (the latter will be assumed to have value 1 here). This will be illustrated in detail in next section.

3 THE COMPUTATIONAL MODEL

Three different elements will be addressed by the computational model: the *contagion* principle, the *homophily* principle, and the *more becomes more* principle.

Contagion principle. This principle indicates that levels of states of connected nodes affect each other. A most basic form is that they are adjusted in a way that they become more equal, which can be considered a form of averaging; sometimes this is called absorption [5]. Also possible is that the level of one state amplifies the level of another state, so that spirals can occur; this is called amplification [5]. In the current paper an absorption model is used.

Homophily principle. This principle indicates that the more similar (the levels of) the states of two connected nodes are, the stronger their connection will become: ‘birds of a feather flock together’ [6, 12, 13].

More becomes more principle. This principle expresses that nodes that already have more and stronger connections get more and stronger additional connections than nodes with less or weaker connections (the rich become more rich and the poor remain poor). Analyses have been made showing that applying this principle usually leads to scale-free networks [2, 15]. When both the states and the connection weights are assumed dynamic, this leads to a circular causal relation state \rightleftarrows connection. This may cause difficulties in explaining by which causes in the past observed phenomena in networks have developed. For example, when in a network it is found that similar state levels and strong connections occur together, due to such a circular causal relation it is difficult to tell which type of principle(s) was originally causing this situation; see for example: [1, 18, 20, 14].

3.1 Dynamics of States: Social Contagion

In this section, X_A denotes the level of state X for member A . This state X can be any type of state, either internal or externally observable (e.g. an internal state of feeling an emotion, or an expressed emotion state, or an (internal) intention, or an action performed, or a belief or opinion). The connection weights are denoted by $\omega_{A,B}$ for the connection from A to B . The following general model for contagion is used (see [21]):

$$\begin{aligned} \mathbf{d}X_B/\mathbf{d}t &= \eta_B [\mathbf{c}_B(\omega_{A_1,B}X_{A_1}, \dots, \omega_{A_k,B}X_{A_k}) - X_B] \\ X_B(t+\Delta t) &= X_B(t) + \end{aligned}$$

$$\eta_B [\mathbf{c}_B(\omega_{A_1,B}X_{A_1}(t), \dots, \omega_{A_k,B}X_{A_k}(t)) - X_B(t)] \Delta t$$

For the states X_B the model uses the scaled sum combination function (also see [5, 21]):

$$\mathbf{c}_B(V_1, \dots, V_k) = \mathbf{ssum}_k(V_1, \dots, V_k) = (V_1 + \dots + V_k) / \lambda$$

with $\lambda = \omega_{A_1,B} + \dots + \omega_{A_k,B}$ the sum of the incoming weights for state X_B . This makes

$$c_B(\omega_{A_1,B}X_{A_1}(t), \dots, \omega_{A_k,B}X_{A_k}(t)) = (\omega_{A_1,B}X_{A_1}(t) + \dots + \omega_{A_k,B}X_{A_k}(t)) / (\omega_{A_1,B} + \dots + \omega_{A_k,B})$$

This is the weighted average of the state levels $X_{A_1}(t), \dots, X_{A_k}(t)$ with weights proportional to $\omega_{A_1,B}, \dots, \omega_{A_k,B}$, respectively. Using this combination function for the aggregated impact on state X_B , the differential and difference equation are

$$\begin{aligned} dX_B/dt &= \eta_B [(\omega_{A_1,B}X_A) + \dots + \omega_{A_k,B}X_{A_k}) / (\omega_{A_1,B} + \dots + \omega_{A_k,B}) - X_B] \\ X_B(t+\Delta t) &= X_B(t) + \eta_B [(\omega_{A_1,B}X_{A_1}(t) + \dots + \omega_{A_k,B}X_{A_k}(t)) / (\omega_{A_1,B} + \dots + \omega_{A_k,B}) - X_B(t)] \Delta t \end{aligned}$$

3.2 Dynamics of Connections: Homophily

The network characteristics are specified by the connection strengths $\omega_{A,B}$. A first question to be answered is how these connection strengths are changing, and, in particular, the other states affecting them have to be identified. By the homophily principle the connection strengths $\omega_{A,B}$ are affected by the activation levels of the connected states of A and B . Such a dependency is depicted in fig. 1. Note that by adding these effects on the connection strengths cyclic relationships occur; for example see Figure 1.

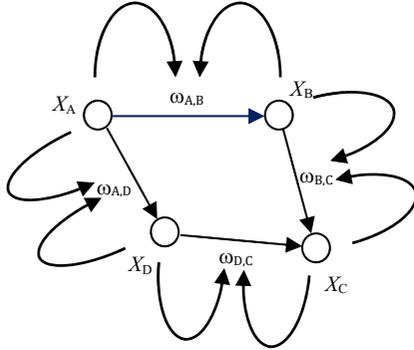


Figure 1. Conceptual representation: homophily principle

A next step is to determine how exactly the connection strengths are affected by the activation levels. This is needed to obtain a dynamic equation for $\omega_{A,B}$. For the current model the dynamic connection weights $\omega_{A,B}$ are assumed to change over time based on a principle similar to the one from [16]: the closer the activation levels of the states, the stronger the mutual connections between the members will become, and the higher the difference between the activation levels, the weaker they will become. In other words: activation levels close to each other imply a strong upward change in $\omega_{A,B}$, and activation levels far apart imply a downward change of $\omega_{A,B}$. This is how the homophily principle works: the more you are alike, the more you like (each other); the inspiration for the model below was obtained from [6, 12, 13, 19].

As an example of this principle in practical use, if you wonder whether there is a chance to become connected with somebody, you might consider whether you often like and agree on the same things. It can often be observed that persons that have close relationships or friendships are alike in some respects; e.g. they go to the same clubs, take the same drinks, have the same opinions, vote for the same or similar parties. Such observations might be considered support for the homophily principle: in the past they were attracted to each other due to being alike. However, also a different explanation is possible: they were often together and due

to that they affected each other's states by social contagion, and therefore they became alike. So, the cyclic relation between X_B and $\omega_{A,B}$ as mentioned above leads to two possible causal explanations of a state of being alike and a state of being connected:

$$\begin{aligned} \text{being connected} &\rightarrow \text{being alike} \\ \text{being alike} &\rightarrow \text{being connected} \end{aligned}$$

Such circular causal relations make it difficult to determine what came first. It may be a state just emerging from a cyclic process without a single cause. For more discussion on this issue, for example, see [1, 14, 18, 20].

The homophily principle may be formalised using a combination function $c_{A,B}(V_1, V_2, W)$ and speed factor $\eta_{A,B}$ according to the following general format (where connections to connection weights themselves are assumed to have weight 1):

$$\begin{aligned} \omega_{A,B}(t+\Delta t) &= \omega_{A,B}(t) + \eta_{A,B} [c_{A,B}(X_A(t), X_B(t), \omega_{A,B}(t)) - \omega_{A,B}(t)] \Delta t \\ d\omega_{A,B}/dt &= \eta_{A,B} [c_{A,B}(X_A, X_B, \omega_{A,B}) - \omega_{A,B}] \end{aligned}$$

Here it is assumed that the values of $\omega_{A,B}$ stay within the interval $[0, 1]$ and in particular the conditions

$$c_{A,B}(V_1, V_2, 0) \geq 0 \text{ and } c_{A,B}(V_1, V_2, 1) \leq 1$$

are fulfilled. The combination function $c_{A,B}(\dots)$ is assumed to depend on the one hand on W and on the other hand on the difference $|V_1 - V_2|$ (which is always between 0 and 1) in such a way that lower values of $|V_1 - V_2|$ relate to higher values of $c_{A,B}(V_1, V_2, W)$, and higher values of $|V_1 - V_2|$ relate to lower values of $c_{A,B}(V_1, V_2, W)$: the higher $|V_1 - V_2|$, the lower $c_{A,B}(V_1, V_2, W)$ and in particular:

$$\begin{aligned} |V_1 - V_2| = 1 &\Rightarrow \omega_{A,B} \text{ decreasing} \Rightarrow d\omega_{A,B}(t)/dt \leq 0 \\ &\Rightarrow c_{A,B}(V_1, V_2, W) \leq W \\ |V_1 - V_2| = 0 &\Rightarrow \omega_{A,B} \text{ increasing} \Rightarrow d\omega_{A,B}(t)/dt \geq 0 \\ &\Rightarrow c_{A,B}(V_1, V_2, W) \geq W \end{aligned}$$

Furthermore, it is assumed that $c_{A,B}(V_1, V_2, W)$ only depends on the difference $|V_1 - V_2|$ and not on the values of V_1 and V_2 themselves. Then as a simplification in notation the combination function $c_{A,B}(\dots)$ can be expressed as a function $h_{A,B}(D, W)$ of $D = |V_1 - V_2|$ and W : $c_{A,B}(V_1, V_2, W) = h_{A,B}(D, W)$. As discussed above, the function $h_{A,B}$ is assumed to be monotonically decreasing in D :

$$\begin{aligned} D_1 \leq D_2 &\Rightarrow h_{A,B}(D_1, W) \geq h_{A,B}(D_2, W) \\ D = 1 &\Rightarrow h_{A,B}(D, W) \leq W \\ D = 0 &\Rightarrow h_{A,B}(D, W) \geq W \end{aligned}$$

Moreover,

$$h_{A,B}(D, 0) \geq 0 \text{ and } h_{A,B}(D, 1) \leq 1$$

Somewhere between low values of $D = |V_1 - V_2|$ (with $h_{A,B}(D, W) \geq W$) and high values of D (with $h_{A,B}(D, W) \leq W$) a value for D is assumed for which $h_{A,B}(D, W) = W$. This is called the *homophily threshold value*, indicated by $\tau_{\text{homophily}}$ or by $\tau_{A,B}$; so

$$\begin{aligned} h_{A,B}(D, W) &\geq W && \text{when } D \leq \tau_{A,B} \\ h_{A,B}(D, W) &= W && \text{when } D = \tau_{A,B} \\ h_{A,B}(D, W) &\leq W && \text{when } D \geq \tau_{A,B} \end{aligned}$$

So, for this threshold value $\tau_{A,B}$ it holds:

- an upward change of connection weight $\omega_{A,B}$ occurs when $|V_1 - V_2| < \tau_{A,B}$
- no change of connection weight $\omega_{A,B}$ occurs when $|V_1 - V_2| = \tau_{A,B}$
- a downward change of connection weight $\omega_{A,B}$ occurs when $|V_1 - V_2| > \tau_{A,B}$

A simple example of a continuous function $h_{A,B}(D, W)$ satisfying the above conditions for a given value of W is obtained when the threshold value $\tau_{A,B}$ is assumed to be a fixed value and then use a simple decreasing linear function in D through the point with coordinates $(\tau_{A,B}, W)$ (i.e., through the point with $D = \tau_{A,B}$ and $h_{A,B}(D, W) = W$):

$$h_{A,B}(D, W) = W + \beta(\tau_{A,B} - D)$$

for some β , that still can be chosen. To fulfil the conditions

$$h_{A,B}(D, 0) \geq 0 \text{ and } h_{A,B}(D, 1) \leq 1$$

which prevent the weight value $\omega_{A,B}$ go outside the interval $[0, 1]$, β can be chosen as a function $\beta(W)$ of W which can suppress the term $\tau_{A,B} - D$ when W comes closer to 0 or 1:

$$h_{A,B}(D, W) = W + \beta(W)(\tau_{A,B} - D)$$

When this function $\beta(W)$ is assumed to be always ≥ 0 and close to 0 when W is close to 0 or 1, then this can keep the value of $\omega_{A,B}$ within the interval $[0, 1]$. This can be satisfied by the function

$$\beta(W) = W(1-W)$$

which is 0 for $W = 0$ and for $W = 1$, and positive between these values with a maximum 0.25 for $W = 0.5$. This makes that ω is changing slowly in the neighbourhood of 0 or 1, thus achieving that ω does not cross these boundaries. Then the following example function fulfilling the above conditions is obtained:

$$h_{A,B}(D, W) = W + W(1-W)(\tau_{A,B} - D)$$

For the combination function $c_{A,B}(V_1, V_2, W)$ the above choice for $h_{A,B}(D, W)$ translates into:

$$c_{A,B}(V_1, V_2, W) = W + W(1-W)(\tau_{A,B} - |V_1 - V_2|)$$

Using this combination function, the dynamic relations for $\omega_{A,B}$ are:

$$d\omega_{A,B}/dt = \eta_{A,B} \omega_{A,B}(1 - \omega_{A,B})(\tau_{A,B} - |X_A - X_B|)$$

$$\omega_{A,B}(t+\Delta t) = \omega_{A,B}(t) +$$

$$\eta_{A,B} \omega_{A,B}(t)(1 - \omega_{A,B}(t))(\tau_{A,B} - |X_A(t) - X_B(t)|) \Delta t$$

Note that in the example experiments discussed below one and the same homophily threshold value $\tau_{homophily}$ has been used for all connections.

3.3 Dynamics of Connections: More Becomes More

Another type of model for a dynamic connection from a member B to A takes into account to which extent other member's C connect to member A . The idea behind this is that somebody who is very popular seems worth connecting to. This is called the 'more becomes more' principle. For example, if B is followed by many others C on Twitter, then B seems to be interesting to follow for A as well. As the connections of others to B may change over time, this will imply that also A will have a dynamic connection to B , and in turn this connection will affect the connection of others to B over time as well. This can be modelled taking into account the weights $\omega_{C_i,B}$ for $i = 1, \dots, k$ of all connections from others C_i to B as follows. The inspiration for this model was obtained from [2, 15]. For a conceptual representation, see Fig. 2.

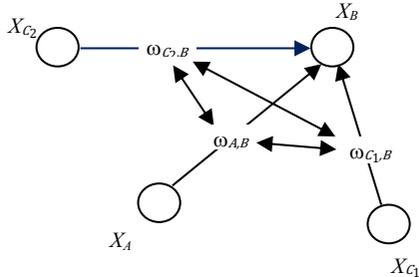


Figure 2. Conceptual representation: more becomes more principle

From this the following numerical representation is obtained:

$$d\omega_{A,B}/dt = \eta_{A,B}[c_{A,B}(\omega_{C_1,B}, \dots, \omega_{C_k,B}) - \omega_{A,B}]$$

$$\omega_{A,B}(t+\Delta t) = \omega_{A,B}(t) + \eta_{A,B}[c_{A,B}(\omega_{C_1,B}(t), \dots, \omega_{C_k,B}(t)) - \omega_{A,B}(t)]$$

Here $c_{A,B}(\dots)$ is a combination function for the values $\omega_{C_1,B}, \dots, \omega_{C_k,B}$, for example, a logistic sum function, or a scaled sum function. The latter is chosen here with as scale factor the number k of other members involved:

$$c_{A,B}(\omega_{C_1,B}, \dots, \omega_{C_k,B}) = \text{ssum}_k(\omega_{C_1,B}, \dots, \omega_{C_k,B}) = (\omega_{C_1,B} + \dots + \omega_{C_k,B})/k$$

Note that in a network modelling the adaptation of connection weights the direction of this influence is not automatically the direction involving social contagion; this will depend on the application considered. For example, a network modelling a connection from A to B when A is following B on Twitter will not play a role in social contagion from A to B . For social contagion the opposite network plays a role where a connection from A to B occurs when A is followed by B , which is not initiated by A but by B : on Twitter and most other social media you cannot appoint your own followers. In other cases, it may be different. For example, if A wants to announce an event or new product, he or she can choose an occasion where many others will see the message, for example, posting it on a suitable forum; in such a case both the initiation and the social contagion are directed from A to the others. For the sake of simplicity, the latter is assumed here.

3.4 Integration of the Different Models

Within the integrated model the connection weights are affected by both the homophily principle and the more becomes more principle. These two effects have to be integrated. To achieve this a combination function is used that combines both the combination function for the homophily principle and for the more becomes more principle. Recall that the combination function for $\omega_{A,B}$ based on the homophily principle is:

$$c_{\text{homo}, \omega_{A,B}}(X_A, X_B, \omega_{A,B}) = \omega_{A,B} + \omega_{A,B}(1 - \omega_{A,B}) * (\tau_{A,B} - |X_A - X_B|)$$

For the more becomes more principle the combination function for $\omega_{A,B}$ is:

$$c_{\text{more}, \omega_{A,B}}(\omega_{C_1,B}, \dots, \omega_{C_k,B}) = (\omega_{C_1,B} + \dots + \omega_{C_k,B})/k$$

In order to incorporate the influence of the separate models into the combined model, a parameter α between 0 and 1 is introduced and then for the integrated combination function $c_{\omega_{A,B}}(\dots)$ the weighted average is chosen of the two separate combination functions:

$$c_{\omega_{A,B}}(X_A, X_B, \omega_{A,B}, \omega_{C_1,B}, \dots, \omega_{C_k,B}) = \alpha * c_{\text{homo}, \omega_{A,B}}(X_A, X_B, \omega_{A,B}) + (1-\alpha) * c_{\text{more}, \omega_{A,B}}(\omega_{C_1,B}, \dots, \omega_{C_k,B})$$

This is used as aggregated impact in the following difference for weight $\omega_{A,B}$:

$$\omega_{A,B}(t+\Delta t) = \omega_{A,B}(t) +$$

$$\eta_{\omega_{A,B}} [c_{\omega_{A,B}}(X_A(t), X_B(t), \omega_{A,B}(t), \omega_{C_1,B}(t), \dots, \omega_{C_k,B}(t)) - \omega_{A,B}(t)]$$

Parameter α is called the *homophily influence fraction*. When α is close to 1, the *homophily* model is dominating, and when α is close to 0, the *more becomes more* model dominates. When α is 0.5, both models have equal influence on the new weight value.

4 SIMULATION EXPERIMENTS

The model described above can be used to make predictions about emergent properties of dynamic social networks. To examine this, some questions concerning emerging properties were formulated and these were tested by simulation experiments. Due to the *contagion* principle it may be expected that the state levels of the nodes connected in a social network to become more alike and due to the *homophily* principle and the *more becomes more* principle their connections will change over time. Via the *more becomes more* principle it may be expected that the nodes that are connected to nodes that have the most and stronger connections will keep

those connections and they will become stronger. The homophily principle also changes the connections depending on the levels of the nodes. An example social network has been designed to see what the outcome is for these questions in a case study. The network contains 14 nodes and consists of two groups that each are strongly interconnected and have only a few (bridge) connections between members of different groups; see Fig. 3. These could, for example, be two groups of friends of different high schools that have a few connections between them due to four members that went to the same primary school. The assumed connection weights are shown in the connection matrix in Table 1.

So, will the integrated model change the levels of the states in each of the groups in such a way that they will converge to the same value for this group, depending on the popular nodes and the amount and strength of the connections? Or will the levels of all nodes converge to one and the same value for both groups? Besides that, due to the homophily principle, will the bridge connections between the two separate groups become weaker as their states are too different, or will they become stronger due to the more become more principle and due to the levels in the groups becoming more similar? Many types of emerging dynamic patterns may occur, and it is not so easy to predict them at forehand.

In the example about the high school friend groups, by the homophily principle the bridge connections could disappear when the persons in first group all play basketball and the people in the other friend group don't like basketball. However, when both groups like basketball, it is quite likely that the connections between the two groups will become stronger. And the more becomes more principle may still have a stronger effect to increase the bridge connections.

Table 1. Initial weights of the connections in the example network

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A		0.3	0.6	0.8										
B	0.5			0.6										
C				0.7										
D	0.6				0.8		0.3							
E				0.8			0.6	0.7	0.6			0.6		
F				0.3			0.9							
G				0.7		0.4		0.8						
H					0.7		0.9							
I									0.6	0.6	0.9			
J									0.7		0.8			
K									0.8	0.7			0.6	
L				0.4			0.5	0.5				0.6	0.7	
M									0.4					0.6
N					0.8							0.7	0.6	

4.1 Analysis of the Example Social Network

Gephi version 0.8.2. [3] has been used to analyse the example social network, which is shown in Figure 3. It consists of 14 nodes or nodes and 39 edges. The average degree is 2.786 with a highest in-degree of 6, which means that there is one node who has 6 connections directed towards him/her; this is node D. As can be seen from Fig. 3, the most 'popular' nodes in the network are nodes D, E, I and L, these are the biggest and the size here indicates the number of connections for nodes. Node E has the highest betweenness centrality of 80.167, after which node D follows with 72.5, node L with 41 and node I with 38.167. The highest values for the between-ness centrality are most likely the popular members and have the most influence on the rest of the connecting nodes. The number of communities is 3 and the modularity is 0.369. The example network was designed as having 2 communities with a few nodes that also had bridge connections with the other community: nodes E, H, I, L, N.

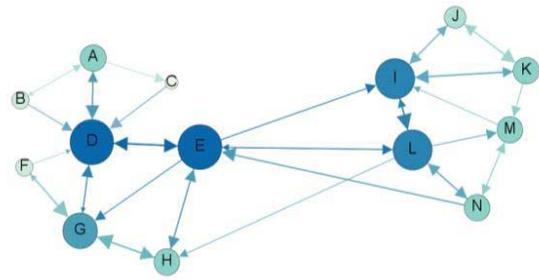


Figure 3. The example social network with nodes (circles) and edges (arrows with direction)

This group is also considered a community, probably since the modularity is 0.369 and the communities used here are small. For the two simulation experiments discussed here the parameter settings and initial values used are shown in Tables 1 to 3. Simulations were done in Matlab 2014v [4].

Table 2. Parameters used for the simulation experiments

Parameter	α	$\eta_{weights}$	η_{states}	$\tau_{homophily}$	time
Value	0.5	0.2	0.2	0.05	100

4.2 Simulation Results for Experiment 1

The first experiment addresses the case that the initial levels in the two groups do not differ much: the two groups have preferences about sports that match. The values of the levels of all nodes in the social network vary between 0.3 and 0.8. Because the levels are not so different, it may be expected that the bridge connections between the two groups (involving nodes E, H, I, L and N) will become stronger and the levels of all the nodes converge to the same value. Fig. 4 depicts the change of the levels of the nodes over time. The levels start at different initial values ranging from 0.3 to 0.8, as can be seen in Table 3. The graph shows that indeed all levels converge into one value, around 0.53.

Table 3. Initial values of the levels of the nodes

Node	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Experiment 1	0.4	0.6	0.3	0.7	0.5	0.6	0.4	0.8	0.5	0.3	0.6	0.6	0.7	0.4
Experiment 2	0.8	0.6	0.7	0.9	0.7	0.5	0.6	0.8	0.2	0.4	0.1	0.3	0.4	0.2

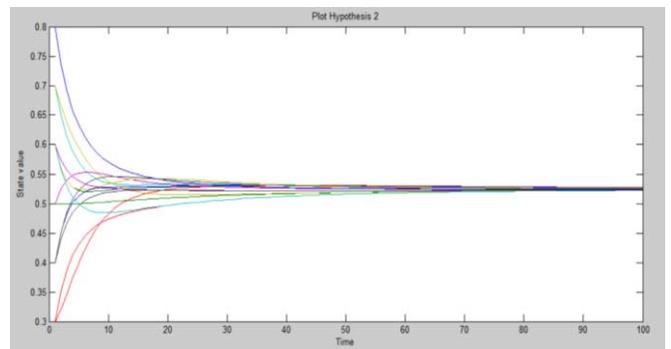


Figure 4. Experiment 1: changing levels of the states

In Fig. 5 the weights of the bridge connections between the two groups are shown which are expected to develop higher values. This is indeed the case; eventually they all converge to 1. This

happens because after time point 20 all levels differ less than 0.05, so both the homophily principle and the more becomes more principle have an increasing effect.

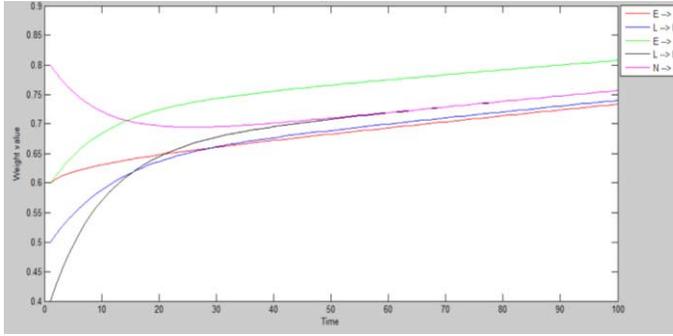


Figure 5. Experiment 1: changing bridge connection weights (between the nodes E, H, I, L and N)

4.3 Simulation Results for Experiment 2

In the second Experiment 1: changing bridge connection weights simulation experiment the two groups have their own opinion about sports. The first group, consisting of nodes A, B, C, D, E, F, G, H, considers basketball a really nice sport, therefore the initial levels of their states are high (values between 0.5 and 0.9). The second group, consisting of nodes I, J, K, L, M, N, have low initial levels as they do not like basketball (values between 0.1 and 0.4). One question was whether in some cases the bridge connections between nodes E, H, I and L will vanish over time due to the effect of homophily. Fig. 6 depicts the change of the levels of the nodes over time. As can be seen from the plot, the levels in both groups start of from the different initial values. Group 1 consists of the lines in the graph that start from 0.5 to 0.9 and group two consists of the lines starting below ranging from 0.1 to 0.4. The graph shows two things. First, there is an effect of increased clustering within each group (most in Group 2), but later the values of these clusters still converge to one value, around 0.53.

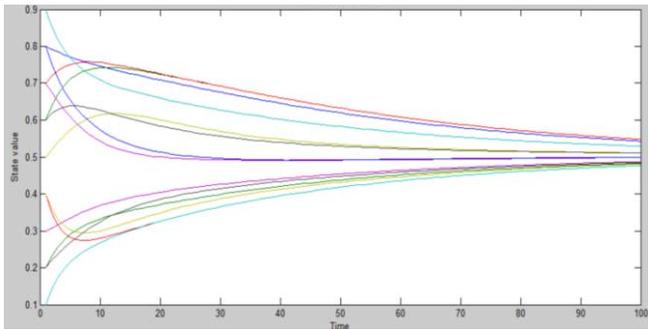


Figure 6. Experiment 2: changing levels of the states

So, in the longer term the bridge connections still make that one value is reached, which indicates that the bridge connections do not vanish. In Fig. 7 the weights of the bridge connections between the two groups are shown. A question was whether they would decrease to close to 0, due to the homophily principle. As can be seen from the plot, this is not the case. The connections between these nodes stay quite high and they are becoming higher over time. This can be explained by the more becomes more principle as follows. Note that Fig. 3 shows that the bridge connections all

involve quite popular nodes. Therefore, the more becomes more principle makes these connections stronger instead of weaker, as would the homophily principle do. Apparently the more becomes more principle dominates in this case, and makes that effects of the homophily principle do not emerge. When after some time the differences between the levels become less than the homophily threshold $\tau_{homophily} = 0.05$, even the homophily principle starts to contribute to the increase of the connection weights.

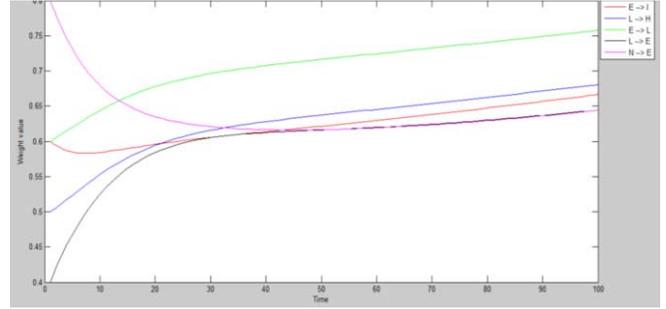


Figure 7. Experiment 2: changing bridge connection weights (between the nodes E, H, I, L and N)

5 VERIFICATION BY MATHEMATICAL ANALYSIS

This section presents some of the results of a mathematical analysis for the model. These results agree with the example simulations that have been performed, some of which are shown in Section 4. This provides verification of the model. First, a variable V indicating a state or connection weight is called *stationary* at t if $\mathbf{d}V(t)/\mathbf{d}t = 0$, it is *increasing* at t when $\mathbf{d}V(t)/\mathbf{d}t > 0$ and *decreasing* at t when $\mathbf{d}V(t)/\mathbf{d}t < 0$. For a connection weight $\omega_{A,B}$ being stationary means $\mathbf{d}\omega_{A,B}(t)/\mathbf{d}t = 0$; this is equivalent to:

$$\alpha \mathbf{c}_{\text{homo}, \omega_{A,B}}(X_A(t), X_B(t), \omega_{A,B}(t)) + (1-\alpha) \mathbf{c}_{\text{more}, \omega_{A,B}}(\omega_{C_{1,B}}(t), \dots, \omega_{C_{k,B}}(t)) = \omega_{A,B}(t)$$

which is modelled as

$$\alpha (\omega_{A,B}(t) + \omega_{A,B}(t) (1 - \omega_{A,B}(t)) (\tau_{A,B} - |X_A(t) - X_B(t)|)) + (1-\alpha) (\omega_{C_{1,B}}(t) + \dots + \omega_{C_{k,B}}(t))/k = \omega_{A,B}(t)$$

Similarly for states X_A . An *equilibrium* is when all states and connections are stationary at t . By rewriting the above formulae the following criteria can be derived for $\omega_{A,B}$ being stationary, increasing, and decreasing at time t .

Proposition 1 (Criteria for stationary, increasing, decreasing)

Stationary at t :

$$\alpha \omega_{A,B}(t) (1 - \omega_{A,B}(t)) (\tau_{A,B} - |X_A(t) - X_B(t)|) + (1-\alpha) [(\omega_{C_{1,B}}(t) + \dots + \omega_{C_{k,B}}(t))/k - \omega_{A,B}(t)] = 0$$

Increasing at t :

$$\alpha \omega_{A,B}(t) (1 - \omega_{A,B}(t)) (\tau_{A,B} - |X_A(t) - X_B(t)|) + (1-\alpha) * [(\omega_{C_{1,B}}(t) + \dots + \omega_{C_{k,B}}(t))/k - \omega_{A,B}(t)] > 0$$

Decreasing at t :

$$\alpha \omega_{A,B}(t) (1 - \omega_{A,B}(t)) (\tau_{A,B} - |X_A(t) - X_B(t)|) + (1-\alpha) * [(\omega_{C_{1,B}}(t) + \dots + \omega_{C_{k,B}}(t))/k - \omega_{A,B}(t)] < 0$$

Next, an equilibrium analysis will be made for the cases when $0 < \alpha < 1$. Later, in Proposition 4 the special case $\alpha = 0$ (more becomes more) will be addressed and in Proposition 5 the case $\alpha = 1$ (homophily). An equilibrium state can be considered for a connection weight $\omega_{A,B}$ for the three cases $\omega_{A,B} = 0$, $\omega_{A,B} = 1$ or $0 < \omega_{A,B} < 1$.

Proposition 2 (Equilibrium) Let values \underline{X}_A and $\underline{\omega}_{A,B}$ for all A and B for an equilibrium state be given. Then for all A and B the following hold:

- (a) If $\underline{\omega}_{A,B} = 0$ and $\alpha < 1$, then $\underline{\omega}_{C,B} = 0$ for all C connected to B
 (b) If $\underline{\omega}_{A,B} = 1$ and $\alpha < 1$, then $\underline{\omega}_{C,B} = 1$ for all C connected to B
 (c) If $0 < \underline{\omega}_{A,B} < 1$, and $\alpha > 0$ then

$$|\underline{X}_A - \underline{X}_B| = \tau_{A,B} + (1/\alpha - 1) * [(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k - \underline{\omega}_{A,B}] / [\underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B})]$$

Proof (a) From $\underline{\omega}_{A,B} = 0$ and $\alpha < 1$ it follows that $(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k = 0$, and since $0 \leq \underline{\omega}_{C,B} \leq 1$ for all C this entails $\underline{\omega}_{C,B} = 0$ for all C

(b) Similar to (a)

(c) This follows from:

$$\begin{aligned} \alpha * \underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B})(\tau_{A,B} - |\underline{X}_A - \underline{X}_B|) = \\ -(1-\alpha) * [(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k - \underline{\omega}_{A,B}] \\ \tau_{A,B} - |\underline{X}_A - \underline{X}_B| = \\ -(1-\alpha) * [(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k - \underline{\omega}_{A,B}] / [\alpha \underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B})] \\ |\underline{X}_A - \underline{X}_B| = \tau_{A,B} \\ + (1/\alpha - 1) * [(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k - \underline{\omega}_{A,B}] / [\underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B})] \quad \blacksquare \end{aligned}$$

More specifically, for an equilibrium in which all values X_A are the same the following is found.

Proposition 3 (Equilibrium with equal state values) Assume $0 < \alpha < 1$. Let for an equilibrium state values \underline{X}_A and $\underline{\omega}_{A,B}$ for all A and B be given, such that $\underline{X}_A = \underline{X}_B$ for all A and B . Then for each B the equilibrium values $\underline{\omega}_{C,B}$ for all C either are all equal to 0 or are all equal to 1:

$$\underline{\omega}_{C,B} = 0 \text{ for all } C \text{ or } \underline{\omega}_{C,B} = 1 \text{ for all } C$$

Moreover, all connection weights $\omega_{A,B} > 0$ are attracted to the equilibrium value $\underline{\omega}_{A,B} = 1$, and not to the value $\underline{\omega}_{A,B} = 0$.

Proof From $\underline{X}_A = \underline{X}_B$ it follows that

$$\underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B})(\tau_{A,B} - |\underline{X}_A - \underline{X}_B|) = \underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B}) \tau_{A,B} \geq 0$$

Take the smallest $\underline{\omega}_{A,B}$ for B . Then

$$(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k - \underline{\omega}_{A,B} \geq 0$$

Therefore since their sum is 0, both $\underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B}) \tau_{A,B}$ and $(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k - \underline{\omega}_{A,B}$ are 0:

$$\begin{aligned} \underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B}) = 0, \text{ which is equivalent to } \underline{\omega}_{A,B} = 0 \text{ or } \underline{\omega}_{A,B} = 1 \\ (\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k = \underline{\omega}_{A,B} \end{aligned}$$

By Proposition 2 if $\underline{\omega}_{A,B} = 0$, then all $\underline{\omega}_{C,B} = 0$, and if $\underline{\omega}_{A,B} = 1$, then all $\underline{\omega}_{C,B} = 1$. So, either $\underline{\omega}_{C,B} = 0$ for all C or $\underline{\omega}_{C,B} = 1$ for all C .

The case $\underline{\omega}_{C,B} = 0$ for all C is not attracting: if for one of the $\underline{\omega}_{C,B} > 0$ and $\underline{\omega}_{C,B} < 1$, then $(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k - \underline{\omega}_{A,B} \geq 0$ and $\underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B}) \tau_{A,B} > 0$, so it is increasing.

Some special cases have been excluded in parts of the above analysis: $\alpha = 0$ or $\alpha = 1$. When $\alpha = 0$ the model describes only the more becomes more principle. Then the equilibrium equations are:

$$\begin{aligned} \mathbf{c}_{\text{more}, \omega_{A,B}}(\underline{\omega}_{C_1,B}, \dots, \underline{\omega}_{C_k,B}) = \underline{\omega}_{A,B} \\ (\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k = \underline{\omega}_{A,B} \end{aligned}$$

For this case the following can be found.

Proposition 4 (More becomes more: $\alpha = 0$). Assume $\alpha = 0$ and let equilibrium values $\underline{\omega}_{C,B}$ be given. Then for all C and D the equilibrium values $\underline{\omega}_{C,B}$ and $\underline{\omega}_{D,B}$ are equal: $\underline{\omega}_{C,B} = \underline{\omega}_{D,B}$.

Proof Take the A such that $\underline{\omega}_{A,B}$ is the lowest from the $\underline{\omega}_{C,B}$. Then from

$$(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B})/k = \underline{\omega}_{A,B}$$

it follows

$$(\underline{\omega}_{C_1,B} + \dots + \underline{\omega}_{C_k,B}) - k \underline{\omega}_{A,B} = 0$$

$$(\underline{\omega}_{C_1,B} - \underline{\omega}_{A,B}) + \dots + (\underline{\omega}_{C_k,B} - \underline{\omega}_{A,B}) = 0$$

where each term $\underline{\omega}_{C_i,B} - \underline{\omega}_{A,B} \geq 0$. Therefore $\underline{\omega}_{C_i,B} = \underline{\omega}_{A,B}$ for all i . ■

Next, consider $\alpha = 1$, a model for only the homophily principle.

Then the equilibrium equation becomes

$$\mathbf{c}_{\text{homo}, \omega_{A,B}}(\underline{X}_A, \underline{X}_B, \underline{\omega}_{A,B}) = \underline{\omega}_{A,B}$$

$$\underline{\omega}_{A,B} + \underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B})(\tau_{A,B} - |\underline{X}_A - \underline{X}_B|) = \underline{\omega}_{A,B}$$

$$\underline{\omega}_{A,B}(1 - \underline{\omega}_{A,B})(\tau_{A,B} - |\underline{X}_A - \underline{X}_B|) = 0$$

The solutions of this equation are:

$$\underline{\omega}_{A,B} = 0 \text{ or } \underline{\omega}_{A,B} = 1 \text{ or } |\underline{X}_A - \underline{X}_B| = \tau_{A,B}$$

Proposition 5 (Homophily: $\alpha = 1$). Assume $\alpha = 1$ and let equilibrium values $\underline{\omega}_{A,B}$ be given. Then for all A and B it holds

$$\underline{\omega}_{A,B} = 0 \text{ or } \underline{\omega}_{A,B} = 1 \text{ or } |\underline{X}_A - \underline{X}_B| = \tau_{A,B}$$

6 VALIDATION FOR REAL WORLD DATA

For validation, data have been used from [8] for a large network of adolescents at three time points in subsequent years at which measurements have been done about how the state or opinion about alcohol drinking is and how their friendships are. In total, there were 160 participants, but only 129 were present at all three measurements. For the network structure, the participants were asked to name a maximum of six friends. They were also asked about their behaviour, for example, about their alcohol drinking behaviour, whether they were using drugs and their smoking behaviour. The following data files from the collection were used:

Friendship 1: About the relationships between subjects at time point 1

Friendship 2: About the relationships between subjects at time point 2

Friendship 3: About the relationships between subjects at time point 3

Alcohol: The alcohol drinking behaviour of every subject at the 3 time points

The model uses continuous values between 0 and 1. Therefore, the data points from the dataset were transformed into values that are comparable with data that will be predicted by the model. The score ‘best friend’ has been mapped on value 0.9, ‘just a friend’ on 0.5 and ‘no relation’ on 0.1. The values are chosen in this way partly because the function used to determine the next value of a connection weight ω contains the factor $\omega(1-\omega)$. That means that when ω is exactly 0 or 1, no change can occur. By using the above values, this will not be a problem.

In order to compare the empirical data with the model data used in this project, simulations were performed for 20 times steps. At $t = 0$ the initial values were taken from the first time point of the empirical data, in this way giving the model the same starting position as the empirical data. Then the simulated values at time point 10 and at time point 20 were compared to the second and third time point of the empirical data. To compare the simulated data to the empirical data the parameters have been tuned to this specific network by Simulated Annealing (e.g., [9]); For the Simulated Annealing as an error function, the sum of squares of deviations between simulated model values and empirical values has been used, in order to get a minimal error. The parameters used

and their ranges are shown in Table 4. It was found out that the speed factors $\eta_{weights}$ and η_{states} should not be too high for proper functioning of the model.

As mentioned, the Glasgow data [8] was collected for three time points in subsequent years. There are cases that at the first time point there was no reported relation between two subjects, but at time point 3 there was. However, this is not taken into account by the model used here. The model does not create new connections. An alternative option could be to assume by default a very low weight initially, but that option has not been chosen. Therefore, a selection of the Glasgow data was made: relations between two subjects who did not yet have a relation during the initial time point, have been ignored. By applying the Simulated Annealing parameter estimation method to the empirical data and the model data, the parameter values shown in the rightmost column in Table 4 were identified.

Table 4. Parameters, their ranges and the identified values

Parameter	Notation	Interval	Value
Update speed factor for states	$\eta_{weights}$	[0, 0.55]	0.2811
Update speed factor for connections	η_{states}	[0, 0.25]	0.2065
Fraction of homophily influence	α	[0, 1]	0.2556
Threshold for homophily principle	$\tau_{homophily}$	[0, 1]	0.1945

The pattern for the error, defined as the average of the squares of the deviations between the empirical data and simulated data, is shown in Fig. 8. The lowest value is 0.0035; the square root of this value is 0.0592. Considered as a measure for deviations within the [0, 1] interval where the values of connections and states vary, this means an average deviation of 6%, which is not bad as an approximation; this provides a positive validation result.

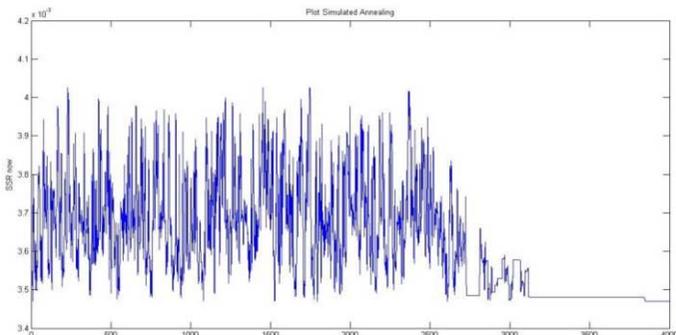


Figure 8. Simulated annealing: pattern of average of squares of deviations between empirical data and simulated data

In Fig. 9 for just two typical examples of connection weights it is shown how the dynamics of the model compares to the empirical values. Note that in Fig. 9 the (isolated) empirical values have been displayed with connecting lines between them; however, there is no linear relation, they are just measurements at different time points. Note that due to the discrete scores in the empirical data, real values can only be 0.1, 0.5 or 0.9. Here the real value of weight 1 stays 0.9 all the time, whereas the value of the model changes a bit, but deviates at most 0.02. The real value of weight 2 first stays 0.9 but then changes from 0.9 to 0.5 between the second and third time point. The value of the model also decreases, with deviations between 0.1 and 0.2.

Note that the identified value of the homophily influence fraction parameter α was around 0.25. This can be interpreted in

the sense that according to the model in this network the evolution of relations is determined for about 25% by the homophily principle based on similarity in alcohol drinking, and for about 75% by the more becomes more principle.

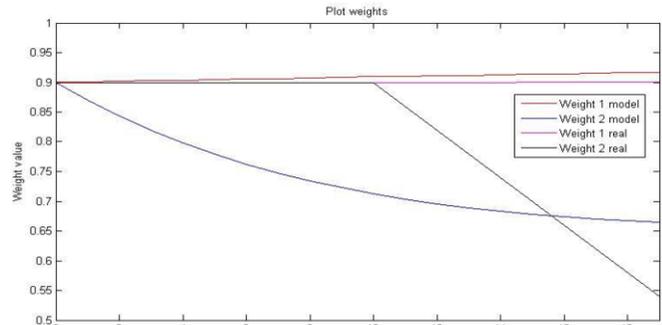


Figure 9. Two examples of the dynamics of connection weights in the model in comparison to the empirical values

7 DISCUSSION

The computational model for dynamics in social networks introduced in this paper was designed by integrating three models for the dynamics of states and relations of persons in a social network, addressing the *contagion* principle [5], the *homophily* principle [6, 12, 13, 19] and the *more-becomes-more* principle [2, 15], respectively. The model relates to and differs from existing work as follows. The first two models were adopted as variations on existing models from [5, 19], and the third model and the integration of the three models are new, as far as the authors know.

The integrated model was evaluated in three different manners: by various simulation experiments, by verification based on mathematical analysis, and by validation against an empirical data set. By all three methods a positive evaluation was found. The simulation results were as expected or at least were well explainable, the verification showed that the model provides the outcomes as predicted by the mathematical analysis, and the validation provided outcomes of the model with deviations from the empirical data within a 6% range of the [0, 1] interval. As part of the validation it was found that for the considered network of adolescents [8], according to the tuned model the dynamics of connections was determined by the homophily principle based on similarity in alcohol drinking for about 25%, and by the more becomes more principle for about 75%.

For future research variations of the model can be analysed. For example, for the homophily sub-model, as an alternative a quadratic variant as introduced in [19] can be used. Moreover, homophily with respect to multiple states can be explored, for example, not only similarities for drinking alcohol but also for smoking and sports behaviour. Also alternatives for the more becomes more and social contagion sub-models can be analysed, for example by choosing other combination functions from the collection of possible combination functions shown in [21].

Finally, validation can be performed for more empirical data on social networks and their dynamics over time. However, often data sets for social networks only provide data for one point in time and only about the connections. It is not easy to find data sets that include data about connections over time and in addition also data about states of the members of the network over time. In this sense the Glasgow data set [8] used here is very valuable and has a rather unique position.

REFERENCES

- [1] S. Aral, L. Muchnik, and A. Sundararajan, Distinguishing Influence Based Contagion From Homophily Driven Diffusion in Dynamic Networks. *Proceedings of the National Academy of Sciences (USA)*, **106**, 1544-1549, (2009).
- [2] A.L. Barabási, and R. Albert, Emergence of Scaling in Random Networks. *Science*, **286**, 509-512, (1999).
- [3] M. Bastian, S. Heymann, and M. Jacomy, *Gephi: an open source software for exploring and manipulating networks*. <https://gephi.org/>, 2009.
- [4] S. Boccaletti, V. Latorab, Y. Morenod, M. Chavez, and D.-U. Hwang, Complex networks: Structure and dynamics. *Physics Reports*, **424**, 175 – 308, (2006).
- [5] T. Bosse, R. Duell, Z.A. Memon, J. Treur, and C.N. van der Wal, Agent-Based Modelling of Emotion Contagion in Groups. *Cognitive Computation Journal*, **7**, 111-136, (2015).
- [6] D. Byrne, The attraction hypothesis: Do similar attitudes affect anything? *Journal of Personality and Social Psychology*, **51**, 1167-1170, (1986).
- [7] J. Giles, Computational Social Science: Making the Links. *Nature*, **488**, 448-450, (2012).
- [8] Glasgow Empirical Data https://www.stats.ox.ac.uk/~snijders/siena/Glasgow_data.htm, 2016
- [9] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, Optimization by Simulated Annealing, *Science, New Series*, **220**, 671-680, (1983).
- [10] B.J. Kuipers, Commonsense reasoning about causality: Deriving behavior from structure, *Artificial Intelligence*, **24**, 169-203, (1984).
- [11] B.J. Kuipers and J.P. Kassirer, How to discover a knowledge representation for causal reasoning by studying an expert physician. In: *Proceedings Eighth International Joint Conference on Artificial Intelligence, IJCAI'83*, Karlsruhe. William Kaufman, Los Altos, CA, 1983.
- [12] M. McPherson, L. Smith-Lovin, and J.M. Cook, Birds of a feather: homophily in social networks. *Annu. Rev. Sociol.*, **27**, 415-444, (2001).
- [13] A. Mislove, B. Viswanath, K.P. Gummadi, and P. Druschel, You Are Who You Know: Inferring User Profiles in Online Social Networks. *Proc. WSDM'10*, 2010, New York City, New York, USA, pp. 251-260, 2010.
- [14] M.P. Mundt, L. Mercken, and L.I. Zakletskaia, Peer selection and influence effects on adolescent alcohol use: a stochastic actor-based model. *BMC pediatrics*, **12**, 115, (2012).
- [15] M.E.J. Newman, The structure and function of complex networks. *SIAM Review*, **45**, 167-256, (2003).
- [16] H.V.D. Parunak, E. Downs, and A. Yinger, *Socially-Constrained Exogenously-Driven Opinion Dynamics*. *Proc. of the Fifth International IEEE Conference Self-Adaptive and Self-Organizing Systems (SASO'11)*, 2011.
- [17] J. Pearl, *Causality*. Cambridge University Press, 2000.
- [18] C.R. Shalizi, and A.C. Thomas, Homophily and Contagion are Generically Confounded in Observational Social Network Studies. *Sociological Methods & Research*, **40**, 211-239, (2011).
- [19] A. Sharpanskykh, and J. Treur, Modelling and Analysis of Social Contagion in Dynamic Networks. *Neurocomputing*, **146**, 140-150, (2014).
- [20] C.E.G. Steglich, T.A.B. Snijders, and M. Pearson, Dynamic networks and behavior: separating selection from influence. *Sociol Methodol.*, **40**, 329-393, (2010).
- [21] J. Treur, Dynamic Modelling Based on a Temporal/Causal Network Modelling Approach. *Biologically Inspired Cognitive Architectures Journal*, **16**, 131-168, (2016).
- [22] J. Treur, *Network-Oriented Modelling: Addressing Complexity of Cognitive, Affective and Social Interactions*. Understanding Complex Systems Series, Springer Publishing, 2016.
- [23] T.W. Valente, *Social Networks and Health: Models, Methods, and Applications*. New York, Oxford University Press, 2010.

Exact Particle Filter Modularization Improves Runtime Performance

Padraic D. Edgington¹ and Anthony S. Maida²

Abstract. Bayesian filters provide a robust and powerful technique for integrating noisy information in dynamic environments. However, the computational cost of the filtering algorithm depends on the size of the problem, and an effective solution may be constrained by execution time. This paper applies basic concepts of clustering and message passing to particle filters making them substantially faster to compute, while still maintaining the original accuracy. An example from vehicle state estimation is provided to illustrate how to implement the technique. Our results indicate that modularization can produce a speed up of over 28 times even on this small problem.

1 Introduction

Bayesian filters, such as the Kalman filter, offer a robust method for addressing problems that can be modeled with a dynamic Bayesian network where we are only interested in the current state [7, 8, 12, 23, 22, 9, 5, 6]. These types of problems tend to have sensors that provide noisy observations about the state of a dynamic environment. In general, the goal is to continuously monitor the state of a set of variables, which are indirectly related to the sensor measurements. However, the computational cost of the filtering algorithm is dependent on the size of the problem and an effective solution may be constrained by execution time. This paper introduces a new technique which reorganizes a Bayesian filter into a set of modules. Each module addresses a conditionally independent subset of the original state variables. Each module uses a single Bayesian filter to process its set of state variables and passes the results to any neighboring modules to distribute the sensor information throughout the network. The primary benefit of this modularization lies in reducing the execution time by reducing the size of individual problems. Specifically, it reduces the problem of supralinear growth in time complexity as a function of problem size. This also has the side effect of requiring less memory to represent the smaller problems.

While Bayesian filters provide effective solutions for dynamic Bayesian networks (DBNs) with continuous variables, there are several other areas of research that are closely related [14]. Three examples are dynamic Bayesian networks with discrete variables, hidden Markov models (HMMs) and Bayesian networks. Dynamic Bayesian networks with discrete variables generally use different techniques than if they only had continuous variables, instead emphasizing building and parsing tables of conditional probabilities. Hidden Markov models focus on a frequentist view of probability rather than a causal model with noise. Bayesian networks are generally simpler forms of dynamic Bayesian networks: they generally

have discrete variables, and their environment does not change as time elapses.

Each of these areas of research are sufficiently similar mathematically that they have enjoyed fertile cross-pollination in recent years. Many popular techniques are derived from Pearl's [19] algorithms for exact computation in Bayesian networks. Loopy belief propagation (LBP) takes Pearl's belief propagation algorithm for singly connected networks and applies it to densely connected networks for both Bayesian networks and dynamic Bayesian networks [16, 15]. Other techniques, such as the Boyen-Koller (B-K) method [2, 3] have found their way from dynamic Bayesian networks to Bayesian networks and Hidden Markov models. Particle filters have been proposed under many names as solutions to many types problems and have been adopted into even more [22, 4, 5, 14, 17, 6].

The work solving these types of problems has fallen into two categories: exact inference methods and approximate inference methods. Pearl's [19] original work fell into the category of exact methods and some research has since been performed to improve upon those algorithms [11, 20, 24]. However, the majority of the research that has been done since then—the B-K method [2, 3], loopy belief propagation [16, 15], the factored frontier algorithm [15], assumed density filtering [13], thin junction tree filtering [18] and particle filtering [22, 4, 5, 14, 17, 6]—has focused on approximate inference.

Several approaches have been developed for improving the execution time of inference in Bayesian networks [19, 11, 24]. While none of those methods are directly applicable to Bayesian filters, the concepts of clustering from poly-tree methods ([11, 19] see also [1, 10, 21]) as well as using message-passing to perform local computations [19] can be applied to this problem as well. Thus, we will be creating statistically independent clusters that can interact with each other.

Many of these algorithms explore similar concepts—primarily clustering and message passing—and the present work is no different. Modularization fundamentally breaks a filtering problem into a series of independent clusters and passes information between them. Thus, modularization brings the clustering concepts popular in traditional inference algorithms and Pearl's message passing concepts to filtering. This paper focuses on replicating the results of a particle filter without adding any approximation. At the end, we will see how the modularized version of the algorithm compares to a traditional particle filter with the same parameters.

2 Modularizing Dynamic Bayesian Networks

This paper applies our modularization technique to particle filters. The modularization focuses on taking an existing dynamic Bayesian network (DBN) (e.g. Figure 1) and decomposing it into smaller mod-

¹ University of Connecticut, email: padraic@engr.uconn.edu

² University of Louisiana at Lafayette, email: maida@cacs.louisiana.edu

ules that are easier to solve than the original problem (e.g. Figure 2).

By *modularized*, we mean that the nodes representing model state variables that compose a Markovian dynamic Bayesian network are partitioned into subsets or clusters each forming an independent filter. Nodes in each cluster are generally correlated with each other, but are independent of nodes in other clusters. This makes it possible to assign a separate Bayesian filter to each cluster. Compared with a monolithic filter, module filters are smaller and can be targeted to the dynamics of each particular cluster. By *exact*, we mean that the modularized Bayesian network—with collective lower run time overhead across the modularized filters—preserves the accuracy of the original solution that used a monolithic filter for the entire network. In this work, we will use *Bayesian filters* to refer to the entire class of filtering algorithms, such as Kalman filters and extended Kalman filters (collectively termed *Gaussian filters*), as well as particle filters.

The primary benefit from modularization lies in the computation time. Since the execution time for a Bayesian filter is super-linearly dependent on the size of either the state variables or the information sources, reducing the number of elements that are processed at any one time will improve the overall execution time. While there is some overhead cost from reassembling the modular results, the final cost will generally be lower than the non-modularized execution time. The speedup of the resulting modular algorithm depends on the size and complexity of the original DBN. Large problems have more potential for improvement and the speedup is correlated with the number of modules produced; however, some improvement can be seen even on simple problems.

In modularizing a given DBN, we will create a set of modules, with each module designed to calculate a disjoint subset of the state variables. Each of these modules will have a separate Bayesian filter to calculate the probability distribution for the state variables contained in that cluster. The relationships between information sources and the modules will remain the same. However, since information sources will only be directly related to a single module, the computation for incorporating the belief about an information source can be limited to the related module. To ensure that the information is well distributed throughout the network, message passing is implemented to allow modules to act as information sources for each other.

As a side effect, modularization also has the benefit of simplifying much of the conceptual and mathematical structure of a Bayesian filter. Instead of having to understand or calculate the complex relationships between distantly related elements in the network, much of the work will be handled locally by the direct relationships. Thus, the message passing mechanism will alleviate the need to calculate all of the distant relationships; instead it uses the local relationships when passing information between modules. Since this information passing is cumulative, long distance relationships are effectively calculated through function composition.

3 Building a Modularized Bayesian Filter

The first step in creating a modularized Bayesian filter is to group the nodes (state variables) into conditionally independent clusters. For the purposes of this paper, we will assume that this can be done. For a small problem, it is possible to do this by hand, but for a larger problem, we will want to use a clustering algorithm. As can be seen in Figure 2, each cluster is present in every time step and they serve to isolate their component nodes from other information sources in the network.

Once the dynamic Bayesian network has been modularized, work can begin on constructing the modularized Bayesian filter to solve the

problem. Constructing a filter has two major steps. First, filters will be selected to process each cluster that was generated by clustering. Once these filters have been defined, the message passing structures that allow information to be propagated can be described.

3.1 Selecting Bayesian Filters

Selecting Bayesian filters for a modular dynamic Bayesian network is quite similar to selecting a Bayesian filter for a regular dynamic Bayesian network. The primary difference is that instead of selecting a single filter to handle all of the calculations of the network, one filter will be selected for each cluster in the modular dynamic Bayesian network.

This set of filters should be selected by the same criteria that are used traditionally. In this case, instead of looking at the problem as a whole, each cluster is examined independently of the rest of the network. The only state variables that are of importance are those within the cluster currently being examined. All of the adjacent nodes in the graph can be considered as if they were static information sources, regardless of whether they consist of a traditional information source or another node in the original graph.

Thus, standard criteria such as linearity and the shape of the probability distribution are only considered within the scope of a single cluster at a time. That is, a modularized filter can be decomposed into a set of heterogeneous Bayesian filters. One caveat to this simplicity is in the case where a cluster with a simple probability distribution is adjacent to a cluster with a complex probability distribution. While it is possible to manipulate the complex probability distribution so that it can be related to the simple distribution, in some cases it may be preferable to use a more complicated Bayesian filter for the simple cluster to make the comparisons easier.

3.2 Message Passing

As stated earlier, the modularized filters communicate by message passing. Message passing is—for the most part—a straightforward process: messages will go in both directions, from a causal cluster to a resultant cluster and vice-versa. Message passing utilizes the existing Bayesian filter algorithms for most of the work. Causal messages improve the update information that is passed to a Bayesian filter. Resultant messages act as an additional observation to be incorporated. The major complication comes when information that was passed causally may be returned in a resultant message.

Often causal links in a DBN indicate how a variable changes based on the state of another variable. This is encapsulated in the notion of updating the previous state with the changes that have occurred since the last update. In this case, message passing works to provide a more accurate update command for the receiving filter. The other possibility is that the receiving filter only uses the information to set the state of the variables. In this case, the message would not be used to modify the previous state of the variables, but as the predicted value for those variables. The basic difference between a traditional Bayesian filter and a modular one here is that the state variables in the causal cluster will be updated with the available information before being used as a control action by the receiving filter.

In the case of resultant messages, it is expected that the receiving filter has already predicted the state of the variables for this cycle. As such, the message should be framed in terms of a measurement of the state variables. Since the message is being viewed as a measurement, it can be processed as such by the receiving Bayesian filter. This means running just the measurement section of the receiving

Bayesian filter to integrate the message with the existing probabilistic belief.

It is hard to overstate the simplicity of the basic message passing method, as it primarily involves passing the results of one Bayesian filter to a related Bayesian filter using the existing relations defined by the equations underlying the Bayesian network. This simplicity will be shown more explicitly in our example. However, the message passing method has one complication: there is the potential for information to be duplicated.

Since problems will require passing information in both directions along graph edges, we should expect that information which is passed as a causal message would also be returned as part of a resultant message. To solve this problem, a novel inverse Bayesian filter is used to remove the information in the causal message from the information in the resultant message, thus only passing novel information to the causal module. This can be seen in [Figure 3](#) with precise descriptions of the messages being passed. Since Bayesian filters work in a fixed manner, it is easy to work backwards through a filter to retrieve a piece of information given the other pieces. This method will also ensure a relatively fixed cost for the operation that is independent of the number of information sources that the resultant module has incorporated into its state variables.

3.3 Non-Standard Information Representations

The simplest information sources are represented as Gaussian functions when they are used as parameters for a Bayesian filter. Since message passing is a major part of modularized Bayesian filters, it needs to be possible to pass the state of a particle filter as a message to other filters.

There are two potential problems with using anything besides a unimodal Gaussian function as an information source for other Bayesian filters. The first—and most obvious—is that existing Bayesian filters have all been designed to process Gaussian functions as their information sources. Thus, either the probability distribution must be represented as a Gaussian function or the Bayesian filter needs to be modified to handle the complex probability distribution directly. The second problem is more subtle: complex probability representations are often used to represent complex probability distributions. Attempting to shoehorn them into a filter designed to handle simple representations is likely to require some approximation. In the case that approximation is needed, it is beneficial to minimize the amount of error added, while still being mindful of the execution time of the approximation methods.

In general, when using a complex probability representation in a simpler Bayesian filter, the probability representation will need to be converted to something that the filter can process. An example of this is shown in [Section 3.3.1](#) in the form of converting particle sets to Gaussian distributions for Kalman filters and extended Kalman filters. In contrast, if the receiving Bayesian filter is sufficiently robust, then it may be possible to modify the filter to accept the complex probability representation directly. An example of this is shown in [Section 3.3.2](#) where the particle filter will be modified to use particle sets as updates and measurements. Conceptually, these ideas should generalize well to other complex probability representations such as mixtures of Gaussians or kernel functions.

3.3.1 Particle Sets as Information Sources for Kalman-Filter Based Bayesian Filters

Particle sets have traditionally been limited to representing the probability density function for the state variables in a particle filter [5, 6, 22]. Since the Kalman filter and extended Kalman filter perform operations on the Gaussian distributions directly, modifying the algorithm would be overly complicated. Instead, it is easier to convert a particle set to a Gaussian distribution.

The simplest method involves calculating the mean and covariance of the particle set and using it as the parameter for the Bayesian filter. The equations are similar whether calculating the Gaussian form for an unweighted particle set or a weighted particle set. Equations (1) and (2) show the formulas to calculate the mean and covariance for an unweighted particle set. Equations (3) and (4) show the formulas to calculate the mean and covariance for a weighted particle set. In both cases the equations evaluate all n particles in the set \mathbf{x} . In general, the individual particles, \mathbf{x}_i s, are composed of multiple state variables and thus are processed as a vector.

This method can work quite well for many problems. However, this is the simplest method for converting a particle set to a Gaussian representation. For distributions that are more complex, alternative manipulations may be required. Additionally, for the more complicated variants of the Kalman filter, converting the particle set to a mixture of Gaussians or a kernel function can be used to provide a more accurate approximation of the particle set.

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \quad (1)$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \hat{\boldsymbol{\mu}}) (\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T \quad (2)$$

$$\hat{\boldsymbol{\mu}} = \frac{1}{\left(\sum_{i=1}^n w_i\right)} \sum_{i=1}^n w_i \mathbf{x}_i \quad (3)$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{\left(\frac{n-1}{n} \sum_{i=1}^n w_i\right)} \sum_{i=1}^n w_i (\mathbf{x}_i - \hat{\boldsymbol{\mu}}) (\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T \quad (4)$$

3.3.2 Particle Sets as Information Sources for Particle Filters

Using particle sets as information sources for particle filters provides a more interesting case to consider. Since the particle filter itself is quite robust, it is actually possible to use the particle sets as information sources directly. There is some additional complexity in that the methods for using a particle set as the control action will differ from the methods for using a particle set as an observation. However, both of these cases provide an interesting look at how complex probability representations can be manipulated within a Bayesian filter.

The first information source that a particle filter uses is the control action, which represents the belief about how the state of the environment has changed since the previous time step. Traditionally, the algorithm would take samples from the provided Gaussian distribution and apply them to selected particles from the previous time step to arrive at a prediction about the current state of the environment. Thus, the goal in modifying the particle filter to use a particle set as the representation of the update command involves finding a way to sample from the particle set. Those samples can then be used to

update selected particles representing possible previous states of the environment.

A particle set represents the true probability distribution as a set of particles. If the true probability density function is well represented by the set of particles, then it is sufficient to sample from the particle set. If the particle set is unweighted, then simply selecting particles randomly with equal weight is sufficient to provide a high quality sampling function for a particle set. However, if the particle set is weighted, then a discrete pseudo-random number generator is used to select particles with probability proportional to their weights. Both of these techniques are generally easy to implement, as they are already part of handling the main particle set in a particle filter.

Unfortunately, particle sets are not well suited for grading other particles. While the weights on weighted particles are readily accessible to a particle filter, the distribution described by the density of the particle set is not readily accessible. As such, using a particle set as the information provided as an observation is similar to using a particle set with a Gaussian filter. The main difference in working with a particle filter is that the particle set does not need to be converted to a specific type of probability representation. Instead, any representation that can quickly produce an estimation of the full distribution and then be used to grade particles will be effective for this task.

However, there is one case—which is of particular interest in modularized Bayesian filters—in which particle sets can be used by a particle filter directly. In modularized Bayesian filters, messages will often be passed circularly. In these cases, the information passed as the update command needs to be removed from the particle set. Normally, this would be done using an inverse Bayesian filter; however, in this case the inverse filter only needs to ignore the information about the distribution of the particles. The weights on the individual particles can be readily used as the probability of that particle based on other observations. As such, a particle can be graded simply by applying the weight of the relevant part of the particle set. The exact weight applied can be obtained by any of a number of simple techniques, such as interpolation, or just selecting the nearest point in the particle set and using its weight directly.

4 Computation Time for Modularized Bayesian Filters

This section will examine a few simple cases for particle filters that will show how the algorithm is capable of scaling for large problems. The actual speedup obtained from applying these techniques depends on the specific problem being addressed. There are two types of particle filters: one that uses a fixed number of particles and one that varies the number of particles based on the complexity of the probability distribution. The original particle filter, which uses a fixed number of particles at each time step, has a fixed execution time based on that number of particles. Since the number of particles produced is chosen by hand, looking at this algorithm is not terribly interesting. Instead, the KLD sampling variant provides a more robust and generalizable view of the cost of using a particle filter.

In the KLD sampling particle filter, the Kullback-Leibler divergence ensures that the generated probability distribution represents the true distribution within a specified tolerance. As long as the accuracy of the information provided to the KLD sampling particle filter is constant, the number of particles generated remains reasonably constant between iterations of the algorithm. For a single particle filter, the time complexity is $O(p^n)$; where p is the number of particles required to represent a single dimension of the state space adequately

and n is the number of dimensions in the state space. For this example, we will assume that p is constant across dimensions, though it does not have to be. As an illustration, if the particle set was representing a unimodal distribution, the particle distribution could be seen as forming a hyper-ellipse, where the number of particles would roughly correspond to the volume of the hyper-ellipse.

Dividing the problem into a set of m equally sized modules produces a time complexity of $O(mp^{n/m})$ to calculate all of the modules. The time complexity of the inverse particle filter is based on the method used to obtain a weight for each particle of interest. For this example, we will assume that a k -d tree is used to select the nearest particle. k -d trees are efficient, having a time complexity of $O(n \log n)$ to create the tree and $O(\log n)$ to find the nearest element in the tree, where n is the number of particles in the set. For this example, that translates into a time complexity of $O(p^{n/m} \log p^{n/m})$ to construct each k -d tree and $O(p^{n/m} \log p^{n/m})$ to grade all the particles in a set. Summing these together yields a full time complexity of

$$\begin{aligned} O\left(mp^{n/m} + 2(m-1)p^{n/m} \log p^{n/m}\right) \\ = O\left(mp^{n/m} \log p^{n/m}\right). \end{aligned} \quad (5)$$

For additional clarification, let us consider two test cases: in the first case, there will be two modules with state variables evenly distributed between them; in the second case, there will be n modules, each processing one state variable. In the first case, substituting $m = 2$ into (5) produces a time complexity of $O(2p^{n/2} \log p^{n/2}) = O(p^{n/2} \log p^{n/2})$, which will clearly provide a healthy speedup as compared to the original $O(p^n)$. In the second case, $m = n$ is substituted into (5). This produces the more attractive result of $O(np^{n/n} \log p^{n/n}) = O(np \log p)$, which can be further reduced to $O(n)$.

For a particle filter, the effects on memory usage are the same as with time complexity. Since the number of particles generated directly influences the computation time and the amount of space required to hold the result, reducing the number of particles affects both complexities the same way.

5 Vehicle State Estimation Example

Next, we will look at a complete example of building a modularized Bayesian filter. This example considers a hypothetical automobile moving in a planar environment. It respects the basic physical concepts of motion, but ignores more complex aspects such as slip and drag. The formulation of the problem provides a set of equations relating acceleration, velocity and position variables. These equations can then be used to create a dynamic Bayesian network that can be clustered. Bayesian filters will then be applied to each cluster and their interactions can be described explicitly.

The vehicle described here operates in a two-dimensional plane, so it needs an x and y variable to represent its position in the plane. Since it is a non-holonomic vehicle, its orientation also affects the motion of the vehicle, so an angular pose variable θ is also necessary. This example is modeled in terms of position, velocity and acceleration so a set of three variables are needed for each level. This produces nine state variables to estimate. The equations of motion

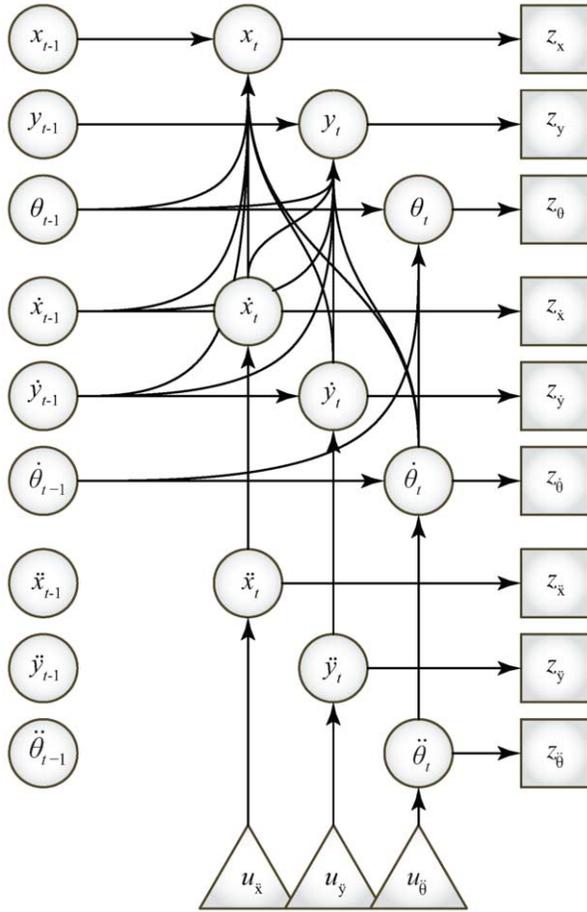


Figure 1. The initial dynamic Bayesian network for the vehicle state estimation example. Circles indicate state variables, triangles denote control values, and squares indicate observations or measurements.

for this vehicle are as follows:

$$\dot{x}_t = \dot{x}_{t-1} + \ddot{x}_t \Delta t \quad (6)$$

$$\dot{y}_t = \dot{y}_{t-1} + \ddot{y}_t \Delta t \quad (7)$$

$$\dot{\theta}_t = \dot{\theta}_{t-1} + \ddot{\theta}_t \Delta t \quad (8)$$

$$v_m = \frac{1}{2} \left(\sqrt{\dot{x}_{t-1}^2 + \dot{y}_{t-1}^2} + \sqrt{\dot{x}_t^2 + \dot{y}_t^2} \right) \quad (9)$$

$$v_\theta = \frac{1}{2} (\dot{\theta}_{t-1} + \dot{\theta}_t) \quad (10)$$

$$x_t = x_{t-1} - \frac{v_m}{v_\theta} \sin(\theta_{t-1}) + \frac{v_m}{v_\theta} \sin(\theta_{t-1} + v_\theta \Delta t) \quad (11)$$

$$y_t = y_{t-1} + \frac{v_m}{v_\theta} \cos(\theta_{t-1}) - \frac{v_m}{v_\theta} \cos(\theta_{t-1} + v_\theta \Delta t) \quad (12)$$

$$\theta_t = \theta_{t-1} + v_\theta \Delta t \quad (13)$$

These equations can then be used to define a dynamic Bayesian network. With the addition of a set of singly connected control (u) and measurement (z) nodes, the graph will look like [Figure 1](#).

This problem can be separated into two conditionally independent clusters. The first cluster is comprised of the position and velocity nodes, with the acceleration nodes in the second cluster. This clustered network is shown in [Figure 2](#). By collapsing all of the related nodes and focusing on the information flow, we arrive at [Figure 3](#). The graph shows the two clusters as a pair of circles. Both of the clusters have a set of related measurement variables. However, the con-

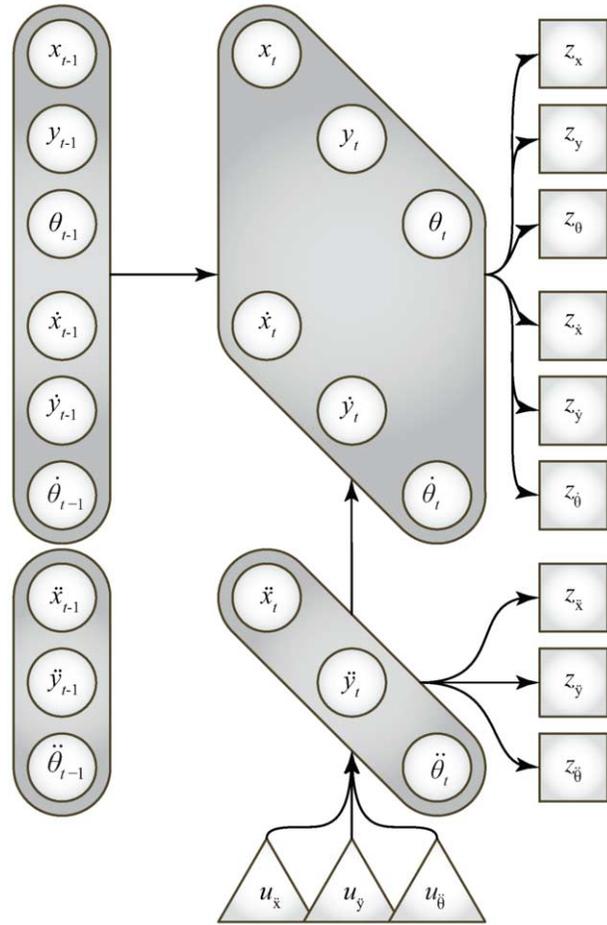


Figure 2. The clustered dynamic Bayesian network for the vehicle state estimation example. The original problem with 9 state variables reduced to subproblems of size 3 and 6.

trol variables only affect the acceleration cluster directly. The results from the acceleration cluster will be fed to the position/velocity cluster. The results from the position/velocity cluster are also fed back to the acceleration cluster via an inverse Bayesian filter. Since the inverse filter will be decomposing the results of the position/velocity cluster, it requires several of the position/velocity filter's parameters as well.

Most of the relationships in [Figure 3](#) are directly derived from [Figure 2](#), where the clusters and their relationships to the other parts of the graph are fully defined. These relationships have been carried over into [Figure 3](#). The addition here is in the information flow. The basic flow of causality to and from the information sources is unchanged as indicated by the thin-bodied arrows. However, in order for the information in these elements to be applied to all of the state variables in the graph, the information must be able to pass between clusters.

Since the state of the acceleration cluster causes the position and velocity variables to change, the state of the acceleration cluster is passed directly to the position/velocity cluster as its update. This allows the information in the external update (control) command and the information in the acceleration sensors to be applied to the position and velocity state variables. Since we also want the information from the position and velocity sensors to be applied to the acceleration calculations, this information also needs to be passed from the position/velocity cluster to the acceleration cluster. How-

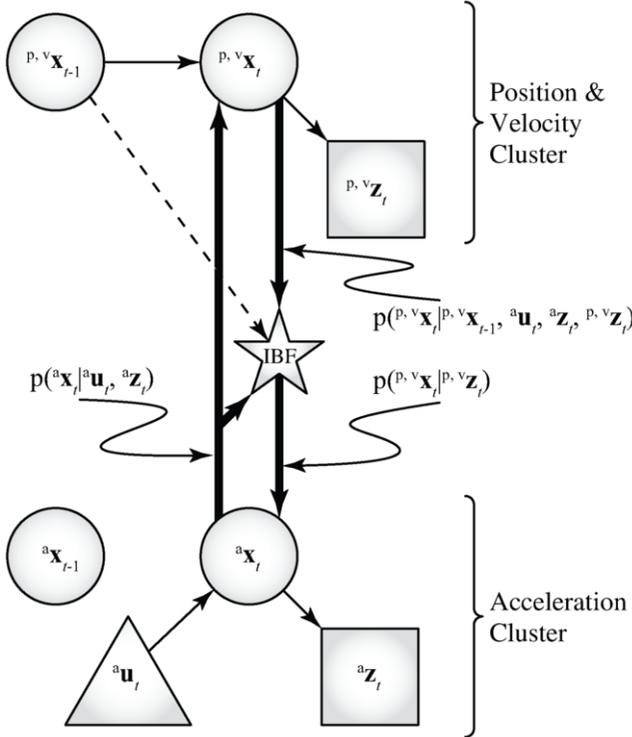


Figure 3. The clustered dynamic Bayesian network with information flow made explicit. Within cluster nodes are collapsed. The star-shaped node denotes an inverse filter.

ever, since the position and velocity calculations have already incorporated the information about the acceleration, the results of the position/velocity cluster cannot be passed back to the acceleration cluster directly. Instead, the current state of both the acceleration state variables, as well as the position and velocity state variables are passed to an inverse Bayesian filter, which can separate the acceleration information from the position and velocity information, allowing the remaining position and velocity measurements to be used to update the acceleration information.

5.1 Selecting Bayesian Filters

Having defined two clusters for the problem, the next step is to select a Bayesian filter for each individual cluster. Since the individual clusters do not need to employ the same filters, we can select a Bayesian filter that is appropriate for each cluster separately. As defined in Equations 6–8, the three acceleration nodes have a simple linear relationship with the velocity nodes. Since the control and measurement nodes are directly related for this example, the acceleration cluster is well suited to a simple Kalman filter. The position and velocity cluster on the other hand has several non-linear relationships between its nodes and other nodes in the graph. As such, it would be better suited to an extended Kalman filter (EKF) or particle filter. In this paper, we will show how this case unfolds with a particle filter.

5.2 Assembling the Modular Particle Filter Algorithm

These concepts can now be translated into a modularized Bayesian filter algorithm. The algorithm for the modular particle filter is shown

Algorithm 1 A modular particle filter algorithm for the vehicle state estimation example.

```

1: procedure MODULAR PARTICLE FILTER(...)
2:    $[\hat{\mu}_t \ \hat{\Sigma}_t] \leftarrow \text{KF} \left( \begin{matrix} a\mu_{t-1}, a\Sigma_{t-1}, \mathbf{u}_t, \mathbf{M}_t, a\mathbf{z}_t, \\ a\mathbf{Q}_t, a\mathbf{A}_t, a\mathbf{B}_t, a\mathbf{C}_t \end{matrix} \right)$ 
3:    ${}^{P,V}P_t \leftarrow \text{PARTICLE FILTER} \left( \begin{matrix} {}^{P,V}P_{t-1}, [a\hat{\mu}_t \ a\hat{\Sigma}_t], \\ [{}^{P,V}z_t \ {}^{P,V}Q_t] \end{matrix} \right)$ 
4:    $[\hat{\mathbf{z}}_{\text{IBF}} \ \hat{\mathbf{Q}}_{\text{IBF}}] \leftarrow \text{INVERSE PARTICLE FILTER}({}^{P,V}P_t)$ 
5:    $[\hat{\mu}_{t-1} \ \hat{\Sigma}_{t-1}] \leftarrow \text{PS2GAUSSIAN}({}^{P,V}P_{t-1})$ 
6:    $\mathbf{z}_{\text{IBF}} \leftarrow \text{PREDICTION}^{-1}(\hat{\mu}_{t-1}, \hat{\mathbf{z}}_{\text{IBF}})$ 
7:    $\mathbf{Q}_{\text{IBF}} \leftarrow \text{PREDICTION}^{-1}(\hat{\mathbf{Q}}_{\text{IBF}})$ 
8:    $[a\mu_t \ a\Sigma_t] \leftarrow \text{KF}_m(a\hat{\mu}_t, a\hat{\Sigma}_t, \mathbf{z}_{\text{IBF}}, \mathbf{Q}_{\text{IBF}}, \mathbf{I})$ 
9:   return  $[{}^{P,V}P_t \ a\mu_t \ a\Sigma_t]$ 
10: end procedure

```

Algorithm 2 An inverse particle filter algorithm for the vehicle state estimation example.

```

1: procedure INVERSE PARTICLE FILTER( $P$ )
2:    $[\mu_t \ \Sigma_t] \leftarrow \text{WEIGHTED PS2GAUSSIAN}(P)$ 
3:    $[\psi_t \ \Psi_t] \leftarrow \text{UNWEIGHTED PS2GAUSSIAN}(P)$ 
4:    $\mathbf{K}_t^{-1} \leftarrow \Psi_t(\Psi_t - \Sigma_t)^{-1}$ 
5:    $\mathbf{Q}_t \leftarrow \mathbf{K}_t^{-1}\Sigma_t$ 
6:    $\mathbf{z}_t \leftarrow \mathbf{K}_t^{-1}(\mu_t - \psi_t) + \psi_t$ 
7:   return  $[\mathbf{z}_t \ \mathbf{Q}_t]$ 
8: end procedure

```

in Algorithm 1. The modular particle filter largely serves as a meta-algorithm to call other algorithms based on the required parameters.

\mathbf{x}	a vector
\mathbf{X}	a matrix
$f(\cdot)$	a function
KF	Kalman filter
a_x	an element related to the acceleration cluster
${}^{P,V}x$	an element related to the position and velocity cluster
x_{IBF}	an element related to the inverse Bayesian filter
\hat{x}	a temporary estimate of an element

Table 1. List of symbols used.

Line 2 begins by using a Kalman filter to integrate the previous acceleration state ($a\mu_{t-1}, a\Sigma_{t-1}$) with the update information ($\mathbf{u}_t, \mathbf{M}_t$) and the acceleration measurements ($a\mathbf{z}_t, a\mathbf{Q}_t$). For consistency, the previous acceleration state and the \mathbf{A}_t matrix are both passed to the Kalman filter. In this example, the \mathbf{A}_t matrix is also the zero matrix, so these variables do not actually contribute anything to the calculations. As such, a tightly optimized algorithm would omit these.

Since the update commands are only related to the acceleration variables, there is no need to qualify any of these variables with the cluster name. This is contrasted with the measurements, where only the variables related to acceleration are used directly in the Kalman filter. The result of the Kalman filter is another Gaussian distribution reflecting the belief about the current state of the vehicle's acceleration ($a\hat{\mu}_t, a\hat{\Sigma}_t$).

Line 3 uses a particle filter to integrate the previous state of the position and velocity variables (${}^{P,V}P_{t-1}$) with the control action ($a\hat{\mu}_t, a\hat{\Sigma}_t$) and the related sensor measurements (${}^{P,V}z_t, {}^{P,V}Q_t$). The difference between a regular particle filter and the one used here, is that

it uses the results from line 2 as the control action, instead of something defined externally. The result of the particle filter is a particle set that contains all of the available information about the position and velocity variables (${}^{P,V}P_t$).

Line 4 calls an inverse particle filter to decompose the particle set generated in line 3. If we assume that the particle filter produces a weighted particle set, then the decomposition is simple. If the particle filter returns an unweighted particle set, then the prediction needs to be calculated directly. Since this is a simple problem, the particle set is just converted into a pair of Gaussian distributions.

Algorithm 2 shows the algorithm used for the inverse particle filter for this example. This allows the prediction distribution to be removed from the result distribution. Line 2 converts the particle set into a Gaussian distribution taking the particle weights into account. This produces a Gaussian representation of the current state distribution (μ_t, Σ_t). Line 3 extracts the predicted state information ($\psi_t \Psi_t$) by not using the weight information. Lines 4–6 calculate the change in the distribution as though an EKF had been used. Line 4 calculates the inverse Kalman gain. Lines 5 and 6 then remove the predicted state from the final state to calculate the information that was added by the particle filter algorithm. The measurement information is left in terms of the position and state velocity variables, which makes it relatable to the acceleration state variables.

Back in the modular particle filter, the goal of lines 5–7 is to transform the measurement from the velocity space to a space that the Kalman filter can access. There is a linear relationship between the acceleration and the velocity, which enables this transformation to be applied easily. In some complicated problems, it may be easier to move this transformation into the inverse particle filter so that it can be addressed at the same time. Furthermore, a couple of simplifications could be made for this problem. Since the velocity variables are the only ones directly related to the acceleration variables, line 5 of the modular particle filter and lines 2 and 3 of the IPF algorithm could be modified to only calculate the Gaussian distribution for the velocity variables and ignore the position state variables. In addition, since the inverse particle filter calculates a Gaussian distribution for the particle set in line 2, this result could be held to be used directly in line 5 of the modular particle filter instead of recalculating it.

The measurement part of the Kalman filter used in line 8 begins with the assumption that there is an existing distribution about the acceleration state variables that merely needs a measurement integrated with it. The results from line 2 satisfy the first criterion. The measurement data calculated in lines 4–7 satisfy the second part. Thus, we can use the Kalman filter to integrate the new information into the existing distribution. The measurement part of the Kalman filter starts by calculating the Kalman gain and then integrates the new measurement.

6 Methods

To test this example and compare it to a traditional particle filter, a simulated test bed was created to match the environment described by (6)–(13) and Figure 1. Each of the information sources is calculated independently by adding noise to the true state of the related state variable, where the noise was sampled from a zero mean Gaussian distribution with a standard deviation of 0.10.

A standard KF was used for the simulation; however, the KLD sampling variant of the particle filter was used for the particle filters. This allowed the particle set to adapt to the complexity of the distribution for each situation. The KLD-sampling particle filter has some extra parameters which control the precision of the particle set;

these parameters were fixed for both algorithms: $\epsilon = 0.01$, z-score of 1.645 (95%), and bin sizes of 0.025 in their respective measurements. The algorithms were initialized with a single particle representing the true state of the vehicle.

This simulator was built using Java and tested on Java version 1.6.29. EJML version 0.17 was used as the linear algebra package to calculate the Kalman filter equations. The code was not subjected to substantial optimization. Instead, code reuse was the focus of algorithm implementation so that none of the algorithms had an unfair advantage due to coding strategies. Likewise, the code was not multi-threaded during testing. The algorithms were timed by starting a nanosecond scale timer before the simulator entered into the respective modularized Bayesian filter procedure and stopped immediately after it exited the procedure. The results were then reported. The simulator repeated this process at every time step until the list of commands for the vehicle had been exhausted.

Each algorithm was run 50 times on each of three control plans. The control plans described the motion the vehicle should take for five minutes. Each plan was created separately. However, they were each similar in that the vehicle always started in the same position and ended at a stop after the five minutes. During the simulation the vehicle varied its velocity or its turning radius every few seconds.

The simulations were performed on the Big Red cluster at the University of Louisiana at Lafayette. Big Red was a homogeneous Beowulf cluster that consisted of up to 70 compute nodes connected to a primary distribution node. The cluster was built at the end of 2008 using relatively modern hardware for the time. Each of the machines had a dual-processor motherboard with two Intel[®] Xeon[™] 2.8 GHz dual core processors and 1 GB of main memory. The nodes ran CentOS 4.4 with a 32-bit x86 SMP Linux kernel.

The simulations were scheduled so that only one simulation was run on an individual machine at a time. This helped limit memory issues with the particle filters; however, the particle filters were still limited to one million particles per particle set to keep the algorithms within the 1 GB RAM limit so that paging was not an issue. As we will see, this was only marginally successful, but did substantially reduce the overall time to produce results.

7 Results

Averaging all of the simulations at each time step produces the charts shown in Figures 4 and 5. Figure 4 shows the average amount of time required to process the Bayesian filter (vertical axis) at each time step (horizontal axis). Both algorithms are plotted as they progress through the simulation. A numerical representation of the overall averages is presented in Table 2. The table contains the mean and standard deviation for the execution time of each algorithm, as well as the size of their particle sets.

When the traditional particle filter was not being delayed by paging issues, it took about 18 s to compute; however, the modular particle filter consistently required about 1 s for its computations. As seen in Table 2, just comparing the averages directly, the modularized particle filter was about 29 times faster than the traditional particle filter under these conditions. The memory usage was substantially decreased due to the smaller particle set size. The modular particle filter required less than 9% as many particles as the traditional algorithm. The actual results are far better than this comparison shows. In this case, the traditional particle filter was explicitly limited to using 1 000 000 particles, and it did so every time. Without this limit, the particle set size continues to grow; limited testing on 16 GB machines showed that it would hit this increased limit within the first

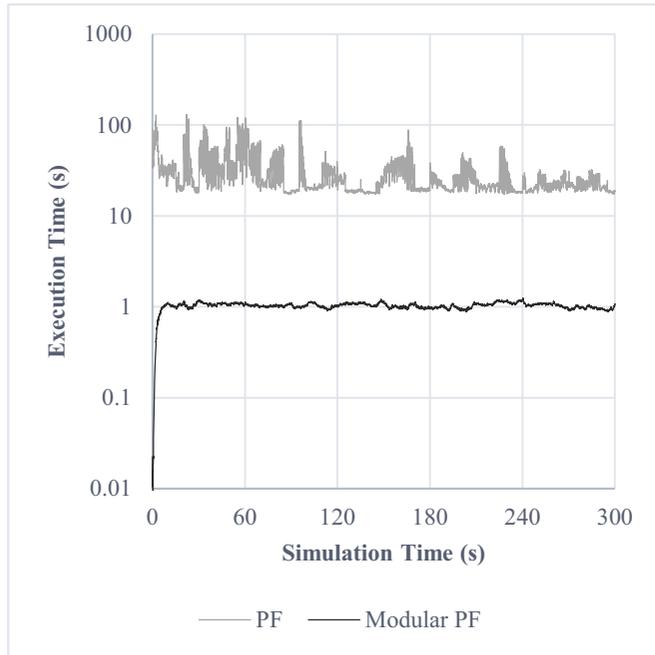


Figure 4. A chart of the simulation results showing the amount of time required at each time step throughout the simulation.

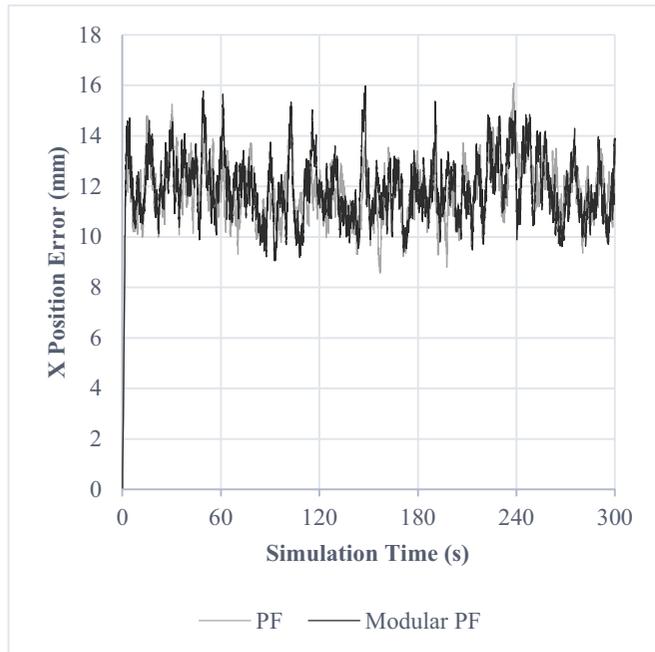


Figure 5. Mean absolute error in the x dimension over the 150 simulated runs for each algorithm. The other dimensions showed similar behavior.

couple seconds of simulation time.

Using the $O()$ time complexity analysis above, we would expect the traditional particle filter to require time commensurate with $O(p^9)$. This modular filter would require $O(p^6)$ for the particle filter, and $O(2p^6)$ to convert the particle set into Gaussian distributions for the inverse filter. The Kalman filter is cheap compared to the processing for the particle sets, and so adds a negligible amount of time to the computation. Thus, the overall computation

Table 2. A numerical presentation of the simulation results with EKF error for comparison.

Algorithm	$\hat{\mu}$	$\hat{\sigma}$
Traditional PF	29.12 s	14.51
Particle Count	1 000 000	0.000
x-Dimension MAE	11.79 mm	1.242
Modular PF	1.024 s	0.1042
Particle Count	87 087	9 445
x-Dimension MAE	11.90 mm	1.316
Traditional EKF MAE	13.07 mm	1.463

time should be $O(3p^6)$. Given that we find 87 087 particles were required to satisfy the conditions for the 6-state-variable particle filter, p should be 6.658 and we would expect the traditional particle filter to want about 25 700 000 particles. Thus we would expect about $25\,700\,000 / (3 \cdot 87\,087) = 98.37$ times speedup on this problem. However, the traditional particle filter was limited to 1 000 000 particles. This would reduce the speedup to 3.83 times. Excluding the paging issues seen in the traditional particle filter, the speedup observed was around 18 times, which is close to the predicted speedup.

Figure 5 shows that modularization does not affect the accuracy of the filter on this problem. The mean error for both algorithms track very closely together, and any discrepancy is clearly being dwarfed by the noise from the measurements.

8 Conclusion

This research applied modularization to particle filters as a means of improving execution time. Modularization works by dividing the problem into a set of simpler sub-problems and recombining the results to maintain the accuracy of the traditional method. These modules then pass messages between them so that incoming information can be distributed throughout the network. This work also addressed a number of issues with filter construction with complex filters. An example was given to show how modularization can be applied to a practical problem. Finally, the example was tested in a simulator to show that the method is effective, even on small problems.

The results from the simulation showed that modularization can be effective even for small particle filters. On larger problems, modularization would allow implementers to choose between improving execution speed and improving the precision of the results.

9 Further Research

The most obvious place for further research is in application. We have shown results for vehicle state estimation here, but it is likely that other Bayesian filter applications would also benefit. When applied to an existing application, a relative performance comparison could be made. Modularization could also be applied to other types of Bayesian filters. We have shown how modularization works with particle filters, but there are several other filtering algorithms in use. Some of those also use specialized probability distributions which would require additional work to use as messages.

REFERENCES

- [1] S.K. Andersen, K.G. Olesen, F.V. Jensen, and F. Jensen, ‘HUGIN*—a shell for building Bayesian belief universes for expert systems’, in *International Joint Conference on Artificial Intelligence*, volume 2, pp. 1080–1085, Detroit, MI, (1989). Morgan Kaufmann Publishers Inc.
- [2] X. Boyen and D. Koller, ‘Tractable inference for complex stochastic processes’, in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI’98*, pp. 33–42, San Francisco, CA, USA, (1998). Morgan Kaufmann Publishers Inc.

- [3] X. Boyen and D. Koller, 'Exploiting the architecture of dynamic systems', in *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pp. 313–320, Menlo Park, CA, USA, (1999). American Association for Artificial Intelligence.
- [4] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, 'Rao-blackwellised particle filtering for dynamic Bayesian networks', in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, pp. 176–183, San Francisco, CA, USA, (2000). Morgan Kaufmann Publishers Inc.
- [5] D. Fox, 'KLD-sampling: adaptive particle filters', *Advances in Neural Information Processing Systems*, **1**(14), 713–720, (2002).
- [6] D. Fox, 'Adapting the sample size in particle filters through KLD-sampling', *International Journal of Robotics Research*, **22**(12), 985–1004, (2003).
- [7] R.E. Kalman, 'A new approach to linear filtering and prediction problems', *Transactions of the ASME—Journal of Basic Engineering*, **82**, 35–45, (1960).
- [8] R.E. Kalman and R.S. Bucy, 'New results in linear filtering and prediction theory', *Journal of Basic Engineering*, **83**, 95–108, (March 1961).
- [9] U. Kjærulff, 'dHugin: A computational system for dynamic time-sliced Bayesian networks', *International Journal of Forecasting*, **11**, 89–111, (1995).
- [10] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, The MIT Press, Cambridge, MA, 2009.
- [11] S.L. Lauritzen and D.J. Spiegelhalter, 'Local computations with probabilities on graphical structures and their application to expert systems', *Journal of the Royal Statistical Society. Series B (Methodological)*, **50**(2), 157–224, (1988).
- [12] J.D. McLean, S.F. Schmidt, and L.D. McGee, 'Optimal filtering and linear prediction applied to a midcourse navigation system for the circumlunar mission', Technical Report NASA-TN-D-1208, NASA, Ames Research Center, Moffett Field, CA, (March 1962).
- [13] T.P. Minka, 'Expectation propagation for approximate Bayesian inference', in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, eds., Jack Breese and Daphne Koller, UAI'01, pp. 362–369, San Francisco, CA, USA, (2001). Morgan Kaufmann Publishers Inc.
- [14] K. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*, Ph.D. dissertation, University of California, Berkeley, Berkeley, CA, USA, 2002.
- [15] K. Murphy and Y. Weiss, 'The factored frontier algorithm for approximate inference in DBNs', in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pp. 378–385, San Francisco, CA, USA, (2001). Morgan Kaufmann Publishers Inc.
- [16] K.P. Murphy, Y. Weiss, and M.I. Jordan, 'Loopy belief propagation for approximate inference: an empirical study', in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pp. 467–475, San Francisco, CA, USA, (1999). Morgan Kaufmann Publishers Inc.
- [17] B. Ng, L. Peshkin, and A. Pfeffer, 'Factored particles for scalable monitoring', in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, pp. 370–377, San Francisco, CA, USA, (2002). Morgan Kaufmann Publishers Inc.
- [18] M.A. Paskin, *Exploiting locality in probabilistic inference*, Ph.D. dissertation, University of California at Berkeley, Berkeley, CA, USA, 2004. AAI3165519.
- [19] J. Pearl, 'Fusion, propagation and structuring in belief networks', *Artificial Intelligence*, **29**(3), 241–288, (September 1986).
- [20] M.A. Peot and R.D. Shachter, 'Fusion and propagation with multiple observations in belief networks', *Artificial Intelligence*, **48**(3), 299–318, (1991).
- [21] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Education Inc., Upper Saddle River, NJ, 2nd edn., 2003.
- [22] A.F.M. Smith and A.E. Gelfand, 'Bayesian statistics without tears: a sampling - resampling perspective', *The American Statistician*, **46**(2), 84–88, (May 1992).
- [23] G.L. Smith, S.F. Schmidt, and L.A. McGee, 'Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle', Technical report, NASA Ames Research Center, Moffett Field, CA, (1962).
- [24] N.L. Zhang and D. Poole, 'Exploiting causal independence in bayesian network inference', *Journal of Artificial Intelligence Research*, **5**, 301–328, (December 1996).

Using the Sugeno Integral in Optimal Assignment Problems with Qualitative Utilities

Soufiane Drissi Oudghiri¹ and Patrice Perny² and Olivier Spanjaard² and Mohamed Hachimi¹

Abstract. This paper is devoted to the assignment problem when the preferences of the agents are defined by qualitative utilities. In this setting, it is not possible to compare assignments by summing up individual utilities because the sum operation becomes meaningless. We study here the optimization of a Sugeno integral of the individual utilities. We show that the problem is NP-hard in the general case, but we also identify special cases that are solvable in polynomial time. Furthermore, we provide a mixed integer programming formulation in the general case, which leads to a compact formulation for k -minitive capacities.

1 INTRODUCTION

Assignment problems appear in many AI applications, in various contexts such as task allocation in robot teams [22], auctions with side constraints [29], heuristic search [21]. Furthermore, the assignment problem is of interest in many problems involving agents expressing preferences, such as automated resource allocation, matching people with services, time-slot allocation. The standard assignment problem can be defined as follows: given a set $N = \{a_1, \dots, a_n\}$ of agents and a set $T = \{t_1, \dots, t_n\}$ of items, an $n \times n$ utility matrix $U = (u_{ij})$, where u_{ij} represents the utility of item t_j for agent a_i , we want to determine a one-to-one assignment maximizing the sum of individual satisfactions. Formally, denoting by $x = (x_1, \dots, x_n)$ an assignment, where $x_i = j$ if item t_j is assigned to agent a_i , the utility of x for agent a_i is $u_i(x) = u_{ix_i}$ and we want to maximize $\sum_{i=1}^n u_i(x)$ (utilitarian criterion). It is well-known that the determination of such an optimal assignment can be performed in polynomial time (e.g., by transforming the problem to a minimization problem and using the Hungarian method [23]).

However the use of the utilitarian criterion requires utilities to be evaluated on a cardinal scale. Cardinal utilities rely on the assumption that the magnitude of increments to satisfaction can be compared across different situations [27]. A cardinal utility scale is a utility index that preserves preference orderings uniquely up to positive affine transformations. Nevertheless, in many situations, we deal with data representing people's opinions, interests, preferences, etc. In this case, data are often ordinal, and therefore should be handled in a *suitable* way unless we have some quite strong arguments making us believe that it can be handled at a higher level (namely as a cardinal scale). As emphasized by Roberts [28], using numerical representations of preference orders may lead people to derive meaningless conclusions when preference aggregation is based on usual

arithmetic operations, even if the utilities u_{ij} take value in a valuation scale \mathbb{L} that is linearly ordered according to a binary relation \succeq . This is illustrated in the following example.

Example 1 Consider an assignment problem where $|N| = |T| = 4$ and the utilities u_{ij} are encoded on an ordinal scale $\mathbb{L} = \{++, +, 0, -, --\}$. The scale is endowed with the following linear order: $++ \succ + \succ 0 \succ - \succ --$. The utility matrix is:

$$\begin{matrix} & t_1 & t_2 & t_3 & t_4 \\ \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{matrix} & \begin{pmatrix} ++ & - & 0 & - \\ - & + & 0 & + \\ ++ & -- & + & 0 \\ + & -- & - & - \end{pmatrix} \end{matrix}$$

Now, assume that the initial scale is transformed into a numerical scale, using an arbitrary monotonic transformation function $\phi : \mathbb{L} \rightarrow \mathbb{Z}^+$. Consider two possible transformation functions ϕ and ϕ' defined by:

u_{ij}	++	+	0	-	--
$\phi(u_{ij})$	3	2	1	-2	-3
$\phi'(u_{ij})$	2	1	0	-1	-2

Note that both resulting numerical scales are compatible with \mathbb{L} , i.e. $\lambda \succ \lambda' \Rightarrow \phi(\lambda) > \phi(\lambda')$ for all $(\lambda, \lambda') \in \mathbb{L}^2$. Let us consider now the two assignments $x = (3, 2, 4, 1)$ and $x' = (1, 2, 3, 4)$. The associated vectors of individual utilities are $u(x) = (0, +, 0, +)$ and $u(x') = (++, +, +, -)$, where the i^{th} component is the utility of agent a_i . With a solver, it can easily be checked that, when using ϕ for numerical encoding, assignment x is optimal and x' is suboptimal. Moreover, when using ϕ' for numerical encoding, we obtain the opposite conclusion since x' is optimal and x is suboptimal. Thus, one observes that slight changes in the numerical values lead to very different conclusions. This illustrates the need for algorithms specifically dedicated to assignment problems with qualitative scales.

In order to compare assignments in a qualitative setting, we need to be able to compare utility vectors $y = (y_1, \dots, y_n)$, where $y = u(x)$ and x is an assignment. Several preference models have been proposed to compare solutions when elements are evaluated on a qualitative scale (for simplicity, we present them in the setting of a qualitative assignment problem):

– Some preference models induce a partial order on the feasible solutions. The most popular model in this class is the pairwise dominance relation defined by: $y \succsim y'$ if there exists a bijection $\pi : [n] \rightarrow [n]$ such that $y_i \succeq y_{\pi(i)}$ for all $i \in [n]$, where $[n] = \{1, \dots, n\}$. The symmetric (resp. asymmetric) part of \succsim , denoted by \sim (resp. \succ), is the indifference (resp. preference) relation.

¹ Laboratoire des sciences de l'ingénieur (LabSi), Université Ibn Zohr, BP 32/S, Agadir, Morocco, email: soufiane.drissioudghiri@edu.uiz.ac.ma, m.hachimi@uiz.ac.ma

² Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France, email: name.surname@lip6.fr

In other words, an assignment x is preferred to x' if there exists a bijection π from A to A such that: every agent a_i is at least as satisfied with x as agent $a_{\pi(i)}$ with x' , and there exists at least an agent a_i that is more satisfied with x than agent $a_{\pi(i)}$ with x' . We call this rule *ordinal dominance* in the following. For example, in the previous instance, $y \succ y'$ for $y = u(1, 2, 3, 4) = (++, +, +, -)$ and $y' = u(4, 3, 2, 1) = (-, 0, --, +)$. To our knowledge, this ordinal dominance relation was introduced by Barteo [1], and revisited by Bossong and Schweigert [2, 31]. Several recent works in a qualitative setting use this relation [3, 4, 5].

– Other preference models induce a complete order on the feasible solutions, namely variations of max ordering, leximax and leximin. The max (resp. min) ordering relation consists in ranking feasible assignments according to the agent whose satisfaction is better off (resp. worse off) among all agents. For example, in the previous instance, if one uses the min ordering relation then $y = u(3, 2, 4, 1) = (0, +, 0, +)$ is preferred to $y' = u(1, 2, 3, 4) = (++, +, +, -)$ since $0 \succ -$. The leximax (resp. leximin) relation is an enrichment of the max (resp. min) relation, that consists in breaking ties by going down the ranking (i.e., if the worst satisfactions are both equal, one compares the second worsts, and so on...). Finally, one can also consider k -max (resp k -min) aggregation operators consisting in evaluating a solution according to its k^{th} largest (resp. smallest) element (in the sense of the order relation on \mathbb{L}).

In this paper, we are interested in the second type of preference models (those inducing a complete order on the solutions). Finding an optimal assignment according to all the above-mentioned models can be performed in polynomial time [7, 15, 19, 32], and sometimes even faster than in the standard cardinal case. Though appealing from the algorithmic viewpoint, these preference models (inducing a complete order on the solutions) are nevertheless quite simple from the normative and descriptive viewpoints.

Before going on with our contribution, let us mention two related works that do not fit with the above classification. The first work deals with assignment problems with ordinal data viewed as social choice problems, typically with assignment functions based on scoring rules [14]. This amounts to consider ranks instead of costs in the standard Hungarian method, which does not raise new algorithmic issues. The second (more recent) work is devoted to a variant of the covering location problems [34] where the objective function is the Sugeno integral. To our knowledge, this is the first attempt to make use of the Sugeno integral in a combinatorial setting. Yet, their motivation is totally different from ours, as they use it to model fuzziness.

We also study here this sophisticated aggregation operator named *Sugeno integral*, that subsums all the previous models and encompasses a much broader set of decision behaviors. Namely, we investigate the optimization of a Sugeno integral of the individual utilities. The paper is organized as follows. In Section 2, we present the Sugeno integral and some special cases of interest. Then, we show that the determination of an optimal assignment in these special cases can be performed in polynomial time (Section 3) while the problem is NP-hard in the general case (Section 4). Finally, we provide a mixed integer programming formulation for the general case, and we briefly discuss the results of preliminary numerical tests (Section 5).

2 THE SUGENO INTEGRAL

In order to compare assignments in a qualitative setting, we need to be able to compare utility vectors $y = (y_1, \dots, y_n)$, where $y_i \in \mathbb{L}$ is the utility of agent a_i (in Example 1, y_i corresponds to the grade of the task assigned to agent a_i) and \mathbb{L} is any finite qualitative scale.

The Sugeno integral is precisely an aggregation operator for ordinal scales, that enables to combine several qualitative values into a single representative one. This operator has been introduced by Sugeno [33], and has many applications in qualitative decision theory under uncertainty [6, 12] and multicriteria decision making [11, 17].

Before coming to the Sugeno integral (and to ease its presentation), let us previously introduce the *weighted maximum* [9], that can be seen as a qualitative version of weighted sum, where the sum is replaced by operation max, and the product by operation min. Assume that each agent a_i has an importance weight w_i , expressed on the same ordinal scale \mathbb{L} as the utilities. The weighted maximum of a vector $y \in \mathbb{L}^n$ then reads as follows: $\max_{i \in [n]} \min\{w_i, y_i\}$. For instance, for allocating the different tasks involved in the writing of a research consortium proposal, the head of the consortium may accept to receive a low-rated task if it helps increase the overall satisfaction. This can be modelled by stating that the utility of the head is at most –let's say– “–” whatever task assigned to her. In other words, the only case in which the satisfaction level y_i of the head is taken into account should be when a task she rated –– is assigned to her. The weighted maximum aggregation operator makes it possible by setting $w_i = -$ for the head. Importantly, note that the discriminative power of the weighted maximum can be enhanced by using for the weights a scale \mathbb{L}' such that $\mathbb{L}' \supseteq \mathbb{L}$. The possible aggregate values are indeed then in \mathbb{L}' , whose range can be much larger than the one of \mathbb{L} .

The Sugeno integral generalizes the weighted maximum by using the notion of *capacity*. In this respect, it is acknowledged as the qualitative counterpart of the Choquet integral [30]. A capacity is a set function $v : 2^N \rightarrow \mathbb{L}$ that represents the importance of a coalition of agents in the present setting. Note that, as for the importance weights in the weighted minimum, the utilities and the capacity values are both expressed on scale \mathbb{L} . Here also, the discriminative power of the Sugeno integral can be significantly enhanced by using a scale $\mathbb{L}' \supseteq \mathbb{L}$ for the capacity values. For simplicity, the range of the capacity is assumed to be \mathbb{L} in the sequel of the paper, but all the presented results are still valid if an enriched scale is used for the capacity values. Another way of breaking ties is to use lexicographic refinements of the Sugeno integral [8, 16]. Most of the algorithms proposed in the remainder of the paper can be adapted to handle such refinements.

Definition 1 Consider the finite set $[n] = \{1, \dots, n\}$. A capacity is a set function $v : 2^{[n]} \rightarrow \mathbb{L}$ such that:

$$i) v(\emptyset) = \perp, v([n]) = \top,$$

$$ii) \forall A, B \subseteq [n], A \subseteq B \Rightarrow v(A) \preceq v(B),$$

where \top (resp. \perp) denotes the max (resp. min) element in \mathbb{L} according to \succeq .

For any subset $A \subseteq [n]$, $v(A)$ represents the importance of coalition A of agents. Let us now recall some definitions about capacities.

Definition 2 Let v be a capacity on $[n]$. We say that v is:

$$- \text{minitive if for any subsets } A, B \subseteq [n], v(A \cap B) = v(A) \wedge v(B);$$

$$- \text{maxitive if for any subsets } A, B \subseteq [n], v(A \cup B) = v(A) \vee v(B);$$

$$- \text{symmetric if for any subsets } A, B \subseteq [n], |A| = |B| \text{ implies } v(A) = v(B);$$

where $\wedge := \min$ and $\vee := \max$.

We are now able to define the Sugeno integral:

Definition 3 The discrete Sugeno integral of $y = (y_1, \dots, y_n)$ has

the following equivalent formulas:

$$S_v(y) := \bigvee_{i=1}^n (y_{(i)} \wedge v(A_{(i)})) \quad (1)$$

$$S_v(y) := \bigwedge_{i=1}^n (y_{(i)} \vee v(A_{(i+1)})) \quad (2)$$

where (\cdot) is a permutation on $[n]$ such that $y_{(1)} \preceq \dots \preceq y_{(n)}$, $\vee := \max$, $\wedge := \min$, $A_{(i)} := \{(i), \dots, (n)\}$ and $A_{(n+1)} := \emptyset$.

Note that, even if there are several permutations σ such that $y_{\sigma(1)} \preceq \dots \preceq y_{\sigma(n)}$ (due to possible ex aequos), the value of the Sugeno integral is uniquely defined. In formula 2 above, $v(A_{(i+1)})$ represents the importance of the coalition of agents that are better off than a_i in y . The utility y_i of agent a_i will be more taken into account in the evaluation of y as the importance of this coalition will be low.

Example 2 Coming back to Example 1, consider assignment $x = (1, 2, 3, 4)$. The corresponding utility vector is $y = (++, +, +, -)$. Assume that $v(\{1\}) = 0$, $v(\{1, 2\}) = +$, $v(\{1, 2, 3\}) = ++$. By definition of a capacity, $v(\emptyset) = --$ and $v([4]) = ++$. The Sugeno integral $S_v(y)$ of y reads as follows:

$$\begin{aligned} & \bigvee_{i=1}^4 (y_{(i)} \wedge v(A_{(i)})) \\ &= (y_4 \wedge v([4])) \vee (y_3 \wedge v(\{1, 2, 3\})) \vee (y_2 \wedge v(\{1, 2\})) \vee (y_1 \wedge v(\{1\})) \\ &= (- \wedge ++) \vee (+ \wedge ++) \vee (+ \wedge +) \vee (+ \wedge 0) = + \end{aligned}$$

Equivalently, it reads as follows:

$$\begin{aligned} & \bigwedge_{i=1}^4 (y_{(i)} \vee v(A_{(i+1)})) \\ &= (y_4 \vee v(\{1, 2, 3\})) \wedge (y_3 \vee v(\{1, 2\})) \wedge (y_2 \vee v(\{1\})) \wedge (y_1 \vee v(\emptyset)) \\ &= (- \vee ++) \wedge (+ \vee +) \wedge (+ \vee 0) \wedge (+ \vee --) = + \end{aligned}$$

A well-known example of Sugeno integral is the so-called h-index, used to measure both the productivity and citation impact of a scientist. If an author has published n papers, let us define (y_1, \dots, y_n) as the citation vector of the author, where y_i is the number of times paper i is cited. By setting $v(A_{(i)}) = |A_{(i)}| = n - i + 1$, the value $S_v(y)$ is nothing else but the h-index of the author [35].

The Sugeno integral is more expressive than the weighted maximum, which is only a special case obtained for a maxitive capacity v such that $v(\{i\}) = w_i$ for all $i \in [n]$ [18] (in this case, $v(A) = \max_{i \in A} w_i$, which makes v a possibility measure, and weighted max a qualitative expected utility [10]). Actually, Marichal [25] showed that if one restricts to the use of a combination of max and min operators (together with qualitative weights), then the Sugeno integral is the only solution for aggregating qualitative values, provided one adds the natural constraint that the aggregated value for (\perp, \dots, \perp) should be \perp and the one for (\top, \dots, \top) should be \top .

The Sugeno integral includes other well identified special cases [18]:

- $S_v(y) = \min_i y_i$ iff $v(A) := \perp$ for all $A \subsetneq [n]$;
- $S_v(y) = \max_i y_i$ iff $v(A) := \top$ for all $A \subseteq [n]$, $A \neq \emptyset$;
- $S_v(y) = k\text{-min}_i y_i$ iff $v(A) := \perp$ for $|A| \leq n - k$ and \top otherwise;
- $S_v(y)$ is the *ordered weighted maximum* w.r.t. w defined by $\text{OWMax}(y) := \bigvee_{i=1}^n (w_i \wedge y_{(i)})$ where $y_{(1)} \preceq \dots \preceq y_{(n)}$ and $w_1 \succeq \dots \succeq w_n$, iff v is a symmetric capacity such that $v(A) = w_{n-|A|+1}$ for any $A \subseteq [n]$, $A \neq \emptyset$;

– $S_v(y)$ is the *ordered weighted minimum* defined by $\text{OWMin}_w(y) := \bigwedge_{i=1}^n (w_i \vee y_{(i)})$ where $y_{(1)} \preceq \dots \preceq y_{(n)}$ and $w_1 \succeq \dots \succeq w_n$, iff v is a symmetric capacity such that $v(A) = w_{n-|A|}$ for any $A \subsetneq [n]$.

The ordered weighted minimum encompasses a gradation of aggregation operators ranging from the min operator ($w_i = \perp$ for all i) to the max operator ($w_n = \perp$ and $w_i = \top$ for $i \neq n$), including the k -min operators. Note that the assumption $w_1 \succeq \dots \succeq w_n$ is not restrictive since $\bigwedge_{i=1}^n (w_i \vee y_{(i)}) = \bigwedge_{i=1}^n ((\bigwedge_{k=1}^i w_k) \vee y_{(i)})$.

Example 3 Coming back to Example 1, assume now that we wish to evaluate utility vector $y = (++, +, +, -)$ with the ordered weighted minimum and weights $w_1 = ++$, $w_2 = 0$, $w_3 = -$ and $w_4 = --$. The ordered weighted minimum $\text{OWMin}_w(y)$ is:

$$\begin{aligned} & (w_1 \vee y_{(1)}) \wedge (w_2 \vee y_{(2)}) \wedge (w_3 \vee y_{(3)}) \wedge (w_4 \vee y_{(4)}) \\ &= (w_1 \vee y_4) \wedge (w_2 \vee y_3) \wedge (w_3 \vee y_2) \wedge (w_4 \vee y_1) \\ &= (+ \vee --) \wedge (0 \vee +) \wedge (- \vee +) \wedge (- \vee +) = + \end{aligned}$$

In the next section, we show that an assignment maximizing the OWMin_w operator can be determined in polynomial time, as well as one optimizing the weighted minimum.

3 POLYNOMIAL CASES

This section is devoted to the presentation of polynomial time solution procedures for the determination of optimal assignments according to special cases of the Sugeno integral.

3.1 Weighted min and weighted max

To warm up, we begin with the weighted minimum and the weighted maximum because it is very simple to show the polynomial time complexity in these cases. The solution procedures are derived from the threshold method for the max min assignment problem.

The max min assignment problem aims at determining an assignment x that maximizes $\min_i u_i(x)$, where $u_i(x)$ is the (qualitative) utility of agent a_i in assignment x . We recall that the threshold algorithm to compute a max min assignment works as follows: it consists in determining $\max\{\lambda \in \mathbb{L} : \exists x \forall i u_i(x) \succeq \lambda\}$. In order to know if the max min value is greater or equal to a given $\lambda \in \mathbb{L}$, one uses a max flow algorithm to compute a maximal matching in the bipartite graph $G_\lambda = (V_1, V_2, E)$ where $V_1 = \{a_1, \dots, a_n\}$ (resp. $V_2 = \{t_1, \dots, t_n\}$) is the set of agents (resp. tasks) and there is an edge between a_i and t_j if $u_{ij} \succeq \lambda$. We recall that a maximal matching is a set of non-adjacent edges of maximal cardinality. It is said to be perfect when the cardinality is n . The maximal matching is perfect in G_λ iff the value of a max min assignment is at least λ . One can use a bisection search on \mathbb{L} to speed up the determination of the minimal λ such that a feasible assignment exists. This assignment is then a max min assignment.

The adaptation of this method to compute a maximal assignment according to a weighted minimum $\bigwedge_{i=1}^n (w_i \vee u_i(x))$ simply consists in replacing u_{ij} by $u_{ij} \vee w_i$ in the matrix, and then applying the threshold method used for the max min case. The correctness of this approach is obvious. The generation of the transformed matrix is in $O(n^2)$, and then there are at most $\log |\mathbb{L}|$ calls to a max flow algorithm in $O(n^3)$ [24]. The overall complexity is therefore $O(n^3 \log |\mathbb{L}|)$.

This positive result also holds for the determination of an assignment maximizing the weighted maximum $\bigvee_{i=1}^n (w_i \wedge u_i(x))$. Such

an assignment can indeed be simply computed as follows. Let i^*, j^* denote indices such that $u_{i^*j^*} = \max_i \max_j (u_{ij} \wedge w_i)$. Any assignment where task t_{j^*} is assigned to agent a_{i^*} is optimal for the weighted maximum.

3.2 OWMin and OWMax

The procedure to determine a maximal assignment according to the $OWMin_w$ operator is also a threshold method, based on the following proposition:

Proposition 1 Let $y \in \mathbb{L}^n$. The following property holds:

$$|\{i \in [n] : y_i \succeq \lambda\}| \geq |\{i \in [n] : w_i \prec \lambda\}| \Leftrightarrow OWMin_w(y) \succeq \lambda.$$

Proof Assume that $|\{i \in [n] : y_i \succeq \lambda\}| \geq |\{i \in [n] : w_i \prec \lambda\}|$. By definition of $OWMin_w(y)$, this implies that all values $y_{(i)} \prec \lambda$ are weighted by $w_i \succeq \lambda$. Consequently, $OWMin_w(y) \succeq \lambda$.

Conversely, assume that $|\{i \in [n] : y_i \succeq \lambda\}| < |\{i \in [n] : w_i \prec \lambda\}|$. By definition of $OWMin_w(y)$, this implies there exists a rank i for which $y_{(i)} \prec \lambda$ and $w_i \prec \lambda$. Consequently, $OWMin_w(y) \prec \lambda$. ■

Proposition 1 may be interpreted in the following way in the setting of ordinal assignment problems: the value of the ordered weighted min of an assignment is greater than or equal to λ if and only if there are at least $g(\lambda)$ agents whose (qualitative) utility is at least λ , where $g(\lambda) = |\{i \in [n] : w_i \prec \lambda\}|$. A threshold method can therefore be used to determine $\min\{\lambda \in \mathbb{L} : \exists x \text{ s.t. } |\{i \in [n] : u_i(x) \succeq \lambda\}| \geq g(\lambda)\}$. In order to know if the optimal value is greater than or equal to a given $\lambda \in \mathbb{L}$, one uses a max flow algorithm to compute a maximal matching in the same bipartite graph G_λ as above, where the set of edges is now defined by $E = \{(a_i, t_j) : u_{ij} \succeq \lambda\}$. If the cardinality of a maximal matching is greater than or equal to $g(\lambda)$, then the optimal value is greater than or equal to λ , otherwise it is the opposite. By using here also a binary search to determine the minimal λ , the complexity of the whole threshold method is of course the same as for the max min case ($O(n^3 \log |\mathbb{L}|)$).

Example 4 Coming back to Example 1, assume that we wish to determine an optimal assignment for $OWMin_w$ with $w = (++, 0, -, --)$. The matrix with the utilities of the agents is recalled on the top of Figure 1. Since we have $|\{i \in [n] : w_i \prec ++\}| = 3$ and there does not exist a matching of cardinality 3 in G_{++} (graph in the bottom left corner of Figure 1), the optimal value is necessarily strictly smaller than $++$. However, we have $|\{i \in [n] : w_i \prec +\}| = 3$ and there exists a matching of cardinality 3 in G_+ (see the dotted edges in the bottom right graph of Figure 1). By completing this matching in a complete assignment (by assigning task t_4 to agent a_4 here), one obtains an assignment x such that $OWMin_w(u(x)) = +$. This assignment is optimal for $OWMin_w$.

A similar approach can be used for maximizing an $OWMax$ operator: the value of the ordered weighted max of an assignment is greater than or equal to λ if and only if there are at least $g(\lambda) + 1$ agents whose qualitative utility is at least λ .

Remark 1 Note that the ordered weighted operators $f : \mathbb{L}^n \rightarrow \mathbb{L}$ suffer from the drowning effect, i.e. one can have $f(y) = f(y')$ while $y' \succ y$ (ordinal dominance). This is due to the fact that all these operators use max and min operations. In order to guarantee that the returned assignment x is non-dominated (i.e., $\forall x'$

$not(u(x') \succ u(x))$), one can adapt the threshold methods as follows. One replaces the max flow algorithm by a max flow min cost algorithm from s to t in the network $G = (V, E, c)$ with unit capacities, where $V = \{a_1, \dots, a_n\} \cup \{t_1, \dots, t_n\} \cup \{s, t\}$, $E = \{(s, a_i) : i \in [n]\} \cup \{(a_i, t_j) : i \in [n], j \in [n]\} \cup \{(t_j, t) : j \in [n]\}$ and the costs of edges (with unit capacities) are defined by $c(a_i, t_j) = \epsilon_\lambda$ (resp. $1 + \epsilon_\lambda$) if $u_{ij} \succeq \lambda$ (resp. $u_{ij} \prec \lambda$), $c(s, a_i) = 0$, $c(t_j, t) = 0$. Furthermore, we assume that $\epsilon_\lambda < \epsilon_{\lambda'}$ for $\lambda \succ \lambda'$ and that $\epsilon_\lambda \ll 1 \forall \lambda \in \mathbb{L}$. The reader can easily verify that the min cost of a max s - t flow in G is strictly smaller than 1 (resp. $n - g(\lambda) + 1$) iff there are at least n (resp. $g(\lambda)$) edges of costs ϵ_λ ($\lambda \in \mathbb{L}$) with a flow 1. We claim that the assignment corresponding to a max flow of min cost for the optimal value of λ necessarily is non-dominated. This comes from the fact that $y' \succ y \Rightarrow \sum_{i \in [n]} \epsilon_{y'_i} < \sum_{i \in [n]} \epsilon_{y_i}$.

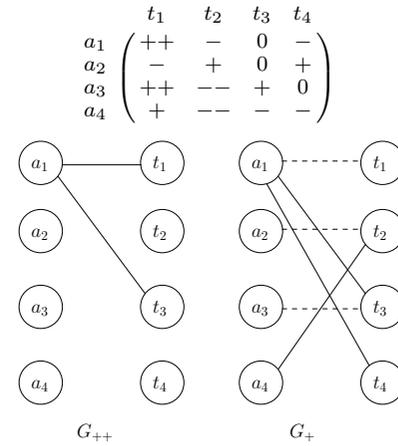


Figure 1: Illustration of Example 4.

3.3 Sugeno with k-maxitive capacities

We now show how to extend the polynomiality result for the weighted maximum to a wider class of Sugeno integrals. In order to define this class, let us recall the definition of the Möbius transform of a capacity v in a qualitative setting and the formulation of the Sugeno integral in terms of ordinal Möbius masses. Note that the Möbius transform is not uniquely defined in the ordinal case [18]. Nevertheless, we use in this paper a specific Möbius transform that we call canonical ordinal Möbius transform.

Definition 4 The canonical ordinal Möbius transform of a capacity v is the set function m on $[n]$ defined by:

$$m(A) := \begin{cases} v(A) & \text{if } v(A) \succ v(A \setminus \{i\}), \forall i \in A \\ \perp & \text{otherwise,} \end{cases}$$

Note that if $v(A) \in \mathbb{L}$ then $m(A) \in \mathbb{L}$. Coefficients $m(A)$, for $A \subseteq [n]$, are called ordinal Möbius masses. Conversely, capacity v can be recovered from m using the following equation:

$$v(E) = \bigvee_{A \subseteq E} m(A)$$

Interestingly enough, any Sugeno integral can be rewritten using ordinal Möbius masses as follows [18]:

$$S_v(y) = \bigvee_{A \subseteq [n]} \left(\bigwedge_{i \in A} y_i \wedge m(A) \right) \quad (3)$$

The computation of $S_v(y)$ with Equation 3 involves 2^n Möbius masses. However only a polynomial number of them must be considered in some particular cases. For example, this is the case for k -maxitive capacities that can be seen as ordinal counterparts of k -additive capacities, see [26].

Definition 5 A capacity v defined on an ordinal scale \mathbb{L} endowed with a smallest (resp. greatest) element denoted by \perp (resp. \top) is said to be k -maxitive if the canonical Möbius transform m satisfies:

- (i) $\forall A \subseteq [n], |A| > k \Rightarrow m(A) = \perp$
- (ii) $\exists A \subseteq [n], |A| = k$ s.t. $m(A) \neq \perp$

Given a capacity v , there is no need to explicitly compute all ordinal Möbius masses to decide whether it is k -maxitive or not. The recognition can be made directly on v using the following characterization result:

Proposition 2 v is a k -maxitive capacity if and only if $v(A) = \top \forall A \subseteq [n]$ such that $|A| \geq k$.

Proof Assume that v is k -maxitive. By definition, $\forall A \subseteq [n], |A| > k \Rightarrow m(A) = \perp$ which implies that $\forall A \subseteq [n], |A| > k \Rightarrow v(A) = v(A \setminus \{i\}) \forall i \in A$. Since $v([n]) = \top$ we obtain the desired result by induction. Conversely, for any A such that $|A| > k$ and any $i \in A$, we have $v(A) = v(A \setminus \{i\}) = \top$. Hence $m(A) = \perp$ by Definition 4, which completes the proof. ■

Interestingly enough, the Sugeno integral can be reformulated as follows for k -maxitive capacities:

$$\begin{aligned} S_v(y) &= \bigvee_{A \subseteq [n]} \left(\bigwedge_{i \in A} y_i \wedge m(A) \right) \\ &= \left(\bigvee_{A \subseteq [n], |A| \leq k} \left(\bigwedge_{i \in A} y_i \wedge m(A) \right) \right) \vee \left(\bigvee_{A \subseteq [n], |A| > k} \left(\bigwedge_{i \in A} y_i \wedge m(A) \right) \right) \\ &= \left(\bigvee_{A \subseteq [n], |A| > k} \left(\bigwedge_{i \in A} y_i \wedge m(A) \right) \right) \vee \left(\bigvee_{A \subseteq [n], |A| > k} \left(\bigwedge_{i \in A} y_i \wedge \perp \right) \right) \\ &= \left(\bigvee_{A \subseteq [n], |A| \leq k} \left(\bigwedge_{i \in A} y_i \wedge m(A) \right) \right) \vee \left(\bigvee_{A \subseteq [n], |A| > k} (\perp) \right) \\ &= \bigvee_{A \subseteq [n], |A| \leq k} \left(\bigwedge_{i \in A} y_i \wedge m(A) \right) \end{aligned}$$

This compact formulation of the Sugeno integral, involving only $\binom{n}{k}$ coefficients, enables to identify a new polynomial case:

Theorem 1 The determination of an assignment x maximizing $S_v(u(x))$ can be performed in polynomial time if the capacity is k -maxitive for a fixed k .

Proof Given a feasible assignment x , the Sugeno value of $u(x)$ for a k -maxitive capacity reads:

$$S_v(u(x)) = \bigvee_{A \subseteq [n], |A| \leq k} \left(\bigwedge_{i \in A} u_{i x_i} \wedge m(A) \right)$$

Thus, we easily deduce the following:

$$S_v(u(x)) \succeq \lambda \Leftrightarrow \left[\exists A \subseteq [n], |A| \leq k \text{ s.t. } \left(\bigwedge_{i \in A} u_{i x_i} \wedge m(A) \right) \succeq \lambda \right]$$

To check if an assignment x s.t. $S_v(u(x)) \succeq \lambda$ exists, we propose the following procedure. For every $A \subseteq [n], |A| \leq k$ s.t. $m(A) \succeq \lambda$, compute a maximal matching $M_\lambda(A)$ in the bipartite graph $G_\lambda^A = (A, T, E_\lambda)$, where $E_\lambda = \{(a_i, t_j) \mid u_{i j} \succeq \lambda\}$. The procedure can be stopped as soon as a set A such that $|M_\lambda(A)| = |A|$ is found. Matching M_λ can then be arbitrarily completed into a feasible assignment, without downgrading the Sugeno value. If no such set A is found, then $\max_x S_v(u(x)) \prec \lambda$. By using a standard bisection algorithm, we can determine the maximal λ such that $S_v(u(x)) \succeq \lambda$ for some feasible assignment x . The corresponding assignment x is, by construction, an optimal assignment. For each value of λ , there are at most $O(n^k)$ calls to a max flow algorithm in $O(n^3)$. The number of examined values for λ is in $O(\log |\mathbb{L}|)$. The overall complexity is therefore $O(n^{3+k} \log |\mathbb{L}|)$. ■

4 COMPLEXITY IN THE GENERAL CASE

In this section we prove that the determination of a Sugeno optimal assignment is an NP-hard problem. We assume that the size function of an instance of the ordinal assignment problem is n . Consequently, we also assume that the capacity function is not defined in extension (which would take up $O(2^n)$ space), but rather that the determination of $v(A)$ for a given $A \subseteq [n]$ can be carried out in polynomial time in n . The following result holds:

Theorem 2 The determination of an assignment x maximizing $S_v(u(x))$ is NP-hard, even if $|\mathbb{L}| = 2$.

Proof The proof relies on a polynomial reduction from the monotone 1-in-3-SAT problem defined as follows:

Instance: A collection of clauses $C = \{C^1, \dots, C^m\}$, each clause consists of exactly three variables, the set of boolean variables is denoted by $B = \{b_1, \dots, b_n\}$, and all literals are positive.

Question: Does there exists a truth assignment of boolean variables in B such that each clause has exactly one true variable ?

From an instance of 1-in-3-SAT, we define an instance of the ordinal assignment problem as follows:

- the set of agents is $N = \{b_{i_1}^1, b_{j_1}^1, b_{k_1}^1, \dots, b_{i_m}^m, b_{j_m}^m, b_{k_m}^m\}$ where i_p (resp. j_p, k_p) denotes the index of the first (resp. second, third) variable in C^p ;
- the set of tasks is $T = \{c_1, \dots, c_m\} \cup \{f_1, \dots, f_{2m}\}$ where task c_p represents clause C^p and tasks f_1, \dots, f_{2m} are added to have $|N| = |T| = 3m$;
- the utility of assigning task c_p to b_p^p ($\forall p \in [m]$) is \top , otherwise it is \perp .
- the capacity v in the Sugeno integral is:

$$v(A) = \begin{cases} \top & \text{if } |A| > m \\ \top & \text{if } |A| = m \text{ and } \forall i \in [n], A \cap N_i \neq \emptyset \Rightarrow N_i \subseteq A \\ \perp & \text{if } |A| = m \text{ and } \text{not}(\forall i \in [n], A \cap N_i \neq \emptyset \Rightarrow N_i \subseteq A) \\ \perp & \text{if } |A| < m \end{cases}$$

where $N_i = \{b_i^p : p \in [m] \text{ and } b_i^p \in N\}$ is the set of occurrences of variable b_i in N (it is easy to check that the capacity is well defined). For illustration, the matrix obtained from formula $(b_1 \vee b_2 \vee b_3) \wedge$

$(b_1 \vee b_2 \vee b_5) \wedge (b_2 \vee b_3 \vee b_4)$ is (all empty components are \perp):

$$\begin{matrix} b_1^1 \\ b_1^2 \\ b_1^3 \\ b_2^1 \\ b_2^2 \\ b_2^3 \\ b_3^1 \\ b_3^2 \\ b_3^3 \\ b_4^1 \end{matrix} \begin{pmatrix} c_1 & c_2 & c_3 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\ \top & & & & & & & & \\ \top & & & & & & & & \\ \top & & & & & & & & \\ & \top & & & & & & & \\ & \top & & & & & & & \\ & \top & & & & & & & \\ & & \top & & & & & & \\ & & \top & & & & & & \\ & & & \top & & & & & \\ & & & \top & & & & & \\ & & & & \top & & & & \\ & & & & & \top & & & \\ & & & & & & \top & & \\ & & & & & & & \top & \\ & & & & & & & & \top \end{pmatrix}.$$

We claim that the value of an optimal assignment w.r.t. S_v is \top iff the answer to the 1-in-3-SAT instance is yes.

First, assume that the value of an optimal assignment x w.r.t. S_v is $S_v(u(x)) = \top$. Let $y = u(x)$ denote the vector of individual utilities ($3m$ components) for x . We recall that our objective function is $S_v(y) = \bigwedge_{i=1}^{3m} (y_{(i)} \vee v(A_{(i+1)}))$. By construction of the instance of the ordinal assignment problem, there are at most m components of y that are equal to \top . By definition of v , $v(A_{(i+1)}) = \perp$ for $i = 2m + 1, \dots, 3m$ (because $|A_{(i+1)}| < m$). Consequently, we necessarily have $y_{(i)} = \top$ for $i = 2m + 1, \dots, 3m$ (otherwise $S_v(y) \neq \top$). By looking at $y_{(2m+1)}, \dots, y_{(3m)}$, for each clause we have therefore one variable (belonging to the clause) assigned to it. By setting these variables to true and the other variables to false, one obtains a feasible truth assignment for the 1-in-3-SAT instance if the following property holds: if a variable belongs to several clauses, if it is assigned to one of them, then it is assigned to all of them. This is ensured by the way capacity v is defined. As shown previously, we have indeed necessarily $y_{(2m)} = \perp$, which implies $v(A_{(2m+1)}) = \top$ (otherwise $S_v(y) \neq \top$). Note that $|A_{(2m+1)}| = m$. By definition of v , it means that $\forall i \in [n] A \cap N_i \neq \emptyset \Rightarrow N_i \subseteq A$, which is precisely the required property.

Conversely, assume that the answer to the 1-in-3-SAT instance is yes. By assigning c_p to b_i^p iff variable b_i is true in the solution of the 1-in-3-SAT instance, one obtains an assignment x such that $S_v(u(x)) = \top$ (this is proved by similar arguments as above). ■

Since a polynomial procedure is unlikely to exist in the general case (unless $P = NP$), we looked for a MIP formulation. This is the topic of the next section.

5 A MIP FORMULATION

To model our problem as a mixed integer program (MIP), we need to find a linear reformulation of the Sugeno integral. In this respect, Equations 1 and 2 are not very useful because they are based on a sorting of the components of y , therefore their linearization would require to introduce binary permutation variables. Equation 3 presents a more interesting form since the ordered aggregation is achieved by an aggregation over all subsets of N . This type of formulation may appear as intractable as the number of agents increase, but we are going to show that it can be efficiently used to minimize the Sugeno integral provided that the capacity is k -maxitive. Note that the maximization case has been treated in Section 3 dedicated to the polynomial cases. Let us consider the following minimization problem, that can be of interest when variables y_i represent disutilities:

$$\begin{aligned} \min \quad & \bigvee_{A \subseteq [n]} \left(\bigwedge_{i \in A} y_i \wedge m(A) \right) \\ \text{s.t.} \quad & y \in Y \end{aligned}$$

where $Y \subseteq [0, 1]^n$ denotes the set of feasible utility vectors. It is equivalent to the MIP:

$$\min w \tag{4}$$

$$\text{s.t.} \begin{cases} w \geq t_A \quad \forall A \subseteq [n] \\ t_A = \sum_{i \in A} b_i^A y_i + b_0^A m(A) \quad \forall A \subseteq [n] \\ \sum_{i \in A} b_i^A + b_0^A = 1 \quad \forall A \subseteq [n] \end{cases} \tag{5}$$

where $b_i^A \in [0, 1]$ and $t_A = \sum_{i \in A} b_i^A y_i + b_0^A m(A)$ represents the value $\bigwedge_{i \in A} (y_i \wedge m(A))$ at optimum. Note that constraint 5 is quadratic, but the quadratic terms can be removed by introducing product variables $p_i^A = b_i^A y_i$ and the following constraints:

$$p_i^A \leq b_i^A \tag{6}$$

$$p_i^A \leq y_i \tag{7}$$

$$p_i^A \geq y_i + b_i^A - 1 \tag{8}$$

This usual reformulation through the linearization constraints 6, 7 and 8 is due to Fortet [13, 20]. Every triple of constraints insures that $b_i^A = 0 \Rightarrow p_i^A = 0$ and $b_i^A = 1 \Rightarrow p_i^A = y_i$. Hence we obtain the following mathematical program:

$$\begin{aligned} \min \quad & w \\ \text{s.t.} \quad & \begin{cases} w \geq t_A \quad \forall A \subseteq [n] \\ t_A = \sum_{i \in A} p_i^A + b_0^A m(A) \quad \forall A \subseteq [n] \\ \sum_{i \in A} b_i^A + b_0^A = 1 \quad \forall A \subseteq [n] \\ p_i^A \leq b_i^A \quad \forall A \subseteq [n] \quad \forall i \in A \\ p_i^A \leq y_i \quad \forall A \subseteq [n] \quad \forall i \in A \\ p_i^A \geq y_i + b_i^A - 1 \quad \forall A \subseteq [n] \quad \forall i \in A \end{cases} \end{aligned}$$

This formulation is not compact for general capacities since the number of variables and constraints grows exponentially with n . However the formulation becomes compact for k -maxitive capacities. Under the k -maxitivity assumption indeed, we have $\bigwedge_{i \in A} (y_i \wedge m(A)) = \perp$ for $|A| > k$. The constraints and variables that were needed for all subsets of $[n]$ become therefore only necessary for all subsets of size not greater than k . Hence the number of variables and constraints becomes polynomial in n (assuming that k is a constant).

Unfortunately the linearization technique presented in the case of a minimization does not apply directly for maximizing the Sugeno integral. To overcome the problem, we now use another formulation of the Sugeno integral based on ordinal co-Möbius masses [18]. We use here the canonical ordinal co-Möbius transform \overline{m} defined by:

$$\overline{m}(A) := \begin{cases} v([n] \setminus A) & \text{if } v([n] \setminus A) < v(([n] \setminus A) \cup \{i\}), \forall i \in A \\ \top & \text{otherwise,} \end{cases}$$

Note that, here also, $\overline{m}(A) \in \mathbb{L}$. The Sugeno integral can be expressed as a function of \overline{m} as follows [18]:

$$S_v(y) = \bigwedge_{A \subseteq [n]} \left(\bigvee_{i \in A} y_{(i)} \vee \overline{m}(A) \right) \tag{9}$$

Let us consider now the maximization problem:

$$\begin{aligned} \max \quad & \bigwedge_{A \subseteq [n]} \left(\bigvee_{i \in A} y_{(i)} \vee \overline{m}(A) \right) \\ \text{s.t.} \quad & y \in Y \end{aligned}$$

This problem can be reformulated as:

$$\max w \tag{10}$$

$$\text{s.t.} \begin{cases} w \leq d_A \quad \forall A \subseteq [n] \\ d_A = \sum_{i \in A} b_i^A y_i + b_0^A \overline{m}(A) \quad \forall A \subseteq [n] \\ \sum_{i \in A} b_i^A + b_0^A = 1 \quad \forall A \subseteq [n] \end{cases} \tag{11}$$

where $b_i^A \in [0, 1]$ and $d_A = \sum_{i \in A} b_i^A y_i + b_0^A \overline{m}(A)$ represents the value $\bigvee_{i \in A} y_{(i)} \vee \overline{m}(A)$ at optimum. Using the Fortet linearization recalled above, the maximization problem can be reformulated as follows:

$$\begin{aligned} \max w & \quad (12) \\ \text{s.t.} & \begin{cases} w \leq d_A \quad \forall A \subseteq [n] \\ d_A = \sum_{i \in A} p_i^A + b_0^A \overline{m}(A) \quad \forall A \subseteq [n] \\ \sum_{i \in A} b_i^A + b_0^A = 1 \quad \forall A \subseteq [n] \\ p_i^A \leq b_i^A \quad \forall A \subseteq [n] \quad \forall i \in A \\ p_i^A \leq y_i \quad \forall A \subseteq [n] \quad \forall i \in A \\ p_i^A \geq y_i + b_i^A - 1 \quad \forall A \subseteq [n] \quad \forall i \in A \end{cases} \quad (13) \end{aligned}$$

Here also, a more compact formulation can be obtained for k -minitive capacities:

Definition 6 A capacity v defined on an ordinal scale \mathbb{L} endowed with a smallest (resp. greatest) element denoted by \perp (resp. \top) is said to be k -minitive if its ordinal co-Möbius transform is equal to \top for any $A \subseteq [n]$ such that $|A| > k$, and there exists at least one subset A of exactly k elements such that $\overline{m}(A) \neq \top$. Formally:

$$\begin{aligned} (i) \quad & \forall A \subseteq [n], |A| > k \Rightarrow \overline{m}(A) = \top \\ (ii) \quad & \exists A \subseteq [n], |A| = k \text{ s.t. } \overline{m}(A) \neq \top \end{aligned}$$

Given a capacity v , there is no need to explicitly compute all ordinal co-Möbius masses to decide whether it is k -maxitive or not. The recognition can be made directly on v by using the following result:

Proposition 3 v is a k -minitive capacity if and only if $v([n] \setminus A) = \perp$ for all $A \subseteq [n]$ such that $|A| \geq k$.

Proof Assume that v is k -minitive. By definition, $\forall A \subseteq [n]$, $|A| > k \Rightarrow \overline{m}(A) = \top$ which implies that $\forall A \subseteq [n]$, $|A| > k \Rightarrow v([n] \setminus A) = v(([n] \setminus A) \cup \{i\}) \forall i \in A$. Since $v(\emptyset) = \perp$ we obtain the desired result by induction. Conversely, let A be a subset such that $|A| = k$. We have $v([n] \setminus A) = v([n] \setminus A) \cup \{i\} = \perp$ for all $i \in A$. Hence $\overline{m}(A) = \top$ by definition of \overline{m} , which completes the proof. ■

Under the k -minitivity assumption, the constraints and variables that were needed for all subsets of $[n]$ become therefore only necessary for all subsets of size not greater than k .

In order to apply this mathematical programming technique to the Sugeno assignment problem (maximization case) where utilities are expressed on a qualitative scale \mathbb{L} , we need first to convert \mathbb{L} to the following numeric scale $C = \{\frac{1}{|\mathbb{L}|}, \frac{2}{|\mathbb{L}|}, \dots, 1\}$. Then we consider boolean variables z_{ij} representing assignment decisions. We have $z_{ij} = 1$ whenever task j is assigned to agent i and $z_{ij} = 0$ otherwise. Hence, assignment constraints are expressed as follows:

$$\sum_{i=1}^n z_{ij} = 1, j = 1, \dots, n \quad (14)$$

$$\sum_{j=1}^n z_{ij} = 1, i = 1, \dots, n \quad (15)$$

The utility of agent i is therefore defined by $y_i = \sum_{j=1}^n u_{ij} z_{ij}$. Note that, due to constraint (15), this definition of y_i ensures that $y_i \in [0, 1]$. Hence, the vector of individual utilities is given by $y = (y_1, \dots, y_n) \in [0, 1]^n$. These variables y_i can be used in combi-

nation with Program 12-13, which yields the following overall MIP:

$$\begin{aligned} \max w & \\ \text{s.t.} & \begin{cases} \sum_{i=1}^n z_{ij} = 1 \quad j = 1, \dots, n \\ \sum_{j=1}^n z_{ij} = 1 \quad i = 1, \dots, n \\ y_i = \sum_{j=1}^n u_{ij} z_{ij} \quad i = 1, \dots, n \\ w \leq d_A \quad \forall A \subseteq [n] \\ d_A = \sum_{i \in A} p_i^A + b_0^A \overline{m}(A) \quad \forall A \subseteq [n] \\ \sum_{i \in A} b_i^A + b_0^A = 1 \quad \forall A \subseteq [n] \\ p_i^A \leq b_i^A \quad \forall A \subseteq [n] \quad \forall i \in A \\ p_i^A \leq y_i \quad \forall A \subseteq [n] \quad \forall i \in A \\ p_i^A \geq y_i + b_i^A - 1 \quad \forall A \subseteq [n] \quad \forall i \in A \end{cases} \\ z_{ij} & \in \{0, 1\} \quad \forall i, \forall j, \quad w \in \mathbb{R}^+, \quad y_i \in \mathbb{R}^+ \quad \forall i \\ d_A & \in \mathbb{R}^+ \quad \forall A \subseteq [n], \quad b_i^A \in \mathbb{R}^+ \quad \forall A \subseteq [n], \quad \forall i \in \{0\} \cup A \end{aligned}$$

Some numerical tests have been carried out using IBM ILOG CPLEX Optimization Studio 12.4 on a computer with 3.8 Gb of memory and an Intel Core 2 Quad 2.40 Ghz processor. We used an ordinal scale \mathbb{L} with $|\mathbb{L}| = 5$. Möbius masses corresponding to 2-minitive capacities were randomly drawn in \mathbb{L} , and the average computation times over 20 instances (for each size) were observed. The results showed that instances up to size $n = 20$ can be solved in a few seconds, while the solution of instances of size 25 requires more than 600 seconds (value of the timeout in our tests). This fast growth of the solution times goes with the $O(n^4)$ growth of the number of constraints in the MIP formulations: from 20 to 30 agents, the number of constraints increases from $\sim 160,000$ to $\sim 1,000,000$. Interestingly, for Möbius masses defined such that $S_v(y) = \min_i y_i$, instances up to 100 agents were solved in a few seconds. This tends to confort the intuition that the more the Sugeno integral differs from the min, the less effective the formulation becomes.

6 CONCLUSION

In this paper, we have studied the optimization of the Sugeno integral in ordinal assignment problems. Table 1 summarizes our contributions in the maximization case. We have identified special cases (weighted min, ordered weighted min) that admit a polynomial time solution procedure, and we have shown that it is NP-hard in the general case, even if the cardinality of the scale is only two. Finally, we have provided the first MIP formulation for the optimization of the Sugeno integral under linear constraints. The preliminary numerical tests carried out showed that the operability of this latter approach is limited to instances of modest size. Nevertheless, for future works, it would be interesting to study the possibility of other MIP formulations. It is indeed worth noting that an efficient linearization of the Sugeno integral would impact on a number of combinatorial problems other than the assignment problem studied here (e.g., minimum spanning tree problem, traveling salesman problem).

Capacity	Operator	Complexity	Algorithm
general	Sugeno	NP-hard	MIP
minitive	weighted min	$O(n^3 \log \mathbb{L})$	Threshold
maxitive	weighted max	$O(n^2)$	$\max_{i,j} (u_{ij} \wedge w_i)$
symmetric	OWMin, OWMax	$O(n^3 \log \mathbb{L})$	Threshold
k -maxitive	Sugeno	$O(n^{3+k} \log \mathbb{L})$	Threshold
k -minitive	Sugeno	?	compact MIP

Table 1: Summary of our contributions.

ACKNOWLEDGMENTS

Soufiane Drissi Oudghiri and Mohamed Hachimi's work was supported by the Centre National de la Recherche Scientifique et Technique (CNRST-Morocco). Patrice Perny and Olivier Spanjaard's work was supported by the ANR-14-CE24-0007-01 project "CoCoRiCo-CoDec".

REFERENCES

- [1] E.M. Bartee, 'Problem solving with ordinal measurement', *Management Science*, **17**(10), 622–633, (1971).
- [2] U. Bossong and D. Schweigert, 'Minimal paths on ordered graphs', Technical Report 24, Report in Wirtschaftsmathematik, Universität Kaiserslautern, (1996).
- [3] S. Bouveret, U. Endriss, and J. Lang, 'Fair division under ordinal preferences: Computing envy-free allocations of indivisible goods', in *European Conference on Artificial Intelligence (ECAI 2010)*, pp. 387–392. IOS Press, (2010).
- [4] S. Brams, P. Edelman, and P. Fishburn, 'Fair division of indivisible items', *Theory and Decision*, **5**(2), 147–180, (2004).
- [5] S. Brams and D. King, 'Efficient fair division – help the worst off or avoid envy?', *Rationality and Society*, **17**(4), 387–421, (2005).
- [6] A. Chateaneuf, M. Grabisch, and A. Rico, 'Modeling Attitudes toward Uncertainty through the Use of Sugeno Integral', *Journal of Mathematical Economics*, **44**(11), 1084–1099, (2008).
- [7] F. Della Croce, V. Th. Paschos, and A. Tsoukias, 'An improved general procedure for lexicographic bottleneck problems', *Op. Res. Letters*, **24**, 187–194, (1999).
- [8] D. Dubois and H. Fargier, 'Lexicographic refinements of sugeno integrals', in *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pp. 611–622. Springer, (2007).
- [9] D. Dubois and H. Prade, 'Weighted minimum and maximum operations in fuzzy set theory', **39**, 205–210, (1986).
- [10] D. Dubois and H. Prade, 'Possibility theory as a basis for qualitative decision theory', in *IJCAI*, volume 95, pp. 1924–1930, (1995).
- [11] D. Dubois, H. Prade, and A. Rico, 'Residuated variants of Sugeno integrals: Towards new weighting schemes for qualitative aggregation methods.', *Information Sciences*, **329**, 765–781, (2016).
- [12] D. Dubois, H. Prade, and R. Sabbadin, 'Qualitative decision theory with sugeno integrals', in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 121–128. Morgan Kaufmann Publishers Inc., (1998).
- [13] R. Fortet, 'Applications de l'algèbre de boole en recherche opérationnelle', *Revue Française de Recherche Opérationnelle*, **4**(14), 17–26, (1960).
- [14] P. Gardenfors, 'Assignment problem based on ordinal preferences', *Management Science*, **20**(3), 331–340, (1973).
- [15] J. Gorski and S. Ruzika, 'On k-max-optimization', *Operations Research Letters*, **37**(1), 23–26, (2009).
- [16] M. Grabisch, 'Some lexicographic approaches to the Sugeno integral', in *Proc. Int. Conf. on Information Processing and Management of Uncertainty (IPMU'06)*, pp. 572–579, (2006).
- [17] M. Grabisch and C. Labreuche, 'A decade of applications of the choquet and sugeno integrals in multi-criteria decision aid', *Annals of Operations Research*, **175**, 247–286, (2010).
- [18] M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap, *Aggregation Functions (Encyclopedia of Mathematics and its Applications)*, Cambridge University Press, New York, NY, USA, 1st edn., 2009.
- [19] O. Gross, 'The bottleneck assignment problem', Technical Report P-1630, The Rand Corporation, Sta. Monica, CA, (1959).
- [20] P. L. Hammer and S. Rudeanu, *Boolean methods in operations research and related areas*, volume 7, Springer Science & Business Media, 2012.
- [21] A. Junghanns and J. Schaeffer, 'Sokoban: Enhancing general single-agent search methods using domain knowledge', *Artificial Intelligence*, **129**(1), 219–251, (2001).
- [22] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, 'A practical, decision-theoretic approach to multi-robot mapping and exploration', in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 4, pp. 3232–3238. IEEE, (2003).
- [23] H.W. Kuhn, 'The hungarian method for the assignment problem', *Naval research logistics quarterly*, **2**(1-2), 83–97, (1955).
- [24] V. M. Malhotra, M. P. Kumar, and S. N. Maheshwari, 'An $O(|V|^3)$ algorithm for finding maximum flows in networks', *Information Processing Letters*, **7**(6), 277–278, (1978).
- [25] J.-L. Marichal, 'Weighted lattice polynomials', *Discrete Mathematics*, **309**(4), 814–820, (2009).
- [26] R. Mesiar, 'Generalizations of k-order additive discrete fuzzy measures', *Fuzzy Sets and Systems*, **102**, 423–428, (1999).
- [27] F.S. Roberts, *Measurement theory, with applications to Decision Making, Utility and the Social Sciences*, Addison-Wesley, Boston, 1979.
- [28] F.S. Roberts, 'Meaningfulness of conclusions from combinatorial optimization', *Discrete Applied Mathematics*, **29**, 221–241, (1990).
- [29] T. Sandholm and S. Suri, 'Side constraints and non-price attributes in markets', *Games and Economic Behavior*, **55**(2), 321–330, (2006).
- [30] David Schmeidler, 'Integral representation without additivity', *Proceedings of the American mathematical society*, **97**(2), 255–261, (1986).
- [31] D. Schweigert, 'Ordered graphs and minimal spanning trees', *Foundations of Computing and Decision Sciences*, **24**(4), 219–229, (1999).
- [32] P.T. Sokkalingam and Y.P. Aneja, 'Lexicographic bottleneck combinatorial problems', *Operations Research Letters*, **23**(1), 27–33, (1998).
- [33] M. Sugeno, *Theory of fuzzy integrals and its applications*, Ph.D. dissertation, Tokyo Institute of Technology, 1974.
- [34] A. Takaci, I. Stajner-Papuga, M. Maric, and D. Drakulic, 'A note on the use of choquet and sugeno integrals in minimal and maximal covering location problems', in *IEEE 12th International Symposium on Intelligent Systems and Informatics, SISIS 2014, Subotica, Serbia, September 11-13, 2014*, pp. 159–162, (2014).
- [35] V. Torra and Y. Narukawa, 'The-index and the number of citations: Two fuzzy integrals', *Fuzzy Systems, IEEE Transactions on*, **16**(3), 795–797, (2008).

Complexity Results for Probabilistic Datalog[±]

İsmail İlkan Ceylan¹Thomas Lukasiewicz²Rafael Peñaloza³

Abstract. We study the query evaluation problem in probabilistic databases in the presence of probabilistic existential rules. Our focus is on the Datalog[±] family of languages for which we define the probabilistic counterpart using a flexible and compact encoding of probabilities. This formalism can be viewed as a generalization of probabilistic databases, as it allows to generate new facts from the given ones, using so-called tuple-generating dependencies, or existential rules. We study the computational cost of this additional expressiveness under two different semantics. First, we use a conventional approach and assume that the probabilistic knowledge base is consistent and employ the standard possible world semantics. Thereafter, we introduce a probabilistic inconsistency-tolerant semantics, which we call inconsistency-tolerant possible world semantics. For both of these cases, we provide a thorough complexity analysis relative to different languages, drawing a complete picture of the complexity of probabilistic query answering in this family.

1 INTRODUCTION

Recent years have led to a significant increase in the number of application domains that generate large volumes of *uncertain data*. This has paved the way for a number of *systems* tailored towards such domains; most notably for large knowledge bases: Yago [22], Nell [31], DeepDive [36], Google’s Knowledge Vault [17], and Microsoft’s Probase [42] are systems containing a large amount of uncertain data. These systems are substantially based on the foundations of *probabilistic databases (PDBs)* [37]. Arguably, PDBs provide the state-of-the-art means for *modeling, storing, and processing* data in the presence of uncertainty.

Enriching databases with *ontological knowledge* is a common paradigm [33], as it allows one to deduce facts that are not explicitly specified in the database. The most widely studied languages for achieving such sophisticated data access are based on *description logics (DLs)* [2] and *existential rules* [9, 8]. Following this tradition, we study *probabilistic query entailment* under existential rules (tuple-generating dependencies) relative to a database. We focus on a particular family of existential rule languages, which is also referred to as Datalog[±] [9, 8].

Our framework is rather general: We assume a set of probabilistic events and annotate the facts and the rules with a Boolean expression formed over these events, which we call *contexts*. This context-based abstraction allows a compact specification of a probability distribution over the knowledge base. Similar approaches have been used in *knowledge representation* [32] and are also related to *data provenance* and *lineage* [23, 32, 37] in PDBs.

The most common semantics for PDBs is the *possible world semantics*: a PDB factorizes into a set of possible worlds, i.e., classical databases, each of which is then associated with a probability. This semantics is also used in probabilistic logic programming (see, e.g., ProbLog [16]) and is closely related to Poole’s independent choice logic [34]. We first study probabilistic query entailment in Datalog[±] under this semantics with a conventional assumption, i.e., the assumption that the probabilistic knowledge base is *consistent*.

Datalog[±] programs can clearly lead to inconsistencies, as negative constraints, such as $\forall x P(x) \wedge R(x) \rightarrow \perp$, are part of these programs. The obvious question is, of course, whether forcing the consistency assumption is always feasible? We answer this question negatively: PDBs are typically constructed in an automated manner; therewith, it is not easy to control which tuple is to be added to the database next. Suppose, e.g., that both atoms $P(u)$ and $R(u)$ are obtained with a positive probability. Clearly, adding both atoms would lead to an inconsistency, as the disjointness imposed by the rule will then be invalidated; i.e., we either throw away one of these atoms, or the whole knowledge base becomes inconsistent.

One way of tackling this problem is to simply ignore the inconsistent worlds imposed by the knowledge base, and as such, to slightly change the possible world semantics to only consider consistent worlds. We argue that this is not a solution to the problem, but rather a patch, and show that considering only consistent worlds could lead to loss of valuable information. In other words, inconsistent worlds may produce meaningful answers that are lost, as they can not be captured with an adequate semantics. Thus, to retrieve as much valuable information as possible, we base ourselves on the foundations of *inconsistency-tolerant* reasoning, which is well-understood both in the context of DLs [26, 5, 6] and Datalog[±] [28, 29, 27]. A well-known approach in inconsistency-tolerant reasoning is based on *repairing* the knowledge base by minimally removing some facts. As there can be many different minimal repairs (see the example above), the safe consequences are considered to be those that follow from *every* possible repair. In this paper, we adopt the *generalized repair (GR)* semantics from a recent work [18], which allows repairs both on the database and on the program. Based on the GR semantics, we define the *inconsistency tolerant possible world semantics*.

For both semantic approaches, we provide a thorough complexity analysis relative to different existential rule languages, drawing a complete picture of the complexity of probabilistic query entailment in Datalog[±]. The most central class for our complexity analysis is the class PP [20], which we describe in detail. Briefly stated, our results show an analogous behavior to the classical case, i.e., moving to inconsistency-tolerant semantics can put the complexity of reasoning one level higher in the respective hierarchy.

¹ TU Dresden, Germany, email: ceylan@tcs.inf.tu-dresden.de

² University of Oxford, UK, email: thomas.lukasiewicz@cs.ox.ac.uk

³ Free University Bozen-Bolzano, Italy, email: rafael.penaloza@unibz.it

2 MOTIVATION AND BACKGROUND

We enrich databases with ontological knowledge allowing to access probabilistic data over a logical abstraction. We concentrate on existential rule languages, also known as tuple-generating dependencies.

2.1 Existential Rules and Datalog[±]

We recall some basics on existential rules from the context of Datalog[±] [9, 8] and briefly introduce conjunctive query answering under existential rules.

General. Consider (possibly infinite) mutually disjoint sets \mathbf{R} of *predicates*, \mathbf{C} of *constants*, \mathbf{V} of *variables*, and \mathbf{N} of *nulls*. A *term* t is a constant, a null, or a variable. An *atom* is an expression of the form $P(t_1, \dots, t_n)$, where P is an n -ary predicate, and t_1, \dots, t_n are terms. A *variable-free atom* does not contain any variables as terms, and a *ground atom* is an atom that contains only constants as terms. An *instance* I is a (possibly infinite) set of variable-free atoms. A *database* \mathcal{D} is a finite set of ground atoms.

Programs. A *tuple-generating dependency (TGD)* (or *existential rule*) σ is a first-order formula $\forall \mathbf{x} \varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} P(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} \cup \mathbf{y} \subset \mathbf{V}$, $\varphi(\mathbf{x})$ is a conjunction of atoms, and $P(\mathbf{x}, \mathbf{y})$ is an atom; $\varphi(\mathbf{x})$ is the *body* of σ , denoted $body(\sigma)$, while $P(\mathbf{x}, \mathbf{y})$ is the *head* of σ , denoted $head(\sigma)$.⁴

A *negative constraint (NC)* ν is a first-order formula of the form $\forall \mathbf{x} \varphi(\mathbf{x}) \rightarrow \perp$, where $\mathbf{x} \subset \mathbf{V}$, $\varphi(\mathbf{x})$ is a conjunction of atoms, called the *body* of ν , denoted $body(\nu)$, and \perp denotes the truth constant *false*; i.e., a contradiction. A Datalog[±] *program* is a finite set Σ of TGDs and NCs. For brevity, we often omit the universal quantifiers in front of TGDs and NCs, and write simply, e.g., $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} P(\mathbf{x}, \mathbf{y})$. Moreover, we often speak simply of *programs* when referring to Datalog[±] programs.

Semantics. The semantics of programs is defined via homomorphisms. Briefly, a *homomorphism* is a substitution $h: \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ that behaves as the identity over \mathbf{C} . For a homomorphism h and a set of variables \mathbf{x} , we denote by $h|_{\mathbf{x}}$ the restriction of h to \mathbf{x} . The instance I *satisfies* the TGD σ , written $I \models \sigma$, if for every homomorphism h such that $h(\varphi(\mathbf{x})) \subseteq I$, there exists $h' \supseteq h|_{\mathbf{x}}$ such that $h'(P(\mathbf{x}, \mathbf{y})) \in I$. The instance I *satisfies* the NC ν , written $I \models \nu$, if there is no homomorphism h such that $h(\varphi(\mathbf{x})) \subseteq I$. Given a program Σ , I *satisfies* Σ , written $I \models \Sigma$, if I satisfies each TGD and NC of Σ . I is a *model* of the program Σ relative to the database \mathcal{D} , if $\mathcal{D} \subseteq I$ and $I \models \Sigma$. We denote the set of all models of Σ relative to \mathcal{D} as $mods(\mathcal{D}, \Sigma)$.

Unions of Conjunctive Queries. A *conjunctive query (CQ)* is an existentially quantified formula $\exists \mathbf{x} \psi(\mathbf{x})$, where ψ is a conjunction of atoms. Consider, e.g., the query

$$q_1(x) = \exists y \text{StarredIn}(x, y) \wedge \text{Mov}(y),$$

which asks for individuals that starred in a movie. Notice that x is a free variable in q_1 , also called an *answer variable*. A *Boolean conjunctive query (BQ)* is a CQ without any free variables. An example is the query

$$q_2 = \exists x, y \text{StarredIn}(x, y) \wedge \text{Mov}(y),$$

which asks whether there exists an individual that starred in a movie.

⁴ Notice that our definition of TGDs requires the head to contain only one atom. This restriction is made w.l.o.g., as a TGD with a conjunction of atoms in the head can be equivalently represented by a set of single-atom-headed TGDs [8].

A *union of Boolean conjunctive queries (UCQ)* Q is a disjunction of BQs. For notational convenience, we write Q to represent UCQs and q to represent BQs. If we consider queries with free variables, we make this explicit and write $Q(\mathbf{x})$, or $q(\mathbf{x})$, respectively.

Query Semantics. The answers to a CQ $q(\mathbf{x})$ over an instance I , denoted $q(I)$, is the set of all mappings Θ from \mathbf{x} to the constants in I such that $q(\Theta(\mathbf{x})) \in I$. A Boolean query q has a positive answer over I , denoted $I \models q$, if $q(I) \neq \emptyset$. Given a database \mathcal{D} and a program Σ , the answers we consider are those that are true in *all* models of Σ relative to \mathcal{D} . Formally, the *answer* to a CQ q w.r.t. \mathcal{D} and Σ is the set of tuples $ans(q, \mathcal{D}, \Sigma) = \bigcap_{I \in mods(\mathcal{D}, \Sigma)} \{t \mid t \in q(I)\}$. The answer to a BQ q is *positive*, denoted $\mathcal{D} \cup \Sigma \models q$, if $ans(q, \mathcal{D}, \Sigma) \neq \emptyset$. These notions are generalized to the class of UCQs in the obvious way. Consider a database

$$\mathcal{D} = \{\text{Actor}(\text{alPacino}), \text{StarredIn}(\text{pMiller}, \text{cw}), \text{Mov}(\text{cw})\},$$

which asserts that *Al Pacino* is an actor, and that *Penelope Miller* has starred in a movie. The query $q_1(x)$ produces only *pMiller* as an answer on \mathcal{D} . In the presence of the program

$$\Sigma = \{\{\text{Actor}(x) \rightarrow \exists y \text{StarredIn}(x, y), \text{Mov}(y)\}\},$$

a new tuple (*alPacino*) is generated, and thus both *alPacino* and *pMiller* become answers to $q_1(x)$.

2.2 Computational Properties of Existential Rules

In general, it is undecidable whether a BQ has an answer or not w.r.t. a database \mathcal{D} and a program Σ [4]. To regain decidability, many different restrictions on the class of allowed TGDs have been proposed. The most important (syntactic) restrictions studied in the literature are guardedness [8], stickiness [9], and acyclicity, along with their “weak” counterparts, namely weak guardedness [8], weak stickiness [9], and weak acyclicity [19], respectively.

A TGD σ is *guarded*, if there exists an atom $\mathbf{a} \in body(\sigma)$ that contains (or “guards”) all the body variables of σ . The class of guarded TGDs, denoted G , is defined as the family of all possible sets of guarded TGDs. A key subclass of guarded TGDs are the so-called linear TGDs with just one body atom, which is automatically the guard. The class of linear TGDs is denoted by L . *Weakly guarded* TGDs extend guarded TGDs by requiring only the body variables that are considered “harmful” to appear in the guard (see [8] for full details). The associated class of TGDs is denoted WG . It is easy to verify that $L \subset G \subset WG$, in terms of the sets of TGDs they contain.

Stickiness is inherently different from guardedness, and its central property can be described as follows: variables that appear more than once in a body (i.e., join variables) must always be propagated (or “stuck”) to the inferred atoms. A TGDs that enjoys this property is called *sticky*, and the class of sticky TGDs is denoted by S . Weak stickiness generalizes stickiness by considering only “harmful” variables, and defines the class WS of *weakly sticky* TGDs. Observe that $S \subset WS$.

A set Σ of TGDs is *acyclic* (and belongs to the class A), if its predicate graph is acyclic. Equivalently, an acyclic set of TGDs can be seen as a non-recursive set of TGDs. We say Σ is *weakly-acyclic*, if its dependency graph enjoys a certain acyclicity condition, which guarantees the existence of a finite canonical model; the associated class is denoted WA . Clearly, $A \subset WA$. Interestingly, it also holds that $WA \subset WS$ [9].

Another key fragment of TGDs, which deserves our attention, are the so-called *full* TGDs, i.e., TGDs without existentially quantified

variables. Their corresponding class is denoted as F. Restricting full TGDs to satisfy linearity, guardedness, stickiness, or acyclicity yields the classes LF, GF, SF, and AF, respectively. A known relation between these classes is that $F \subset WA$ [19] and $F \subset WG$ [8]. We extend all these notions to programs Σ in the obvious way: by considering the properties satisfied by the TGDs in Σ . Thus, for instance, Σ is guarded, if all the TGDs in Σ are guarded.

When analysing the complexity of query answering, we consider all these classes of programs unless explicitly mentioned otherwise. To obtain a fine-grained analysis of the computational complexity, we follow Vardi's taxonomy [40], as described next. The *combined complexity* of UCQ answering is calculated by considering all the components, i.e., the database, the program, and the query, as part of the input. The *bounded-arity combined complexity* (or simply *ba-combined complexity*) assumes that the arity of the underlying schema (i.e., the maximum arity of the predicates in \mathbf{R}) is bounded by an integer constant. In the context of description logics (DLs), the combined complexity in fact refers to the *ba-combined complexity*, since, by definition, the arity of the underlying schema is at most two. The *fixed-program combined complexity* (or simply *fp-combined complexity*) is calculated by considering the program (i.e., the set of TGDs and NCs) as fixed, while the *data complexity* additionally assumes that the query is fixed.

Table 1 summarizes the known complexity results for query entailment in the different classes of programs that we consider. These results will provide the basis for analysing the complexity of probabilistic Datalog[±] programs in the following sections.

	Data	Comb.	ba-comb.	fp-comb.
L, LF, AF	in AC ⁰	PSPACE	NP	NP
G	P	2EXP	EXP	NP
WG	EXP	2EXP	EXP	EXP
S, SF	in AC ⁰	EXP	NP	NP
F, GF	P	EXP	NP	NP
A	in AC ⁰	NEXP	NEXP	NP
WS, WA	P	2EXP	2EXP	NP

Table 1: Complexity of BQ answering [27]. All entries except for “in AC⁰” are completeness ones, where hardness in all entries but the *fp-combined* ones holds even for ground atomic BQs.

2.3 Complexity of Standard Probabilistic Inference

Our approach is based on annotating the facts in the database and the rules in the Datalog[±] program with Boolean events, which we call *contexts*. Here, we briefly introduce the basic notions, our assumptions, and the complexity of probabilistic Boolean inferences.

Consider a finite set of elementary events $\mathbf{E} = \{e_1, \dots, e_n\}$. A *world* is a conjunction $w = s_1 \wedge \dots \wedge s_n$ where $s_i, 1 \leq i \leq n$, is either the event e_i or its negation $\neg e_i$. A *context* is a Boolean combination of elementary events, i.e., if κ_1 and κ_2 are contexts, then so is $\neg \kappa_1$ and $\kappa_1 \wedge \kappa_2$.

Contexts encompass the probabilistic component of our formalism. For representing the probability distribution of events and contexts, we do not restrict to any specific probabilistic model, but rather consider any representation for which deciding whether $P(\kappa) > p$ for some value $p \in [0, 1]$ is PP-complete. Further details on the complexity class PP and its relation to other complexity classes can be found in Section 3.1.

3 PROBABILISTIC DATALOG[±]

To define our probabilistic extension of Datalog[±], we annotate all the rules and negative constraints with contexts, which will be interpreted through the probability distribution. Similarly, all the atoms in a probabilistic database are associated with a context as well.

Definition 1 (Probabilistic Datalog[±]) A *probabilistic TGD* is an expression of the form $\langle \sigma : \kappa \rangle$, where σ is a TGD, and κ is a context. Analogously, a *probabilistic negative constraint* is of the form $\langle \nu : \kappa \rangle$, where ν is a negative constraint, and κ is a context. A *probabilistic program* Γ is a finite set of probabilistic TGDs and probabilistic negative constraints.

A *probabilistic atom* is of the form $\langle \ell : \kappa \rangle$, where ℓ is an atom, and κ is a context. A *probabilistic database* \mathcal{P} is a finite set of probabilistic atoms. A *probabilistic knowledge base* is a pair $\mathcal{K} = (\Gamma, \mathcal{P})$ that represents a probabilistic program Γ relative to a probabilistic database \mathcal{P} .

We extend the special cases of Datalog[±] programs defined in the previous section to probabilistic programs in the obvious way. That is, the probabilistic program Γ is *guarded* if the Datalog[±] program $\{\lambda \mid \langle \lambda : \kappa \rangle \in \Gamma\}$ is guarded, and analogously for linear, sticky, acyclic, and full programs, and their weak versions. For a class \mathcal{L} of Datalog[±] programs, we denote by $\Upsilon_{\mathcal{L}}$ its associated class of probabilistic programs. Thus, for instance Υ_G is the class of all guarded probabilistic programs. Consider the probabilistic program Γ_m relative to the PDB \mathcal{P}_m given in Figure 1. It asserts that actors star in at least one movie and that actors and movies are disjoint entities. Both expressions hold in the *global context* \top . To ease reading, we usually omit the global context from the expressions.

Intuitively, a probabilistic program relative to a PDB compactly encodes a finite number of classical programs relative a classical database, each of which associated with a different context, and therefore a number of worlds. This semantics is commonly referred to as the *possible world semantics*.

Definition 2 (Possible Worlds) Let $\mathcal{K} = (\Gamma, \mathcal{P})$ be a probabilistic knowledge base. Every world w induces a classical knowledge base $\mathcal{K}_{|w} = (\Gamma_{|w}, \mathcal{P}_{|w})$ where

$$\begin{aligned} \Gamma_{|w} &= \{\lambda \mid \langle \lambda : \kappa \rangle \in \Gamma, w \models \kappa\}, \\ \mathcal{P}_{|w} &= \{\ell \mid \langle \ell : \kappa \rangle \in \mathcal{P}, w \models \kappa\}. \end{aligned}$$

A probabilistic knowledge base \mathcal{K} is *consistent*, if all the worlds induced by \mathcal{K} (with positive probability) are consistent.

The probabilistic program Γ_m relative to \mathcal{P}_m encodes exponentially many worlds on the size of the context variables. For instance, given the world w_1 (see Figure 1), $\mathcal{P}_{|w_1}$ contains all tuples from **Actors** and **Movies**, but none from **StarredIn**. Similarly, as the rules in Γ_m are global (i.e., they hold in every world), $\Gamma_{|w_1}$ contains both rules.

Observe also that *tuple-independent* probability models are a special case of our abstraction, where every annotation is independent from others. In this case, one can directly write probability values, instead of the contexts with their independent probabilities.

Definition 3 (Query Semantics) Let $\mathcal{K} = (\Gamma, \mathcal{P})$ be a probabilistic knowledge base, the *probability of a UCQ* Q is given by:

$$P_{\mathcal{K}}(Q) = \sum_{\mathcal{K}_{|w} \models Q} P(w),$$

Given a query Q and $p \in (0, 1]$, *probabilistic query entailment* is the problem of deciding whether $P_{\mathcal{K}}(Q) \geq p$.

Actor	Pr	Movies	Pr
alPacino	a_1	carlitosWay	m_1
rDeNiro	a_2	godfather	m_2
mPfeiffer	a_3	taxiDriver	m_3

StarredIn		Pr
alPacino	carlitosWay	$\neg s_1 \wedge s_2$
alPacino	godfather	$s_3 \wedge \neg s_4$
rDeNiro	godfather	$\neg s_5 \wedge s_6$
pMiller	carlitosWay	$\neg s_7 \wedge s_1$

Γ_m

$R_1 : \langle \text{Actor}(x) \rightarrow \exists y \text{StarIn}(x, y), \text{Mov}(y) \rangle$

$R_2 : \langle \text{Actor}(x), \text{Mov}(x) \rightarrow \perp \rangle$

#	Worlds	Pr
w_1	$\{ a_1, a_2, a_3, m_1, m_2, m_3, s_1, \dots, s_7 \}$.73
w_2	$\{ \neg a_1, a_2, a_3, m_1, m_2, m_3, s_1, \dots, s_7 \}$.11
w_312
w_425
...
w_n	$\{ \neg a_1, \neg a_2, \neg a_3, \neg m_1, \neg m_2, \neg m_3, \neg s_1, \dots, \neg s_7 \}$.01

Figure 1: The probabilistic database \mathcal{P}_m (depicted using tables) and the probabilistic program $\Gamma_m = \{R_1, R_2\}$ composed of a TGD (R_1) and an NC (R_2). The contexts are defined over the elementary events $\mathbf{E} = \{a_1, a_2, a_3, m_1, m_2, m_3, s_1, \dots, s_7\}$.

Briefly, a UCQ describes a desired pattern for a given knowledge base, and query entailment is then the task of deciding whether the specified pattern holds in this KB. Probabilistic query entailment factorizes this decision over different KBs, and we are interested in learning how likely it is for a UCQ to be entailed. Consider, for instance, the probabilistic KB $\mathcal{K}_m = (\Gamma_m, \mathcal{P}'_m)$, where

$$\mathcal{P}'_m = \{ \langle \text{Actor}(\text{alPacino}, \text{godfather} : 0.5) \rangle, \langle \text{Actor}(\text{rDeNiro}, \text{godfather} : 0.5) \rangle \},$$

and Γ_m given as before. The query q_2 would return the probability 0.75 on the program Γ_m relative to the PDB \mathcal{P}'_m . Notice that the only world that does not satisfy the query is $\{\text{Movie}(\text{gf})\} \cup \Gamma_m$, and this world has the probability 0.25. It is easy to see that q_2 would evaluate to 0, if it is posed only on \mathcal{P}'_m .

3.1 Complexity Classes and Assumptions

For the sake of readability, we briefly recall some of the non-standard complexity classes that we consider, and their relation to other classical complexity classes. The most typical counting complexity class

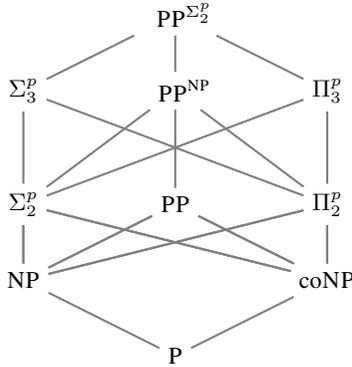


Figure 2: A portion of the counting polynomial-time hierarchy.

is #P [39], which is a functional complexity class originally introduced in the context of counting problems. The corresponding decision class PP [20] defines the set of languages recognized by a polynomially bounded non-deterministic Turing machine (TM) that accepts an input if and only if more than half of the computation paths are accepting [38]; such machines are usually called PP TMs. We also consider PP^{NP} (resp., $PP^{\Sigma_2^P}$, PP^{NEXP}), which as usual corresponds to languages that can be recognized by a PP TM, with an NP (resp., Σ_2^P , NEXP) oracle. Most of these classes belong to the counting polynomial-time hierarchy [41], which is partially illustrated in

Figure 2 along with the first levels of the polynomial hierarchy. The following relations between complexity classes are a consequence of the relationships depicted in Figure 2 and will also be useful throughout the rest of this paper:

$$PP^{\Sigma_2^P} \subseteq PSPACE \subseteq EXP \subseteq NEXP \subseteq P^{NEXP} \subseteq PP^{NEXP} \subseteq 2EXP$$

3.2 Complexity Results

We will consider the complexity of query answering w.r.t. the different classes of probabilistic programs relative to different languages.

	Data	Comb.	ba-comb.	fp-comb.
L, LF, AF	PP	PSPACE	PP^{NP}	PP^{NP}
G	PP	2EXP	EXP	PP^{NP}
WG	EXP	2EXP	EXP	EXP
S, SF	PP	EXP	PP^{NP}	PP^{NP}
F, GF	PP	EXP	PP^{NP}	PP^{NP}
A	PP	NEXP	NEXP	PP^{NP}
WS, WA	PP	2EXP	2EXP	PP^{NP}

Table 2: Complexity of probabilistic entailment

We start with a general result that provides some bounds for the complexity of query entailment in probabilistic KBs parameterized on the complexity of its classical counterpart.

Theorem 4 *Let \mathcal{L} be a class of Datalog[±] programs, and k be the complexity of query entailment in \mathcal{L} relative to databases. Then, probabilistic query entailment in $\Upsilon_{\mathcal{L}}$ relative to probabilistic databases is (i) k -hard, (ii) PP-hard, and (iii) in PP^k .*

PP-hardness follows from the hardness of standard probabilistic inference. The full proof shows a construction of a probabilistic KB upon which standard query entailment can be decided, which proves k -hardness.⁵ Membership to PP^k follows mainly from the observation that a probabilistic knowledge base is a factorized representation of exponentially many classical knowledge bases. Thus, it is possible to solve the problem (after properly adjusting the probabilities of the worlds) by deciding whether the majority of the oracle calls that decide classical query entailment return true.

We analyze the consequences of Theorem 4. Observe first that if k is a deterministic class that contains PP, then $PP^k = k$ and thus Theorem 4 directly provides tight complexity bounds. Notice that this is the case w.r.t. the combined complexities for all the classes except A. In the case of the class A, the complexity of query entailment

⁵ For ease of presentation, we excluded the proofs from the main text; for the interested reader, we refer to the appendix of this paper.

is complete w.r.t. to the class NEXP, and it is not known whether $\text{PP}^{\text{NEXP}} \subseteq \text{NEXP}$. We observe that the non-determinism in the oracle NEXP calls are used in a restricted fashion; this allows us to encode the problem into exponentially many NEXP TMs, which can be simulated with a NEXP TM.

Lemma 5 *Probabilistic query entailment in Υ_A relative to a probabilistic database is in NEXP w.r.t. the combined complexity.*

With the help of Lemma 5 and Theorem 4, we conclude that for all the rule languages, the complexity of probabilistic entailment remains the same w.r.t. the combined complexity (see the second column in Table 2). Clearly, this result transfers to the case where all events are assumed to be independent. Notice, however, that the implication of Theorem 4 is stronger, as it also yields tight complexity bounds for the languages G, WG, WS, and WA w.r.t. *ba*-combined complexity, as well as for the language WG w.r.t. *fp*-combined complexity. For the remaining languages, where $k = \text{NP}$, we prove Theorem 6.

Theorem 6 *If query entailment in \mathcal{L} relative to databases w.r.t. *ba*-combined (resp., *fp*-combined) complexity is NP-complete, then probabilistic query entailment in $\Upsilon_{\mathcal{L}}$ relative to probabilistic databases is PP^{NP} -complete w.r.t. *ba*-combined (resp., *fp*-combined) complexity.*

Membership in PP^{NP} is shown by a setting appropriate threshold values and iterating over nondeterministic oracle calls until this threshold value is exceeded. To show hardness for this class, we require more involved technical constructions. For these constructions, we use the M- \exists QBF problem [41].

Definition 7 (M- \exists QBF) Given an integer constant c and a partially quantified Boolean formula of the form

$$\Phi = \exists y_1 \dots y_m \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k,$$

where every ϕ_i is a clause over $\{x_1, \dots, x_l, y_1, \dots, y_m\}$ and $k, l, m \geq 1$; M- \exists QBF (Φ, c) is to decide whether for at least c of the truth assignments τ to x_1, \dots, x_l , the formula $\tau(\Phi)$ is true.

Note that M- \exists QBF is different from majority satisfiability, as here the threshold is set by an integer (not necessarily majority). M- \exists QBF is an PP^{NP} -complete problem even if the clauses ϕ_i are restricted to 3CNF [3].

The full proof constructs a probabilistic knowledge base $\mathcal{K}_{\Phi} = (\emptyset, \mathcal{P}_{\Phi})$ and a special query Q_{Φ} based on Φ . \mathcal{K}_{Φ} and Q_{Φ} together simulate the satisfiability conditions for the formula Φ . Moreover, the atoms in \mathcal{P}_{Φ} are associated with partial assignments over the variables $\{x_1 \dots x_{\ell}\}$. Notice that these are precisely the variables upon which we want to decide whether the number of assignments are at least c . This construction allows us to factorize the satisfiability problem over the variables $\{x_1 \dots x_{\ell}\}$ and thus to obtain the result. As \mathcal{K}_{Φ} uses an empty program, and all atoms are bounded in the arity by 3, we obtain tight complexity bounds for all entries in Table 2.

We have analyzed the complexity of probabilistic query entailment under the standard possible world semantics. Using novel constructions, we have provided tight complexity bounds for all languages under consideration. Table 2 shows our results. Next, we provide concrete examples on how the possible world semantics can be incompetent under certain conditions, and concentrate on a different semantics.

4 INCONSISTENCY HANDLING

Due to the presence of negative constraints, knowledge bases may contain contradictory knowledge. In fact, this has led to a quest of finding alternative semantics to be able deal with inconsistent knowledge in ontologies. Consider for example the knowledge base

$$\Sigma_{inc} = \{P(x), R(x) \rightarrow \perp\} \quad \text{and} \quad \mathcal{D}_{inc} = \{P(u), R(u), P(v)\}.$$

The NC requires the predicates P and R to be disjoint, but the database states that u belongs to both of them. Thus, the program has no model relative to the database. The fact that a knowledge base contains an inconsistency makes standard reasoning very problematic, as *anything* can be entailed from an inconsistent knowledge base (“*ex falso quodlibet*”) under the standard semantics. Consequently, one loses the ability of distinguishing between queries. From a technical perspective, the inconsistency problem immediately propagates to probabilistic extensions. Consider a probabilistic variant of our example; i.e., the KB $\mathcal{K}_{inc} = (\Gamma_{inc}, \mathcal{P}_{inc})$, where

$$\begin{aligned} \Gamma_{inc} &= \{ \langle P(x), R(x) \rightarrow \perp : 0.5 \rangle \} \quad \text{and} \\ \mathcal{P}_{inc} &= \{ \langle P(u) : 0.5 \rangle, \langle R(u) : 0.5 \rangle, \langle P(v) : 0.5 \rangle \}. \end{aligned}$$

Observe that \mathcal{K}_{inc} factorizes into worlds with positive probability that contain inconsistent knowledge. More concretely, it imposes 16 worlds, two of which are inconsistent, i.e., the ones that contain the NC together with both $\langle P(u) : 0.5 \rangle$ and $\langle R(u) : 0.5 \rangle$. Notice that, even though a vast majority of the worlds are consistent, the knowledge base as a whole is inconsistent, as it assigns a positive probability to an inconsistent world. It is possible to slightly change the possible world semantics to only consider consistent worlds by setting the probabilities of inconsistent worlds to 0 and renormalizing the probability distribution over the set of worlds accordingly. More precisely, assuming $\sum_{w \models \perp} P_{\mathcal{K}}(w) < 1$, we obtain the distribution:

$$P(w) = \begin{cases} 0 & \text{if } w \models \perp \\ P_{\mathcal{K}}(w) / (1 - \sum_{w' \models \perp} P_{\mathcal{K}}(w')) & \text{otherwise.} \end{cases}$$

Notice that this semantics assumes that the *error* is in the probability distribution; accordingly, it modifies the distribution. For our example, it yields a probability less than 0.5 for $P(v)$. Moreover, in the same example, $P(u)$ and $R(u)$ evaluate to the same probability value w.r.t. this semantics. This is not in line with the intuition; particularly, because it puts as much *responsibility* on $P(v)$ as much as it puts on the other tuples. On the other hand, assuming that the *error* is on the logical side, it is easy to see that responsibility needs to be shared only by the NC $\langle P(x), R(x) \rightarrow \perp \rangle$ and the tuples $\{P(u), R(u)\}$, since they serve as the source of inconsistency. Thus, it is more intuitive to expect the probability of $P(v)$ to remain 0.5, as it does not contribute to inconsistency in the logical sense.

The main question is then, how to identify the meaningful answers in inconsistent worlds. We base ourselves on the recent advances on inconsistency-tolerant reasoning developed for Datalog[±] [28, 29, 27] and provide an inconsistency-tolerant possible world semantics. We also show that under this semantics, $P(v)$ evaluates to exactly 0.5.

Several inconsistency-tolerant semantics have been proposed in the literature. One of the central semantics is first developed for relational databases [1] and then generalized as the AR semantics for several DLs [26]. The AR semantics is based on the key notion of a *repair*, which is a \subseteq -maximal consistent subset of the given database \mathcal{D} . Here, it is assumed that errors leading to inconsistencies are only contained in the data, but not in the program. In recent

work [18], authors allow errors also in the programs and introduce the *generalized repair (GR)* semantics, which allows to separate the program and the database into *hard* and *soft* parts, where the hard part is assumed to be fixed, and the soft part can be subject to repairs.

	Data	Comb.	ba-comb.	fp-comb.
$L_{\perp}, LF_{\perp}, AF_{\perp}$	coNP	PSPACE	Π_2^P	Π_2^P
G_{\perp}	coNP	2EXP	EXP	Π_2^P
WG_{\perp}	EXP	2EXP	EXP	EXP
$S_{\perp}, SF_{\perp}, F_{\perp}, GF_{\perp}$	coNP	EXP	Π_2^P	Π_2^P
A_{\perp}	coNP	P^{NEXP}	P^{NEXP}	Π_2^P
WS_{\perp}, WA_{\perp}	coNP	2EXP	2EXP	Π_2^P

Table 3: Complexity of GR-BQ entailment under existential rules [18]; all entries are completeness results. Hardness holds even in the case where the whole database is soft, and the whole program is hard.

Table 3 illustrates the complexity of query answering under this semantics, denoted GR-UCQ. For further details, we refer to [18]. We now extend the *generalized repair (GR)* semantics to probabilistic Datalog[±].

Definition 8 (Flexible programs and databases) A *flexible PDB* is a pair $\mathcal{P} = (\mathcal{P}_h; \mathcal{P}_s)$ of two PDBs \mathcal{P}_h and \mathcal{P}_s , denoted *hard* and *soft PDB*, respectively, while a flexible (probabilistic) program is a pair $\Gamma = (\Gamma_h; \Gamma_s)$ consisting of a finite set Γ_h of TGDs and NCs and a finite set Γ_s of TGDs, denoted *hard* and *soft program*, respectively.

Consider again the probabilistic KB $\mathcal{K}_{inc} = (\Gamma_{inc}, \mathcal{P}_{inc})$, where we would like to fix the whole program, and let the whole database be a soft PDB. This can be achieved by setting $\Gamma_{inc} = (\Gamma_h, \emptyset)$ and $\mathcal{P}_{inc} = (\emptyset, \mathcal{P}_s)$, where $\Gamma_h = \Gamma_{inc}$ and $\mathcal{P}_{inc} = \mathcal{P}_s$. This partition fixes the program and views the whole PDB as a soft database. The notion of generalized repair (GR) for flexible PDBs under flexible probabilistic programs is then given as follows.

Definition 9 (Generalized repair) A *generalized repair* of a flexible PDB $(\mathcal{P}_h; \mathcal{P}_s)$ and a flexible program $(\Gamma_h; \Gamma_s)$ is a probabilistic KB $\mathcal{K} = ((\Gamma_h; \Gamma'_s), (\mathcal{P}_h; \mathcal{P}'_s))$, where $\Gamma'_s \subseteq \Gamma_s$ and $\mathcal{P}'_s \subseteq \mathcal{P}_s$ such that (i) $(\Gamma_h \cup \Gamma'_s \cup \mathcal{P}_h \cup \mathcal{P}'_s)$ is consistent, and (ii) there is no $t \in (\Gamma_s \cup \mathcal{P}_s) / (\Gamma'_s \cup \mathcal{P}'_s)$ such that $(\Gamma_h \cup \Gamma'_s \cup \mathcal{P}_h \cup \mathcal{P}'_s \cup \{t\})$ is consistent. The set of all such repairs is denoted by $rep(\mathcal{K})$.

Clearly, there may be many \subset -maximal repairs for every world: Observe that $\mathcal{K}_w = \sum_{inc} \cup \mathcal{D}_{inc}$ is a world over \mathcal{K}_{inc} . Here, both $\mathcal{K}_w / \{P(u)\}$ and $\mathcal{K}_w / \{R(u)\}$ are \subset -maximal repairs. In this case, these are all possible repairs. The query semantics then considers the consequences that are entailed from every \subset -maximal repair, i.e., the *safe consequences*.

Definition 10 (Inconsistency tolerant query semantics) Let \mathcal{K} be a probabilistic KB, the *probability of a UCQ* Q is given by:

$$P_{\mathcal{K}}(Q) = \sum_{\mathcal{K}|_w \models_{GR} Q} P(w),$$

where $\mathcal{K}|_w \models_{GR} Q$ holds iff for all repairs $r \in rep(\mathcal{K}|_w)$, it holds that $r \models Q$. Given a query Q and $p \in (0, 1]$, *probabilistic GR-UCQ entailment* is to decide whether $P_{\mathcal{K}}(Q) \geq p$.

The implication of this semantics is clear. Consider again \mathcal{K}_{inc} : It is easy to verify that $P(v)$ is entailed from all repairs of all worlds. Therefore, it yields the probability 0.5 for $P(v)$, as desired.

	Data	Comb.	ba-comb.	fp-comb.
$L_{\perp}, LF_{\perp}, AF_{\perp}$	PP ^{NP}	PSPACE	PP ^{Σ_2^P}	PP ^{Σ_2^P}
G_{\perp}	PP ^{NP}	2EXP	EXP	PP ^{Σ_2^P}
WG_{\perp}	EXP	2EXP	EXP	EXP
$S_{\perp}, SF_{\perp}, F_{\perp}, GF_{\perp}$	PP ^{NP}	EXP	PP ^{Σ_2^P}	PP ^{Σ_2^P}
A_{\perp}	PP ^{NP}	in PP ^{NEXP}	in PP ^{NEXP}	PP ^{Σ_2^P}
WS_{\perp}, WA_{\perp}	PP ^{NP}	2EXP	2EXP	PP ^{Σ_2^P}

Table 4: Complexity of probabilistic GR-UCQ entailment under existential rules; all entries but the “in” ones are completeness results. Hardness holds even in the case where the whole database is soft and the whole program is hard.

4.1 Complexity Results

As before, we will consider the complexity of query answering w.r.t. the different classes of probabilistic programs relative to different languages. Observe first that Theorem 4 is rather general, and thus, all the results can be transferred to this semantics.

Corollary 11 Let k be the complexity of GR-UCQ entailment in \mathcal{L} relative to a databases; then probabilistic GR-UCQ entailment in $\Upsilon_{\mathcal{L}}$ relative to probabilistic databases is k -hard, PP-hard, and in PP ^{k} .

As before, this yields tight complexity bounds for languages where k is a deterministic class that contains PP. The language A_{\perp} requires a special attention, as the complexity of GR-UCQ entailment in A_{\perp} is P^{NEXP} -complete. We provide an upper bound for probabilistic GR-UCQ entailment.

Lemma 12 Probabilistic GR-UCQ entailment in Υ_A relative to probabilistic databases is in PP^{NEXP} w.r.t. the combined and ba-combined complexity.

Although we expect this problem to be complete for the class PP^{NEXP}, it is yet open whether this problem is PP^{NEXP}-hard. The results for ba-combined and fa-combined cases require a much more detailed analysis. We prove the following Theorem.

Theorem 13 Let $k = \Pi_2^P$ be the complexity of GR-UCQ entailment in the rule language \mathcal{L} relative to databases w.r.t. ba-combined (resp., fp-combined) complexity. Then, probabilistic GR-UCQ entailment in $\Upsilon_{\mathcal{L}}$ relative to a probabilistic databases is complete in the class PP ^{Σ_2^P} w.r.t. ba-combined (resp., fp-combined) complexity.

While upper bounds can be shown using analogous arguments as in the standard semantics, to be able to show hardness, we first define a problem that is complete for the class PP ^{Σ_2^P} , adopted from [41].

Definition 14 (M- $\forall\exists$ QBF) Given an integer constant c and a partially quantified Boolean formula of the form

$$\Phi = \forall y_1 \dots y_m \exists z_1 \dots z_n \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k,$$

where every ϕ_i is a clause over $\{x_1, \dots, x_l, y_1, \dots, y_m, z_1, \dots, z_n\}$ and $k, l, m, n \geq 1$; M- $\forall\exists$ QBF(Φ, c) is to decide whether for at least c of the truth assignments τ to x_1, \dots, x_l , the formula $\tau(\Phi)$ is true.

For an arbitrary instance of M- $\forall\exists$ QBF, where the clauses are restricted to 3CNF, we construct a probabilistic knowledge base that consists of a program that contains a single negative constraint. The reduction is correct and of polynomial size. As the KB provided is

in the intersection of the class of languages considered, we obtain hardness for all of these languages.

Our last result is about data complexities, for which GR-UCQ entailment is coNP-complete (except for the class WG_\perp , which has exponential data complexity). We prove the following Theorem.

Theorem 15 *If GR-UCQ entailment in \mathcal{L} relative to databases is coNP-complete (or NP-complete) in data complexity, then the data complexity of probabilistic GR-UCQ entailment in $\Upsilon_{\mathcal{L}}$ relative to probabilistic databases is PP^{NP} -complete.*

We use the canonical problem M- \exists QBF to show hardness. Observe, however, that this requires a very different construction than the one in Theorem 6, as here the query is fixed. This result concludes our complexity analysis for inconsistency-tolerant semantics; all of the results are summarized in Table 4.

5 RELATED WORK

Our work is closely related to probabilistic databases [37, 12]. In fact, probabilistic Datalog $^\pm$ can be seen as a generalization of PDBs. As in PDBs, query answering in probabilistic Datalog $^\pm$ is PP-hard. Notice the novel dichotomy result in tuple-independent PDBs that classifies the unions of conjunctive queries as either being safe (P), or unsafe (#P-hard) [13]. Identifying special cases (as in [24]) of probabilistic Datalog $^\pm$ that allows similar dichotomies is part of the future work. We also note the recent work on Open-World PDBs [10], which allows a more flexible representation for PDBs by providing default probability intervals for unknown facts; significantly, this extended setting preserves the full dichotomy result for UCQs.

The possible world semantics is widely employed in probabilistic logic programming [34, 25, 35], and probabilistic query answering has been studied in light-weight probabilistic ontology languages [11, 14, 24] before; see especially [30] for an overview of probabilistic ontology languages. Our approach differs in several aspects: First, we consider a family of existential rule languages that is known to be well-behaved and provide tight complexity bounds for reasoning in these languages for all of the cases except one. Second, our assumptions are rather flexible, as we do not require a specific probabilistic model. Lastly, we propose an inconsistency-tolerant semantics, based on [18], and study query evaluation under this semantics. Note that inconsistency-tolerant semantics have been studied in Datalog $^\pm$ [21] before, which we find closely related. Differently, we adopt a more general repair semantics, as we allow repairs both on the data and on the program, and it is possible to partition the knowledge base into hard and soft components. Finally, we consider the full Datalog $^\pm$ family (not only guarded rules), providing a complete picture of the complexity of query evaluation.

6 SUMMARY AND OUTLOOK

We have studied probabilistic query entailment in Datalog $^\pm$ under the standard possible world semantics and under an inconsistency-tolerant variant of it. We have shown that the inconsistency-tolerant semantics provides more information, while pushing the computational complexity of probabilistic query entailment higher in the counting polynomial-time hierarchy in many cases. The differences between these two semantic considerations represent yet another trade-off between retrieving more information on the one side and the increasing computational cost on the other side. Our analysis is purely complexity-theoretical, and it is an open research problem to

find special cases where efficient algorithms can be developed. Such algorithms can take the advantage of existing methods in knowledge compilation [15, 7], as performing operations on a pre-compiled structure is known to be very efficient.

ACKNOWLEDGEMENTS

This work was supported by the German Research Foundation (DFG) within RoSI (GRK 1907) and by the UK EPSRC grants EP/J008346/1, EP/L012138/1, and EP/M025268/1.

A PROOF SKETCH FOR THEOREM 4

We prove the result only w.r.t. the data complexity; the result w.r.t. the *ba*-combined, *fa*-combined, and combined complexity can be obtained using analogous arguments.

(Hardness) Let k be the data complexity of query entailment in \mathcal{L} . Probabilistic query entailment w.r.t. the data complexity is PP-hard, as it is already so in PDBs w.r.t. data complexity. Thus, we only need to show k -hardness w.r.t. data complexity. Suppose that probabilistic query entailment is not k -hard in $\Upsilon_{\mathcal{L}}$ w.r.t. the data complexity. Let Σ be an arbitrary program relative to an arbitrary database $\mathcal{D} = \{l_i \mid 1 \leq i \leq n\}$ and construct the probabilistic KB $\mathcal{K}' = (\Gamma', \mathcal{P}')$ where

$$\Gamma' = \{\lambda \mid \lambda \in \Sigma\}, \quad \mathcal{P}' = \{\langle l_i : 0.5 \rangle \mid l_i \in \mathcal{D}, 1 \leq i \leq n\}.$$

Clearly, this construction is polynomial, and given a query Q' , it is easy to see that

$$(\Sigma, \mathcal{D}) \models Q' \text{ holds iff } P(Q') > 0.5^n,$$

which implies that query answering in \mathcal{L} is not k -hard in the data complexity, which leads to a contradiction.

(Membership) We assume that the probability of each world is computable in polynomial time, that it is a rational number, and that the rational numbers of the probabilities of all worlds have the same denominator. As for membership in PP^k , intuitively, we first create multiples of each world (which then correspond to the nondeterministic branches of a Turing machine), so that the probability distribution over all thus generated worlds is the uniform distribution. Then, for thresholds properly below (resp., above) 0.5, we introduce artificial success (resp., failure) worlds (which correspond to other nondeterministic success (resp., failure) branches of a Turing machine), so that satisfying the resulting threshold corresponds to having a majority of success worlds. We thus only have to verify whether for the majority of the worlds, the query evaluates to true. As query evaluation is in k , the computation is overall in PP^k . \square

B PROOF SKETCH FOR LEMMA 5

Let Q be a UCQ, and $\mathcal{K} = (\Gamma, \mathcal{P})$ be an arbitrary probabilistic knowledge base where Γ is defined over Υ_A . By Definition 3, it suffices to decide whether $\sum_{\mathcal{K}|w \models Q} P(w) \geq p$. Let W be the set of all worlds w . Guess a subset $\{w_1, \dots, w_n\} \subseteq W$ and verify whether

$$(1) \sum_{\mathcal{K}|w_i} \models Q \text{ for all } \{w_i \mid 1 \leq i \leq n\} \text{ and } (2) \sum_1^n P(w_i) \geq p.$$

It is easy to see that this procedure yields the correct decision. We only need to show that this procedure is in NEXP. First, observe

that the guess is of size exponential and can be produced by a non-deterministic Turing machine that runs in exponential time. Second, the verification of (1) can be done in $(\text{EXP} \times \text{NEXP}) = \text{NEXP}$, as there are exponentially many worlds and $k = \text{NEXP}$. Finally, the verification of (2) can be done by traversing over exponentially many worlds and computing their probabilities. As the latter can be done in polynomial time, this verification is clearly in EXP. \square

C PROOF SKETCH FOR THEOREM 6

(Membership) This result is a consequence of Theorem 4, where k is set to NP.

(Hardness) To show hardness, we provide a reduction from the M- \exists QBF problem (Definition 7). Let the formula in 3CNF

$$\Phi = \exists y_1 \dots y_m \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k,$$

be a partially quantified Boolean formula defined over $V = \{x_1, \dots, x_l, y_1, \dots, y_m\}$, where every ϕ_i is a disjunction of three literals, and c is an integer constant. For every clause $\phi_i = \circ u_i \vee \circ v_i \vee \circ w_i$, define ground atoms $M_i(\nu_i(u_i), \nu_i(v_i), \nu_i(w_i))$, where ν_i is a truth assignment to the variables u_i, v_i, w_i that satisfies ϕ_i . Observe that, for every clause, the number of such assignments is bounded by 2^3 . The partial assignment $\nu_{|s_1 \dots s_n}$ denotes the restriction of ν to the variables $\{s_1, \dots, s_n\}$. We construct the probabilistic KB $\mathcal{K}_\Phi = (\emptyset, \mathcal{P}_\Phi)$, where

$$\mathcal{P}_\Phi = \left\{ \langle M_i(\nu_i(u_i), \nu_i(v_i), \nu_i(w_i)) : \nu_{| \{x_1, \dots, x_l\} \cap \{u_i, v_i, w_i\}} \rangle \mid \nu_i \models \phi_i, 1 \leq i \leq k \right\}.$$

Let the event space be defined over the x -variables such that every world has the probability 0.5^l . For the query

$$Q_\Phi = \exists x_1 \dots x_l, y_1 \dots y_m \bigwedge_{i=1}^k M_i(u_i, v_i, w_i),$$

we obtain the following reduction:

$$P_{\mathcal{K}_\Phi}(Q) \geq c \cdot 0.5^l \text{ iff M-}\exists\text{QBF}(\Phi, c) \text{ answers yes.}$$

Observe that the above reduction can clearly be done in polynomial time in the size of Φ , and that the resulting probabilistic program is empty, and the arity of all predicates in the PDB is 3. \square

D PROOF SKETCH FOR THEOREM 13

(Membership) This result is a consequence of Theorem 4 and Corollary 11, where k is set to Π_2^P , and the fact that $\text{PP}^{\Sigma_2^P} = \text{PP}^{\Pi_2^P}$.

(Hardness) We provide a reduction from an arbitrary instance of M- \forall QBF given in Definition 14. Let

$$\Phi = \forall y_1 \dots y_m \exists z_1 \dots z_n \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k,$$

be a partially quantified Boolean formula in 3CNF over $V = \{x_1, \dots, x_l, y_1, \dots, y_m, z_1, \dots, z_n\}$, where every clause ϕ_i is a disjunction of three literals, and c is an integer constant. For every clause $\phi_i = \circ u_i \vee \circ v_i \vee \circ w_i$, define ground atoms $M_i(\nu_i(u_i), \nu_i(v_i), \nu_i(w_i))$, where ν_i is a truth assignment to the variables u_i, v_i, w_i that satisfies ϕ_i . Observe that, for every clause, the number of such assignments is bounded by 2^3 . The partial

assignment $\nu_{|s_1 \dots s_n}$ denotes the restriction of ν to the variables $\{s_1, \dots, s_n\}$. Construct the probabilistic KB $\mathcal{K}_\Phi = (\Gamma_\Phi, \mathcal{P}_\Phi)$ where

$$\begin{aligned} \mathcal{P}_\Phi &= \left\{ \langle M_i(\nu_i(u_i), \nu_i(v_i), \nu_i(w_i)) : \nu_{| \{x_1, \dots, x_l\} \cap \{u_i, v_i, w_i\}} \rangle \mid \right. \\ &\quad \left. \nu_i \models \phi_i, 1 \leq i \leq k \right\} \\ &\quad \cup \left\{ \langle S(0, 1, i) \rangle, \langle S(1, 0, i) \rangle \mid 1 \leq i \leq m \right\}, \\ \Gamma_\Phi &= \{S(x, y, z) \wedge S(y, x, z) \rightarrow \perp\}. \end{aligned}$$

Here, all probabilistic facts are soft, and the NC is hard. Let the event space be defined over the x -variables such that every world has the probability 0.5^l . Then, for the query

$$Q_\Phi = \exists x_1 \dots x_l, y_1 \dots y_m z_1 \dots z_n \bigwedge_{i=1}^k M_i(u_i, v_i, w_i) \wedge \bigwedge_{i=1}^m S(y_i, z_i, i),$$

we obtain the reduction:

$$P_{\mathcal{K}_\Phi}(Q) \geq c \cdot 0.5^l \text{ iff M-}\exists\text{QBF}(\Phi, c) \text{ answers yes.}$$

Observe that the above reduction can be done in polynomial time in the size of Φ , that the resulting probabilistic program is fixed and consists only of one NC, and the arity of all predicates is at most 3. \square

E PROOF SKETCH FOR THEOREM 15

(Membership) This result is a consequence of Theorem 4 and Corollary 11, where k is set to coNP, and the fact that $\text{PP}^{\text{coNP}} = \text{PP}^{\text{NP}}$

(Hardness) We provide a reduction from the PP^{NP} -complete problem of, given a partially quantified Boolean formula

$$\Phi = \forall y_1 \dots y_m \phi_1 \vee \phi_2 \vee \dots \vee \phi_k,$$

over $V = \{x_1, \dots, x_l, y_1, \dots, y_m\}$, where every ϕ_i is a conjunction of three literals, and an integer constant c , deciding whether for at least c truth assignments τ to $x_1 \dots x_l$, the formula

$$\forall y_1 \dots y_m \tau(\phi_1) \vee \tau(\phi_2) \vee \dots \vee \tau(\phi_k)$$

is true. Let $\phi_i = \circ u_{i,1} \wedge \circ u_{i,2} \wedge \circ u_{i,3}$. We define the PDB \mathcal{P}_Φ that contains the deterministic tuples

$$\langle M(u_{i,1}, u'_{i,1}, u_{i,2}, u'_{i,2}, u_{i,3}, u'_{i,3}, i) \rangle$$

such that $u'_{i,j} = 1$ (resp., $u'_{i,j} = 0$), if $u_{i,j}$ occurs positively (resp., negatively) in ϕ_i , $1 \leq i \leq k$, $1 \leq j \leq 3$. Furthermore, we add the soft probabilistic facts $\langle S(0, 1, x_i) : \neg x_i \rangle$ and $\langle S(1, 0, x_i) : x_i \rangle$ such that $i \in \{1, \dots, l\}$, and the soft probabilistic facts $\langle S(0, 1, y_i) \rangle$ and $\langle S(1, 0, y_i) \rangle$ such that $i \in \{1, \dots, m\}$.

We define the program $\Gamma_\Phi = \{S(x, y, z) \wedge S(y, x, z) \rightarrow \perp\}$, consisting of one hard NC, and therewith the probabilistic KB $\mathcal{K}_\Phi = (\Gamma_\Phi, \mathcal{P}_\Phi)$. The event space is defined over the x -variables such that every world has the probability 0.5^l . Then, for the query Φ

$$\begin{aligned} \exists \dots M(u_i, a_i, v_i, b_i, w_i, c_i, i) \wedge \\ S(a_i, a'_i, u_i) \wedge S(b_i, b'_i, v_i) \wedge S(c_i, c'_i, w_i), \end{aligned}$$

we obtain that $P_{\mathcal{K}_\Phi}(Q) \geq c \cdot 0.5^l$ iff for at least c truth assignments τ to $x_1 \dots x_l$, the formula

$$\forall y_1 \dots y_m \tau(\phi_1) \vee \tau(\phi_2) \vee \dots \vee \tau(\phi_k)$$

is true. This reduction can clearly be done in polynomial time in the size of Φ , the resulting probabilistic program consists of exactly one NC and is fixed, and the query is also fixed. \square

REFERENCES

- [1] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki, ‘Consistent query answers in inconsistent databases’, in *Proc. PODS*, pp. 68–79. ACM Press, (1999).
- [2] *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, Cambridge University Press, 2nd edn., 2007.
- [3] Delbert D. Bailey, Víctor Dalmau, and Phokion G. Kolaitis, ‘Phase transitions of PP-complete satisfiability problems’, *Discrete Appl. Math.*, **155**(12), 1627–1639, (2007).
- [4] Catriel Beeri and Moshe Y. Vardi, ‘The implication problem for data dependencies’, in *Proc. IICALP*, volume 115 of *LNCS*, pp. 73–85. Springer, (1981).
- [5] Meghyn Bienvenu, ‘On the complexity of consistent query answering in the presence of simple ontologies’, in *Proc. AAI*. AAAI Press, (2012).
- [6] Meghyn Bienvenu and Riccardo Rosati, ‘Tractable approximations of consistent query answering for robust ontology-based data access’, in *Proc. IJCAI*, (2013).
- [7] Guy Van Den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt, ‘Lifted probabilistic inference by first-order knowledge compilation’, in *Proc. IJCAI*, pp. 2178–2185, (2011).
- [8] Andrea Cali, Georg Gottlob, and Michael Kifer, ‘Taming the infinite chase: Query answering under expressive relational constraints’, *J. Artif. Intell. Res.*, **48**, 115–174, (2013).
- [9] Andrea Cali, Georg Gottlob, and Andreas Pieris, ‘Towards more expressive ontology languages: The query answering problem’, *Artif. Intell.*, **193**, 87–128, (2012).
- [10] İsmail İlkan Ceylan, Adnan Darwiche, and Guy Van Den Broeck, ‘Open-world probabilistic databases’, in *Proc. KR*, pp. 339–348. AAAI Press, (2016).
- [11] İsmail İlkan Ceylan and Rafael Peñaloza, ‘Probabilistic query answering in the Bayesian description logic \mathcal{BEL} ’, in *Proc. SUM*, volume 9310 of *LNAI*, pp. 21–35. Springer, (2015).
- [12] Nilesh Dalvi, Christopher Ré, and Dan Suciu, ‘Probabilistic databases: diamonds in the dirt’, *Commun. ACM*, **52**(7), 86–94, (2009).
- [13] Nilesh Dalvi and Dan Suciu, ‘The dichotomy of probabilistic inference for unions of conjunctive queries’, *J. ACM*, **59**(6), 1–87, (2012).
- [14] Claudia D’Amato, Nicola Fanzini, and Thomas Lukasiewicz, ‘Tractable reasoning with Bayesian description logics’, in *Proc. SUM*, volume 5291 of *LNAI*, pp. 146–159. Springer, (2008).
- [15] Adnan Darwiche and Pierre Marquis, ‘A knowledge compilation map’, *J. Artif. Intell. Res.*, **17**, 229–264, (2011).
- [16] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen, ‘ProbLog: A probabilistic Prolog and its application in link discovery’, in *Proc. IJCAI*, pp. 2468–2473. Morgan-Kaufmann, (2007).
- [17] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Patrick Murphy, Thomas Strohmman, Shaohua Sun, and Wei Zhang, ‘Knowledge Vault: A Web-scale approach to probabilistic knowledge fusion’, in *Proc. KDD*, pp. 601–610. ACM Press, (2014).
- [18] Thomas Eiter, Thomas Lukasiewicz, and Livia Predoiu, ‘Generalized consistent query answering under existential rules’, in *Proc. KR*. AAAI Press, (2016).
- [19] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa, ‘Data exchange: Semantics and query answering’, *Theor. Comput. Sci.*, **336**(1), 89–124, (2005).
- [20] John T. Gill, ‘Computational complexity of probabilistic Turing machines’, *SIAM J. Comput.*, **6**(4), 675–695, (1977).
- [21] Georg Gottlob, Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari, ‘Query answering under probabilistic uncertainty in Datalog+/- ontologies’, *Ann. Math. Artif. Intell.*, **69**(1), 37–72, (2013).
- [22] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum, ‘YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia’, in *Proc. IJCAI*, pp. 3161–3165, (2013).
- [23] Tomasz Imielinski and Witold Lipski, ‘Incomplete information in relational databases’, *J. ACM*, **31**(4), 761–791, (1984).
- [24] Jean Christoph Jung and Carsten Lutz, ‘Ontology-based access to probabilistic data with OWL QL’, in *Proc. ISWC*, volume 7649 of *LNCS*, pp. 182–197. Springer, (2012).
- [25] Marta Kwiatkowska, Gethin Norman, and David Parker, ‘PRISM: Probabilistic symbolic model checker’, *Computer Performance Evaluation Modelling Techniques and Tools*, **2324**, 200–204, (2002).
- [26] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo, ‘Inconsistency-tolerant semantics for description logics’, in *Proc. RR*, pp. 103–117, (2010).
- [27] Thomas Lukasiewicz, Maria Vanina Martinez, Andreas Pieris, and Gerardo I. Simari, ‘From classical to consistent query answering under existential rules’, in *Proc. AAI*, pp. 1546–1552. AAAI Press, (2015).
- [28] Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari, ‘Inconsistency handling in Datalog+/- ontologies’, in *Proc. ECAI*, pp. 558–563, (2012).
- [29] Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari, ‘Complexity of inconsistency-tolerant query answering in Datalog+/-’, in *Proc. ODBASE*, pp. 488–500, (2013).
- [30] Thomas Lukasiewicz and Umberto Straccia, ‘Managing uncertainty and vagueness in description logics for the Semantic Web’, *J. Web Sem.*, **6**(4), 291–308, (2008).
- [31] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Tahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling, ‘Never-ending learning’, in *Proc. AAI*, pp. 2302–2310. AAAI Press, (2015).
- [32] Thomas Rölleke Norbert Fuhr, ‘A probabilistic relational algebra for the integration of information retrieval and database systems’, *ACM Trans. Inf. Syst.*, **15**(1), 32–66, (1997).
- [33] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati, ‘Linking data to ontologies’, *J. Data Sem.*, **10**, 133–173, (2008).
- [34] David Poole, ‘The independent choice logic for modelling multiple agents under uncertainty’, *Artif. Intell.*, **94**(1-2), 7–56, (1997).
- [35] Joris Renkens, Dimitar Shterionov, Guy Van den Broeck, Jonas Vlasselaer, Daan Fierens, Wannes Meert, Gerda Janssens, and Luc De Raedt, ‘ProbLog2: From probabilistic programming to statistical relational learning’, *Proc. NIPS*, 1–5, (2012).
- [36] Jaeho Shin, Feiran Wang, Christopher De Sa, Ce Zhang, and Sen Wu, ‘Incremental knowledge base construction using DeepDive’, in *Proc. VLDB*, volume 8, (2015).
- [37] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. Probabilistic Databases, 2011.
- [38] Jacobo Torán, ‘Complexity classes defined by counting quantifiers’, *J. ACM*, **38**(3), 753–774, (1991).
- [39] Leslie Gabriel Valiant, ‘The complexity of computing the permanent’, *Theor. Comput. Sci.*, **8**(2), 189–201, (1979).
- [40] Moshe Y. Vardi, ‘The complexity of relational query languages’, *J. ACM*, 137–146, (1982).
- [41] Klaus W. Wagner, ‘The complexity of combinatorial problems with succinct input representation’, *Acta Inf.*, **23**(3), 325–356, (1986).
- [42] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu, ‘Probase: A probabilistic taxonomy for text understanding’, *Proc. SIGMOD*, 481–492, (2012).

Strategic Voting in a Social Context: Considerate Equilibria

Laurent Gourvès and Julien Lesca and Anaëlle Wilczynski¹

Abstract. In a voting system, voters may adopt a strategic behaviour in order to manipulate the outcome of the election. This naturally entails a game theoretic conception of voting. The specificity of our work is that we embed the voting game into a social context where agents and their relations are given by a graph, i.e. a social network. We aim at integrating the information provided by the graph in a refinement of the game-theoretical analysis of an election. We consider coalitional equilibria immune to deviations performed by realistic coalitions based on the social network, namely the cliques of the graph. Agents are not fully selfish as they have consideration for their relatives. The corresponding notion of equilibrium was introduced by Hofer *et al.* [12] and called *considerate equilibrium*. We propose to study its existence and the ability of the agents to converge to such an equilibrium in strategic voting games using well-known voting rules: Plurality, Antiplurality, Plurality with runoff, Borda, k -approval, STV, Maximin and Copeland.

1 Introduction

The way of aggregating the preferences of a society in order to make a collective decision is a fundamental issue in every part of the community life. When several alternatives, or candidates, are available, voting systems are classically used to make collective decisions in many different contexts (political elections, decisions within committees, planning of meetings). In this respect, social choice theorists designed various voting rules whose properties have been analyzed in an axiomatic manner (see e.g. [18, 5] for an overview on that topic).

Most of these works implicitly assume that voters truthfully reveal their preferences. However, there is no possibility to ensure that they tell the truth when they vote. Voters can think they will be better off if they do not exactly align their ballot with their preferences (tactical voting), what is called *manipulation*. This situation can lead to an outcome that is more favorable to the manipulators. However, one may lose the good properties of the voting rule (as underlined in [8]). It is highly desirable for a voting rule to be immune to such strategic behaviors. Unfortunately, the Gibbard-Satterthwaite theorem [10, 25] shows that any reasonable non-dictatorial voting system is manipulable.

One way to circumvent this impossibility result is to resort to voting rules which are computationally hard to manipulate. This computational complexity may constitute a barrier to malicious behaviours. However this approach has raised many criticisms in the community of computational social choice [6, 7]. The main reason is that concepts used to define computational complexity are based on worst case analysis, and there may be various voting situations where a manipulation is easy to compute. Another approach consists in perceiving an election as a strategic situation where the voters are the

players of a game. In this case, a plausible outcome of the election is a situation of equilibrium: those who can dethrone the current winner prefer the winner to its possible contenders.

An important line of research consists in analyzing the existence of an equilibrium and the ability of the participants to reach it in a natural dynamic process where there is a deviation only if it is profitable. These works have contributed to observe that the guaranteed existence of an equilibrium, together with the convergence of the dynamics to such a stable state, depends on the voting rule, the initial state and the type of deviations that are allowed. A fundamental work about specific solution concepts in voting is due to Myerson and Weber [19] who define a voting equilibrium as a state consistent with a preliminary poll. More classically, the existence of a Nash equilibrium [20] — a stable outcome regarding unilateral deviations — has been well studied, especially around an axiomatic study of voting rules which admit it, see e.g. [15].

Recently, many articles deal with convergence to a Nash equilibrium through an *iterative voting* framework [1], a voting process consisting of several rounds, where at each round one voter is allowed to change her ballot. As an illustration, one can think of online voting procedures like *Doodle* polls (online tool to schedule meetings). Convergence for several voting rules has been established in this way in [14, 16, 24] but convergence is far from being guaranteed, conducting to reconsider the dynamic process with restricted deviations [11, 21, 23]. As far as we know, iterative voting has never been adapted to solution concepts different from the Nash equilibrium.

A drawback of the Nash equilibrium is its weakness against coalitional deviations. Actually, it is interesting to consider coalitions over the agents, modeling for instance some political parties or groups of friends. Well-known solution concepts take into account coalitional deviations, e.g. the *strong equilibrium* [3] and the *super-strong equilibrium* [29]. In these equilibria, every subset of agents is a possible coalition. Unfortunately, the existence of such equilibria cannot be guaranteed for most of the classical voting rules. The conditions of existence of a strong equilibrium in a voting game have been explored for instance in [22]. A characterization is provided by Sertel and Sanver [26] with an axiomatic point of view, as well as in [17] for specific voting rules. In another type of equilibrium called *coalition-proof equilibrium* [4], not every type of deviation is allowed.

Considering that any coalition of voters can form may be questionable or unrealistic, because it supposes that all agents in any subset are able to coordinate their moves in order to manipulate. Indeed, such a coordination requires a high level of communication and trust for the manipulators. In practice, one can reasonably exclude some coalitions from the definition of an equilibrium that is stable against group deviations. For example, this is the case of the *partition equilibrium* introduced by Feldman and Tennenholtz [9] where only the coalitions belonging to a prescribed partition of the voter set have the ability to deviate.

¹ Université Paris Dauphine, PSL Research University, CNRS, UMR [7243], LAMSADE, 75016 PARIS, FRANCE email: name.surname@dauphine.fr

More generally, one can determine which coalitions can form by exploiting social networks. Exploiting the social relations which bind the members of group is receiving much attention [13]. In a strategic game context, we can make the same observation and consider that the agents are embedded in a social network and then, relations among them are fully characterized by their links in that network. To go further, one can suppose that a voter is tied by social relations which force her to have consideration for other participants. Consequently, we can assume that an agent is not only guided by her own preferences over the candidates (or alternatives) but pursues the goal of optimizing the welfare of the communities where she belongs, as in [2]. A solution concept which naturally arises from this situation is the one where the social network is represented by a graph G whose node set and the agent set coincide. The possible coalitions are prescribed patterns of G and agents have consideration for their neighbors in the graph. When considering cliques of G as coalitions, such an equilibrium is called a *considerate equilibrium*. Introduced by Hofer et al. [12], a considerate equilibrium is a state robust to deviations by the cliques of the social network, but a coalition given by a clique only deviates when it is not harmful for her relatives (neighbors in the graph). As for the partition equilibrium that it extends, the considerate equilibrium has only been studied, as far as we know, for a special case of congestion game.

To our best knowledge, the social context in which the voters are embedded has been surprisingly underestimated in strategic voting games. Some recent papers started to investigate this question, as in [27, 28] where the authors study iterative voting via social networks. They consider that voters are not aware of every voter's ballot but only of the ballot of their neighbors in the social network. Thus, they use social relations from the perspective of a gain of information, but not in terms of defining which coalitions can form. In this article, we propose to fill this gap.

Concretely, we explore the existence of a considerate equilibrium in a strategic voting game and the ability of the game to converge to such an equilibrium for different voting rules. Our main contributions are existence proofs of a considerate equilibrium in a voting game under well-established voting rules, namely Plurality, Antiplurality, Plurality with runoff, STV and Maximin. We also investigate the possibility for the voters to reach a considerate equilibrium in a natural iterative process. In this respect, our results are rather negative because convergence to a partition equilibrium, or convergence to a Nash equilibrium, which are less demanding goals than convergence to a considerate equilibrium, fail.

The article is organized as follows. We first introduce in Section 2 the strategic voting game framework and the notion of considerate equilibrium. Then we study in Section 3 the case of positioning scoring rules such as Plurality and Antiplurality rules. Section 4 is devoted to two voting rules with runoff and in Section 5, two voting rules based on pair-wise comparisons of the alternatives (Copeland and Maximin) are studied. We conclude in Section 6 with a discussion on the global results and the impact of consideration within deviating moves.

2 Strategic voting games

We consider a strategic game defined on the basis of an election (N, M, \mathcal{F}) where N is a set of $n \geq 2$ voters (also called agents), M is a set of $m \geq 2$ candidates (also called alternatives) and \mathcal{F} is a voting rule. Each voter is a player. All players have the same strategy space S and every element of S is a possible ballot. A strategy profile σ (also called state) is a member of S^n where σ_i designates

the strategy adopted by player $i \in N$. In general, the voting rule is a mapping $\mathcal{F} : S^n \rightarrow 2^M$. In this article we restrict ourselves to voting rules that output a singleton, i.e. $\mathcal{F} : S^n \rightarrow M$. In the game, the elected candidate is $\mathcal{F}(\sigma)$ for σ being the players' strategy profile.

The preference relation that a player i has over M is expressed with a weak linear order \succsim_i . For x and $y \in M$, $x \succsim_i y$ means that player i values x at least as much as y . We write $x \sim_i y$ to say that player i is indifferent between x and y . The strict part of \succsim_i is denoted by \succ_i . We denote by $\mathcal{L}(M)$ (resp., $L(M)$) the set of all possible weak linear orders (resp., linear orders) over M .

We have $\succsim_i \in \mathcal{L}(M)$ for each player $i \in N$ and $\succsim = (\succsim_1, \dots, \succsim_n) \in \mathcal{L}(M)^N$ is called the *truthful profile*, i.e. the profile of the true preferences of the players. We say that the preferences are strict if the truthful profile belongs to $L(M)^N$.

The way the strategy set S is defined depends on the voting rule. A ballot can be a full ranking of the candidates or an unordered subset of M . In the voting game, a player can be insincere and she may strategically report a ballot that does not reflect her true preferences.

2.1 Solution concepts

The focus is on pure strategies and we consider that the players indirectly evaluate a strategy profile, in a sense that they have preferences over $\mathcal{F}(\sigma)$ instead of σ . In this article we analyze plausible outcomes of the game, namely the states (strategy profiles) which are at equilibrium. An equilibrium is a state that is immune to a predefined set of possible moves (also called deviations). A single player or a group of players (also called coalition) may deviate. Individual preferences extend to collective preferences for each group C of players. For x and $y \in M$, we have

- $x \succ_C y \Leftrightarrow [\forall i \in C, x \succ_i y]$,
- $x \geq_C y \Leftrightarrow [\forall i \in C, x \succsim_i y]$ and $[\exists j \in C, x \succ_j y]$,
- $x \succeq_C y \Leftrightarrow [\forall i \in C, x \succsim_i y]$,
- $x \sim_C y \Leftrightarrow [\forall i \in C, x \sim_i y]$.

Let us give two notions related to the Pareto dominance. An alternative $x \in M$ is said to be *undominated* (resp., *strictly undominated*) over a coalition $C \subseteq N$ if there does not exist any alternative y such that $y \geq_C x$ (resp., $y \succeq_C x$).

Example 1 Take four alternatives $\{a, b, c, d\}$ and three players $\{1, 2, 3\}$ with the following preferences.

- 1 $(a \sim b) \succ c \succ d$
- 2 $(a \sim b) \succ d \succ c$
- 3 $d \succ (a \sim b) \succ c$

For $C = \{1, 2, 3\}$, alternative d is strictly undominated, a and b are undominated but not strictly undominated, and c is dominated.

An undominated alternative always exists for any non-empty coalition $C \subseteq N$ but strictly undominated alternatives may be absent.

Let \mathcal{C} be a non-empty family of coalitions of N , i.e. $\mathcal{C} \subseteq 2^N \setminus \emptyset$; it is the collection of coalitions that can possibly deviate. In the following, σ_C and σ_{-C} stand for the restriction of σ to the strategy adopted by the players of C and $N \setminus C$, respectively. Thus, (σ'_C, σ_{-C}) denotes σ in which σ_i is replaced by σ'_i iff $i \in C$.

Definition 1 (Improving move (IM)) For a coalition $C \in \mathcal{C}$, an improving move from a state σ is a joint strategy $\sigma'_C \in S^{|C|}$ such that $\mathcal{F}(\sigma'_C, \sigma_{-C}) \succ_C \mathcal{F}(\sigma)$.

In an improving move, a coalition deviates only if each member strictly prefers the new state. This type of deviation can be relaxed by considering *weak improving moves*.

Definition 2 (Weak improving move (WIM)) For a coalition $C \in \mathcal{C}$, a weak improving move from a state σ is a joint strategy $\sigma'_C \in S^{|C|}$ such that $\mathcal{F}(\sigma'_C, \sigma_{-C}) \geq_C \mathcal{F}(\sigma)$.

Weak improving moves are appealing because they allow the participation of some players who do not benefit, but the deviation cannot harm its instigators.

Interestingly, we can use a pair (\mathcal{C}, μ) for $\mu \in \{\mathbf{IM}, \mathbf{WIM}\}$ to define a notion of equilibrium: a state σ is a (\mathcal{C}, μ) -equilibrium if no coalition $C \in \mathcal{C}$ can deviate from σ with a move of type μ . Thus, a *Nash equilibrium* (NE) [20] is a $(\mathcal{C}, \mathbf{IM})$ -equilibrium where $\mathcal{C} = \{\{i\} : i \in N\}$. Similarly a *strong equilibrium* (SE) [3] is a $(\mathcal{C}, \mathbf{IM})$ -equilibrium where $\mathcal{C} = 2^N \setminus \emptyset$. A *super strong equilibrium* (SSE) [29, 9] is a $(\mathcal{C}, \mathbf{WIM})$ -equilibrium where $\mathcal{C} = 2^N \setminus \emptyset$. These three solutions concepts are linked as follows: $\text{NE} \supseteq \text{SE} \supseteq \text{SSE}$.

On one hand, the strong equilibrium and the super strong equilibrium are more sustainable than the Nash equilibrium because they preclude a larger set of possible deviations. On the other hand, (super) strong equilibria are less likely to exist than Nash equilibria.

An argument against the (super) strong equilibrium is that in many situations, not all coalitions are conceivable. In this article we consider a special collection of coalitions \mathcal{C} that takes into account the social context in which the players are embedded. Given a social network of voters represented by a graph G with node set N and edge set E , an edge $(u, v) \in E$ indicates that players u and v are related, e.g. they have the possibility to communicate, so these two players can participate in a deviating coalition. As done in [9, 12], it makes sense to define the cliques of graph G as the possible coalitions. In that case $\mathcal{C} = \{C \in 2^N \mid \forall i, j \in C, (i, j) \in E\}$ and each coalition $C \in \mathcal{C}$ has a set of neighbors $\mathcal{N}(C) = \{i \in N \setminus C \mid \exists j \in C \text{ such that } (i, j) \in E\}$.

Interestingly, the fact that the players are related implies that each individual is not only guided by her own preferences, but she can also care about how deviating can negatively impact her relatives. In order to take into account the social context and the fact that a player can have consideration for other players (her neighbors in the graph), an appropriate notion of deviation, called *considerate improving move*, was introduced in [12].

Definition 3 (Considerate improving move (CIM)) For coalition $C \in \mathcal{C}$, a considerate improving move σ'_C from a state σ is a weak improving move where in addition, $\mathcal{F}(\sigma'_C, \sigma_{-C}) \geq_{\mathcal{N}(C)} \mathcal{F}(\sigma)$.

In a considerate improving move by coalition C , at least one player in C is better off and no player of $C \cup \mathcal{N}(C)$ can be worse off.

Consequently, the pair $(\mathcal{C}, \mathbf{CIM})$, where \mathcal{C} contains all the cliques of the social network $G = (N, E)$, leads to a new type of equilibrium called *considerate equilibrium* [12]. In what follows, we will say that a game always admits a considerate equilibrium if for any instance of the game, and any social network $G = (N, E)$, there exists at least one state σ for which no coalition of players forming a clique in G has a considerate improving move.

Example 2 Consider an instance where $N = \{1, 2, 3\}$ and $M = \{a, b\}$. The social network is a path, so the possible coalitions are $\mathcal{C} = \{\{1, 2\}, \{2, 3\}, \{1\}, \{2\}, \{3\}\}$. The profile of preferences is:

- 1 $b \sim a$
- 2 $b \succ a$
- 3 $a \succ b$

Consider a state σ where a is elected. Coalitions $\{1, 2\}$ and $\{2\}$ are the only coalitions having incentive to move, they want to make b win since $b \geq_{\{1,2\}} a$ and $b \geq_{\{2\}} a$. Because $\{3\} \in \mathcal{N}(\{1, 2\}) \cap \mathcal{N}(\{2\})$ and $a \succ_3 b$, coalitions $\{1, 2\}$ and $\{2\}$ cannot perform a **CIM** (without even taking into account the ability of the coalitions to change the outcome). Thus, σ is a considerate equilibrium.

Following the same idea, if the social network $G = (N, E)$ is composed of a set of disjoint cliques and only maximal cliques of G are considered in \mathcal{C} , then an equilibrium associated with $(\mathcal{C}, \mathbf{WIM})$ is called a *partition equilibrium* [9] (in this case, a **CIM** move corresponds to a **WIM** move). Clearly, if a considerate equilibrium exists, then a partition equilibrium exists. Furthermore, if $E = \emptyset$, then a considerate equilibrium corresponds to a Nash equilibrium. Thus, a Nash equilibrium is a special case of a considerate equilibrium and if a considerate equilibrium exists then a Nash equilibrium exists.

2.2 Dynamics

Beyond the existence of an equilibrium, we also take into account the *dynamics* of the voting game. Starting from an initial state, the players (or coalitions of players) can manipulate by successively changing their strategy. Each such move is supposed to be an improvement for the deviator(s) and we will exclusively study the moves defined above: **IM**, **WIM** and **CIM**. We will resort to indices to stress the step at which a state occurs. Namely σ^0 is the initial state whereas σ^t denotes the state at step t . A dynamics is a sequence $\sigma^0 \sigma^1 \dots \sigma^r$ such that each pair of consecutive steps $i, i+1$ is associated with an improving move for some coalition² that turns σ^i into σ^{i+1} . If the possible deviating coalitions belong to \mathcal{C} and moves are of type μ then the dynamics is said to be associated with the (\mathcal{C}, μ) -equilibrium. A dynamics ends when no further move is possible and therefore a (\mathcal{C}, μ) -equilibrium is reached. We have *convergence* when the dynamics is finite for every initial state. A natural restriction (see e.g. [16]) is to suppose that the initial state is truthful. Indeed, the players start by giving their true opinion and if they are not satisfied with the outcome then they reconsider their vote. A dynamics fails to converge when a state appears more than once.

In a non-equilibrium state σ , a coalition C (which can be a singleton) may have more than one possible improving moves. For improving moves of type μ , the *better replies* are the set of all joint strategies σ'_C such that $\mathcal{F}(\sigma_{-C}, \sigma'_C) \geq_C \mathcal{F}(\sigma)$, provided that a move of type μ turns σ into (σ_{-C}, σ'_C) . A better reply σ'_C is a *best reply* if no other joint strategy σ''_C satisfies $\mathcal{F}(\sigma_{-C}, \sigma''_C) \geq_C \mathcal{F}(\sigma_{-C}, \sigma'_C)$, i.e. it leads to an outcome that is undominated for C . We say that an improving move is *unanimous* regarding a coalition C if all members of C adopt the same strategy.

2.3 Voting rules

Numerous voting systems use *scores* and the rule is to elect the candidates who reach the highest score. Let S_C be a function which associates score $S_C(\sigma, x) \in \mathbb{R}$ with every pair $(\sigma, x) \in S^m \times M$. Since we focus on voting functions that output a single alternative, we use in this work a deterministic tie-breaking rule which follows an absolute priority ranking over the alternatives. Namely, a fixed ranking \triangleright of the candidates is employed and amongst the candidates that attain the best score, the one coming first in \triangleright wins the election.

Let us assume for the moment that $S = L(M)$, i.e. every player is asked to report a strict ordering of M . In *positioning scoring rules*

² No two coalitions can move simultaneously.

(PSR), the score $Sc(\sigma, x)$ of alternative x under profile σ depends on the absolute position of x in each ballot. Concretely, we are given a vector $\alpha = (\alpha_1, \dots, \alpha_m)$ such that $\alpha_1 \geq \dots \geq \alpha_m$ and $\alpha_1 > \alpha_m$. If x is placed at position k in a ballot, then x receives α_k points. The score of x is defined as the sum of these points over all ballots.

Thus, each PSR is characterized by its vector α . We can mention in particular the *Borda* rule in which $\alpha = (m-1, m-2, \dots, 0)$ and the *k-approval* rule for $k \in [m-1] = \{1, \dots, m-1\}$ with $\alpha = (1, \dots, 1, 0, \dots, 0)$ in which k consecutive ones are followed by $m-k$ consecutive zeros. If k is equal to 1, then the associated voting rule is called *Plurality*. If k is equal to $m-1$, then the associated voting rule is called *Anti-plurality* (also known as *Veto*).

Note that for *Plurality* and *Anti-plurality*, only a single alternative in each ballot is useful. This alternative is respectively the first one (the only candidate approved by the voter) and the last one (the only candidate vetoed by the voter). Thus, in these two cases, we assume that the strategy space S is equal to M instead of $L(M)$, and therefore a strategy profile σ belongs to M^n .

In *runoff voting rules*, the election runs in several rounds. We can mention in particular the well-known *Plurality with runoff* which is actually used for political elections in many countries. In such a rule, given a strategy profile in $L(M)^n$ as an input, the first round selects the two best ranked alternatives regarding *Plurality* (use \triangleright to break ties). Then, all eliminated candidates are removed from the profile, providing a profile where only the two selected alternatives are present. The second round elects the winner of this resulting profile under *Plurality*. Note that only the first round is necessary if an alternative gets a majority of the votes. The *Single Transferable Vote (STV)* rule is an iterated process taking as an input a strategy profile in $L(M)^n$. At each step, the loser of *Plurality* gets eliminated (use \triangleright to break ties), and the profile is updated by removing this alternative from the ballots of the agents. The process continues until an alternative obtains an absolute majority of votes under *Plurality* and thus gets elected.

Given a strategy profile σ with the associated linear order $\succ^\sigma \in L(M)^n$ and two alternatives $a, b \in M$, let $W(a, b)$ and $\omega(a, b)$ be the set of voters who prefer a to b and the number of voters who prefer a to b , respectively. That is, $W(a, b) = \{i \in N \mid a \succ_i^\sigma b\}$ and $\omega(a, b) = |W(a, b)|$.

Some voting rules, instead of taking into account the position of an alternative in a ballot like PSRs, are based on pair-wise comparisons. The *Copeland* rule assigns to each alternative x the number of alternatives that x beats in a pair-wise election in a given state $\sigma \in L(M)^n$, that is $Sc(\sigma, x) = |\{y \in M \setminus x \mid \omega(x, y) > \frac{n}{2}\}|$. The *Maximin* rule assigns to x the minimum number of voters in favour of x in any pair-wise comparison in σ , i.e. $Sc(\sigma, x) = \min_{y \in M \setminus x} \omega(x, y)$.

3 Positioning Scoring Rules (PSRs)

3.1 Plurality

A profile σ is said to be *unanimous* if there exists a strategy $s \in S$ such that $\sigma_i = s$ for all $i \in N$. Since $S = M$ in the voting game with *Plurality*, every alternative x induces a unanimous profile (x, \dots, x) .

The voting game under *Plurality* always admits a Nash equilibrium, e.g. the unanimous profile (x, \dots, x) where x is the best alternative in the tie breaking rule \triangleright . However, it is known that a strong equilibrium is not guaranteed to exist (see e.g. [26]), as we can see in the following example showing a well-known Condorcet paradox.

Example 3 $N = \{1, 2, 3\}$, $M = \{a, b, c\}$, $a \triangleright b \triangleright c$ and the profile of preferences is:

- 1 $a \succ b \succ c$
- 2 $c \succ a \succ b$
- 3 $b \succ c \succ a$

If a is elected then players 2 and 3 have incentive to deviate to c . If c is elected then players 1 and 3 have incentive to deviate to b . If b is elected then players 1 and 2 have incentive to deviate to a .

This example also rules out the existence of a super strong equilibrium but we shall see that a considerate equilibrium must exist.

Theorem 1 Every instance of the voting game with players' preferences in $\mathcal{L}(M)$ and $\mathcal{F} = \text{Plurality}$ admits a considerate equilibrium.

Proof: The social network is a graph $G = (N, E)$. Let $a \in M$ be the best alternative w.r.t. \triangleright . Under *Plurality*, the only coalitions which are able to change the outcome by deviating from a unanimous state are cliques of size at least $n/2$. For the unanimous state (a, \dots, a) , these *powerful coalitions* are only those of size strictly larger than $n/2$, that we denote \mathcal{C}_p . Let $\mathcal{C}_{n/2}$ be the set of cliques of size exactly $n/2$. Let us denote by \mathcal{Q} the set of all agents belonging to a coalition which can change the outcome of a unanimous state, that is $\mathcal{Q} = \bigcup_{Q \in \mathcal{C}_p \cup \mathcal{C}_{n/2}} Q$. If $\mathcal{C}_p = \emptyset$ then the unanimous profile (a, \dots, a) is a considerate equilibrium. Suppose from now on that $\mathcal{C}_p \neq \emptyset$.

For any coalitions $C \in \mathcal{C}_p$ and $C' \in (\mathcal{C}_p \cup \mathcal{C}_{n/2})$, C and C' have at least one member in common, therefore $C \subseteq (\mathcal{N}(C') \cup C')$ and $\mathcal{Q} \subseteq (C \cup \mathcal{N}(C'))$, for all $C \in \mathcal{C}_p$. Hence, C' cannot deviate so that a worse candidate, from the viewpoint of at least one member of C , is elected. If an alternative x strictly undominated over $C \in \mathcal{C}_p$ exists, then the unanimous profile (x, \dots, x) is a considerate equilibrium.

Suppose from now on that for every coalition $C \in \mathcal{C}_p$, there does not exist any strictly undominated alternative over C . However, an undominated alternative over C must exist. For a coalition C and an alternative x , let I_x^C be the *indifference set* of x within C , i.e. $I_x^C = \{y \in M \setminus \{x\} : y \sim_C x\}$. Let ND be the subset of alternatives which are both undominated over at least one coalition of \mathcal{C}_p and undominated over \mathcal{Q} . The set ND is never empty because $\mathcal{C}_p \neq \emptyset$. We denote by x the best alternative of ND w.r.t. \triangleright and $C \in \mathcal{C}_p$ the coalition for which x is undominated. We will analyze deviations from the unanimous profile (x, \dots, x) and every time a deviation is performed, then we will consider for the next step the unanimous profile of the corresponding winner. Since we start from unanimous profiles, the deviations are only performed by coalitions in $(\mathcal{C}_p \cup \mathcal{C}_{n/2})$. The winner of every deviation belongs to $I_x^C \cup \{x\}$ because the deviating coalitions have consideration for C . A coalition $C' \in \mathcal{C}_p$ cannot deviate from (x, \dots, x) , because $\mathcal{Q} \subseteq (C' \cup \mathcal{N}(C'))$, x is undominated over \mathcal{Q} by definition of ND and $C' \subseteq \mathcal{Q}$. However, a coalition $C' \in \mathcal{C}_{n/2}$ can deviate if there exists $y \in I_x^C$ such that $y \geq_{C'} x$ and $y \triangleright x$. If $y \in \text{ND}$ then $y \triangleright x$ contradicts the fact that x is the best alternative of ND w.r.t. \triangleright , so $y \notin \text{ND}$. Since $y \geq_{C'} x$ and x is undominated over \mathcal{Q} , there exists $j \in \mathcal{Q} \setminus (C \cup C')$ such that $x \succ_j y$. If there exists $C'' \in (\mathcal{C}_p \cup \mathcal{C}_{n/2})$ for which $j \in C''$ with $C'' \cap C' \neq \emptyset$, then C' cannot deviate to y by consideration for j because $j \in \mathcal{N}(C')$, therefore $\overline{C'} = N \setminus C'$ is a coalition belonging to $\mathcal{C}_{n/2}$ and $j \in \overline{C'}$. It follows that $\mathcal{Q} = N$. Obviously, $C' \cap C \neq \emptyset$ and $\overline{C'} \cap C \neq \emptyset$. These voters cannot be the beneficiaries of a deviation since they are indifferent among alternatives of I_x^C . Let $I = C' \setminus C$ and $J = \overline{C'} \setminus C$ denote respectively the voters of C' and $\overline{C'}$ who do not belong to C . Then, C' deviates to y and we consider for the next step the profile (y, \dots, y) . Observe that for a given succession of such deviations performed by coalitions in $\mathcal{C}_{n/2}$,

the winners are not in ND and the rank of the winner in \triangleright strictly improves. Thus, there is only a finite number of such deviations. Now, every time a coalition $C'' \in \mathcal{C}_p$ can deviate to $z \in I_x^C$ from a unanimous profile (y^t, \dots, y^t) where y^t is the winner at step t (we can verify that C'' cannot deviate to x), it follows that $z \geq_N y^t$ because $\mathcal{Q} \subseteq (C'' \cup \mathcal{N}(C''))$. There exists a voter i such that $z \succ_i y^t$, thus $i \in I \cup J$, say $i \in J$, implying that $j \in \mathcal{N}(C')$. So, no coalition can make j less satisfied. Hence, the improvements within the sequence of the defined deviations follow the preferences of every such j and during a succession of deviations performed by coalitions of $\mathcal{C}_{n/2}$ the rank of the winner in \triangleright is improved. Since we have a finite number of alternatives, we finally reach an alternative for which the unanimous associated profile is a considerate equilibrium. ■

Note that when $n > 2$, the considerate equilibria that we constructed in the previous proof are also Nash equilibria because they are unanimous profiles, showing that we can combine two requirements which may be conflicting. Indeed, there exist instances for $n = 2$ where the set of considerate equilibria and the set of Nash equilibria do not intersect.

Theorem 1 gives the existence of a considerate equilibrium and therefore, existence of a partition equilibrium, in any instance of the voting game under Plurality. However, if we let the players deviate, do we reach this equilibrium? Unfortunately, the answer may be negative even for the partition equilibrium and additional natural restrictions. We impose for instance that every deviation is a unanimous direct best reply³, the initial profile is truthful and the players' preferences are strict.

Proposition 1 *The dynamics associated with the partition equilibrium may not converge in the voting game with \mathcal{F} =Plurality, even if the initial profile is truthful, each move is a unanimous direct best reply and the preferences are strict, single-peaked and single-crossing.*

Proof: Consider an instance where $N = \{1, 2, \dots, 12\}$, $M = \{a, b, c, d\}$ and $c \triangleright d \triangleright b \triangleright a$. The profile of preferences is:

- 1, 2, 3 $a \succ b \succ d \succ c$
- 4 $b \succ a \succ d \succ c$
- 5, 6 $b \succ a \succ d \succ c$
- 7, 8, 9 $c \succ d \succ b \succ a$
- 10, 11 $d \succ b \succ a \succ c$
- 12 $d \succ b \succ a \succ c$

The preferences are single-peaked using the axis over the candidates (c, d, b, a) and single-crossing using the axis over the voters $(1, 2, 3, 4, 5, 6, 10, 11, 12, 7, 8, 9)$. The partition over the voters is $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7, 8, 9\}, \{10, 11\}, \{12\}\}$. The next table gives a sequence of states where the first and last ones coincide. Deviations are marked with bold letters.

Steps	Players												Winner
	1	2	3	4	5	6	7	8	9	10	11	12	
0	a	a	a	b	b	b	c	c	c	d	d	d	c
1	a	a	a	a	b	b	c	c	c	d	d	d	a
2	a	a	a	a	b	b	c	c	c	b	b	d	b
3	a	a	a	a	b	b	d	d	d	b	b	d	d
4	a	a	a	b	b	b	d	d	d	b	b	d	b
5	a	a	a	b	b	b	d	d	d	d	d	d	d
6	a	a	a	b	b	b	c	c	c	d	d	d	c

Interestingly, Proposition 1 can be mitigated if we consider a special partition of N where all coalitions have the same size. The result follows from a simple extension of a proof given in [16].

Proposition 2 *If \mathcal{P} is a partition of N such that all coalitions of \mathcal{P} have the same size, then the dynamics associated with a $(\mathcal{P}, \mathbf{WIM})$ -equilibrium converges for any profile of preferences, any initial profile, and if unanimous direct replies are performed.*

A *Condorcet winner* is an alternative x which wins with an absolute majority against any other alternative, i.e. $\omega(x, y) > \frac{n}{2}$ for all $y \in M \setminus \{x\}$. Plurality is not *Condorcet consistent* which means that it does not always elect the Condorcet winner when it exists. A unanimous profile with the Condorcet winner is always a considerate equilibrium (as stated in [26] for strong equilibria) since no absolute majority of voters prefer another alternative. However, when a Condorcet winner exists, the game may not converge to a state electing it, even with a truthful initial profile. This observation is due to the non-Condorcet consistency of the Plurality voting rule.

Example 4 *Let $N = \{1, 2, 3\}$, $M = \{a, b, c, d\}$, $a \triangleright b \triangleright c \triangleright d$, the coalitions are $\{\{1\}, \{2\}, \{3\}\}$ and the profile of preferences is:*

- 1 $a \succ b \succ c \succ d$
- 2 $c \succ b \succ d \succ a$
- 3 $d \succ b \succ a \succ c$

From the truthful initial profile $\{a, c, d\}$ electing a , the only possible deviation is that agent 2 deviates to alternative d , making d winner. We reach the state $\{a, d, d\}$ electing d , which is a considerate equilibrium whereas b is Condorcet winner.

3.2 Antiplurality

Before going into detailed analysis of antiplurality, we provide the definition of a *feasible elimination procedure* (f.e.p.), a tool introduced in social choice theory (see e.g. [22]).

Definition 4 *For a mapping $\beta : M \rightarrow \mathbb{N}$ such that $\sum_{x \in M} \beta(x) = n + 1$, an f.e.p. is a sequence $(x_1, C_1; x_2, C_2; \dots, C_{m-1}; x_m)$ where $\forall i \in [m - 1], C_i \subseteq N$ is such that: (i) $|C_i| = \beta(x_i)$ and $\forall j \in [m - 1] \setminus \{i\}, C_i \cap C_j = \emptyset$, (ii) $M = \bigcup_{k \in [m]} x_k$, and (iii) $\forall l \in C_i$ and $\forall k \in \{i + 1, \dots, m\}, x_k \succ_l x_i$.*

It has been shown that an f.e.p. exists for any preference profile in $L(M)^n$ and for any mapping β such that $\sum_{x \in M} \beta(x) = n + 1$ (see e.g. Remark 9.2.1 in [22]). We will use this fact⁴ to show the following proposition:

Proposition 3 *A strong equilibrium exists in any instance of the voting game with players' preferences in $L(M)$ and \mathcal{F} =Antiplurality.*

Proof: In order to prove the proposition, we show that it is possible to construct, from a given f.e.p., a state σ which is a strong equilibrium for antiplurality. First of all, let us define the mapping β used to define the f.e.p. The value $\beta(x)$ corresponds to the minimum amount of vetos required to ensure that x cannot be chosen by Antiplurality (whatever the other ballots). Let q and r be the quotient and the rest of the euclidean division of n by m , respectively. It is easy to check that $\beta(x) = q + 1$ if x is ranked among the $r + 1$ first alternatives

³ Convergence to a Nash equilibrium is not guaranteed with better or best replies but it converges with direct replies [16]. For Plurality, replies are direct when the voters deviate by voting for the new winner.

⁴ Note that the condition $n + 1 \geq m$ appears in existence result of [22] but it is easy to fulfill this condition by only keeping the $n + 1$ first candidates in the tie-breaking rule. Indeed, only these alternatives can be elected.

in the tie-breaking rule \triangleright , and $\beta(x) = q$ otherwise. Note that by the definition of q and r , $\sum_{x \in M} \beta(x) = n + 1$ holds.

Let $(x_1, C_1, x_2, C_2, \dots, C_{m-1}, x_m)$ be an f.e.p. for the mapping β described above, and let σ be a state such that $\forall i \in [m-1]$ and $\forall j \in C_i, \sigma_j = x_i$. Note that by the definition of β , $\mathcal{F}(\sigma) = x_m$. We conclude the proof by showing that σ is a strong equilibrium. By contradiction, if σ is not a strong equilibrium then there exists a coalition $C \subseteq N$ and a joint strategy σ'_C such that $\mathcal{F}(\sigma'_C, \sigma_{-C}) \succ_C x_m$. Let y denote the candidate $\mathcal{F}(\sigma'_C, \sigma_{-C})$. By the definition of β , there must be $i \in [m-1]$ and $l \in C_i$ such that $y = x_i$ and $l \in C$, because otherwise y is vetoed by at least $\beta(y)$ voters and cannot be chosen by \mathcal{F} . But by (iii) of Definition 4, this implies that $x_m \succ_l x_i = y$, a contradiction with $y \succ_C x_m$. ■

Note that the strong equilibrium exhibited in the above proof can be constructed in polynomial time.

Observe that when a Condorcet winner exists in the preference profile, a strong equilibrium electing it may not exist, as we can see in the next example. Actually, the notion of winner under Antiplurality is far from being related to the concept of Condorcet winner, and a minority of voters can have a significant power within this rule.

Example 5 Consider an instance where $N = \{1, 2, 3\}$ and $M = \{a, b, c, d\}$. The profile of preferences is:

- 1 $b \succ a \succ c \succ d$
- 2 $c \succ d \succ a \succ b$
- 3 $b \succ d \succ c \succ a$

Alternative b is the Condorcet winner. However, from a state where b is elected, voter 2 has always incentive to deviate by vetoing b , and this veto is sufficient to avoid the election of b . Hence, there is no strong equilibrium electing b .

Even if we relax the assumption of strict preferences, Proposition 3 continues to hold. However, this proposition does not ensure the existence of a super strong equilibrium when the preferences are not strict, as illustrated by the following counterexample.

Example 6 Let $N = \{1, 2, 3\}$, $M = \{a, b\}$ and the profile of preferences is:

- 1 $a \succ b$
- 2 $b \succ a$
- 3 $a \sim b$

If a is elected then coalition $\{2, 3\}$ can deviate and veto a . If b is elected then coalition $\{1, 3\}$ can deviate and veto b .

Nevertheless, the following theorem shows that a considerate equilibrium is guaranteed to exist even with non-strict preferences.

Theorem 2 A considerate equilibrium exists in any instance of the voting game with players' preferences in $\mathcal{L}(M)$ and $\mathcal{F} = \text{Antiplurality}$.

Proof: The proof relies on a refinement of the f.e.p. to the concept of considerate equilibrium. The *considerate f.e.p.* (c.f.e.p.) is defined in a similar fashion as f.e.p., except that condition (iii) is replaced by (iii') $\forall l \in C_i$ and $\forall k \in \{i+1, \dots, m\}, x_k \succ_l x_i$, or $[x_k \sim_l x_i$ and $x_i \not\prec_{\delta(l)} x_k]$, where $\delta(l)$ is the set of neighbors of l in G . First of all, let us show that a c.f.e.p. exists for any preference profile $\succ \in \mathcal{L}(M)^n$. To this end, for any voter $i \in N$, we construct a strict preference \succ'_i which is consistent with the strict part of \succ_i , and

where ties are broken by $\geq_{\delta(i)}$ (or arbitrarily in case of incomparability for $\geq_{\delta(i)}$). This construction leads to a profile of strict preferences \succ' . By Remark 9.2.1 of [22], we know that an f.e.p. exists for \succ' . We state that such an f.e.p. is also a c.f.e.p. for \succ . Indeed, conditions (i) and (ii) trivially hold because they are similar for both f.e.p. and c.f.e.p., and they do not depend on the considered profile of preferences. Furthermore, the construction of \succ' ensures that for any $l \in N$, $x \succ'_l y$ implies $x \succ_l y$, or $[x \sim_l y$ and $y \not\prec_{\delta(l)} x]$. Therefore, (iii') holds and the f.e.p. for \succ' is also a c.f.e.p. for \succ .

The remainder of this proof follows the same line as the proof of Proposition 3. Let $(x_1, C_1; x_2, C_2; \dots, C_{m-1}; x_m)$ be a c.f.e.p. for the mapping β described in the proof of Proposition 3, and let σ be a state such that $\sigma_j = x_i, \forall i \in [m-1]$ and $\forall j \in C_i$. Let us show that σ is a considerate equilibrium. By contradiction, if σ is not a considerate equilibrium then there exists a coalition $C \in \mathcal{C}$ and a joint strategy σ'_C such that $\mathcal{F}(\sigma'_C, \sigma_{-C}) \geq_C x_m$ and $\mathcal{F}(\sigma'_C, \sigma_{-C}) \geq_{N(C)} x_m$. Let y denote the candidate $\mathcal{F}(\sigma'_C, \sigma_{-C})$. There must be $i \in [m-1]$ and $l \in C_i$ such that $y = x_i$ and $l \in C$. Furthermore, any neighbor of l in G belongs to C or $N(C)$, and no other voter belongs to C . Therefore, $C \cup N(C) = \delta(i) \cup \{i\}$. But by condition (iii') of the definition of a c.f.e.p., this implies that $x_m \succ_l x_i = y$ or $x_m \sim_i x_i$ and $y = x_i \not\prec_{C \cup N(C)} x_m$, a contradiction with $y \geq_C x$ and $y \geq_{N(C)} x_m$. ■

Unfortunately, the dynamics associated with the considerate equilibrium is not guaranteed to converge, even if we restrict ourselves to the dynamics associated with the partition equilibrium and we consider unanimous direct best replies⁵.

Proposition 4 The dynamics associated with the partition equilibrium may not converge in the voting game with $\mathcal{F} = \text{Antiplurality}$, even if the initial profile is truthful, each move is a unanimous direct best reply and the preferences are strict, single-peaked and single-crossing.

Proof: Consider an instance where $N = \{1, 2, \dots, 6\}$, $M = \{a, b, c, d\}$ and $a \triangleright b \triangleright c \triangleright d$. The preferences are single-peaked using the axis over the candidates (c, d, a, b) and single-crossing using the axis over the voters $(2, 1, 6, 3, 4, 5)$. The profile of preferences is:

- 1 $d \succ c \succ a \succ b$
- 2 $c \succ d \succ a \succ b$
- 3 $a \succ d \succ b \succ c$
- 4, 5 $b \succ a \succ d \succ c$
- 6 $a \succ d \succ c \succ b$

The partition over the voters is $\{\{1\}, \{2\}, \{3\}, \{4, 5\}, \{6\}\}$. The next table gives a sequence of states where the steps 1 to 7 form a cycle. Deviations are marked with bold letters. Step 0 corresponds to the truthful profile.

Steps	Players						Winner
	1	2	3	4	5	6	
0	$-b$	$-b$	$-c$	$-c$	$-c$	$-b$	a
1	$-b$	$-a$	$-c$	$-c$	$-c$	$-b$	d
2	$-b$	$-a$	$-c$	$-c$	$-c$	$-d$	a
3	$-b$	$-a$	$-c$	$-a$	$-a$	$-d$	b
4	$-b$	$-a$	$-c$	$-a$	$-a$	$-b$	d
5	$-b$	$-d$	$-c$	$-a$	$-a$	$-b$	c
6	$-b$	$-d$	$-c$	$-c$	$-c$	$-b$	a
7	$-b$	$-a$	$-c$	$-c$	$-c$	$-b$	d

⁵ Replies are direct under Antiplurality when the voters veto the current winner. Convergence to a Nash equilibrium is guaranteed when direct replies are performed [14, 24].

3.3 Other PSRs

In this section, the strategy space S is $L(M)$. Each PSR is characterized by its score vector α . After normalization, the score vector of every PSR can be written as $\alpha = (1, \alpha_2, \dots, \alpha_{m-1}, 0)$ where $\alpha_i \in [0, 1]$ and the α_i s remain non increasing. The PSR is neither Plurality nor Anti-plurality iff $\alpha_2 > 0$ and $\alpha_{m-1} < 1$. If $\alpha_i \in \{0, 1\}$ for all $i \in \{2, \dots, m-1\}$, then the associated PSR is k -approval.

We know that for Borda and k -approval (k is a constant), a voting game may not converge to a Nash equilibrium [14], even if the initial state is truthful and only best replies are used. Therefore, this result holds too for a considerate equilibrium. Furthermore, we can prove that a Nash equilibrium (and thus, a considerate equilibrium) is not guaranteed to exist for Borda and k -approval where k is fixed according to either the number of consecutive ones, or to the number of consecutive zeros.

Proposition 5 *Let l be an integer such that $l > 1$ and consider the k -approval rule where $k = l$ or $k = m - l$. A Nash equilibrium is not guaranteed to exist in the voting game, even if the players' preferences are strict and single-peaked.*

Proof: Consider an instance where $N = \{1, 2\}$, $M = \{x_1, x_2, \dots, x_{2k}\}$, and $x_1 \triangleright x_2 \triangleright \dots \triangleright x_{2k}$. Thus, $m = 2k$. The profile of preferences is:

$$\begin{array}{l} 1 \quad x_1 \succ x_2 \succ \dots \succ x_m \\ 2 \quad x_m \succ x_{m-1} \succ \dots \succ x_1 \end{array}$$

With such a k and m , either the winner is present in the k first ranked candidates of both voters, or the winner is x_1 . Moreover, there is always one agent who prefers at least k alternatives to the current winner. Suppose alternative x_i is elected. If $i \leq k$, then a better reply for agent 2 is to rank candidates x_1, \dots, x_i last in her new ballot and put at the first position her preferred candidate within the top k of agent 1. If $i > k$, a better reply for agent 1 is to place candidates x_i, \dots, x_m among the k last ranked of her new ballot. ■

Proposition 6 *A Nash equilibrium is not guaranteed to exist in the voting game where \mathcal{F} is a PSR with $0 < \alpha_2 \leq \frac{1}{2}$, even if the players' preferences are strict and single-peaked.*

Proof: Consider a 2-player instance where $M = \{a, b, c\}$, $a \triangleright b \triangleright c$ and the profile of preferences is:

$$\begin{array}{l} 1 \quad a \succ b \succ c \\ 2 \quad c \succ b \succ a \end{array}$$

If a is elected then agent 2 puts a last in her new ballot and puts first the best ranked alternative within agent 1's ballot between b and c . If c is elected then agent 1 puts c last in her new ballot and puts first the best ranked alternative within agent 2's ballot between a and b . Finally, if b is elected then agent 1 can make a the winner: if c is ranked first by agent 2 then agent 1 plays $a \succ b \succ c$, otherwise she plays $a \succ c \succ b$. ■

Note that Borda is covered by this proposition so a Nash equilibrium (and thus, a considerate equilibrium) is not guaranteed to exist.

4 Runoff voting rules

The voting game under Plurality with runoff and STV always admits a Nash equilibrium, e.g. the strategy profile where the best alternative coming first in the tie breaking rule \triangleright is placed on the top of each

ballot. More generally, we prove that a considerate equilibrium is guaranteed to exist in any instance of the game under Plurality with run-off and STV. We cannot generalize again to the existence of a strong equilibrium, as we can see for the profile given in Example 3.

Theorem 3 *Every instance of the voting game with players' preferences in $\mathcal{L}(M)$ and $\mathcal{F} \in \{STV, \text{Plurality with runoff}\}$ admits a considerate equilibrium.*

Proof: The proof follows the same idea as the proof for Plurality (Theorem 1). It suffices to show that under STV and Plurality with runoff the set of powerful coalitions \mathcal{C}_p also corresponds to the set of coalitions with size strictly larger than $n/2$. Indeed, if a is the best alternative w.r.t. \triangleright , then in the unanimous profile where a is placed on the top of each ballot, a is elected without any further round because it gets the absolute majority of votes. Thus, only a coalition of size strictly larger than $n/2$ can make the outcome change by placing another alternative unanimously on the top of the new ballot. ■

We have existence of a considerate equilibrium. Unfortunately, the game may not converge to it since it may not even converge to a Nash equilibrium, as the next proposition states. Moreover, this is still the case for restricted best replies.

Proposition 7 *The dynamics associated with the Nash equilibrium is not guaranteed to converge in the voting game with $\mathcal{F} \in \{STV, \text{Plurality with runoff}\}$, even if the initial profile is truthful, the players' preferences are strict, single-peaked and single-crossing, and each move is a best reply minimizing the distance to the truthful ballot in terms of number of differences in pair-wise comparisons.*

Proof: Consider an instance where $N = \{1, 2, 3, 4\}$, $M = \{a, b, c, d\}$ and $a \triangleright b \triangleright c \triangleright d$. The preferences are single-peaked using the axis over the candidates (a, c, d, b) and single-crossing using the axis over the voters $(2, 3, 4, 1)$. The next table gives a sequence of states where the first and last ones coincide. Deviations are marked with bold letters. Step 0 represents the truthful profile. Each linear order is the ballot of a voter and the last line of the table specifies the winner at each step.

Step 0	Step 1	Step 2	Step 3	Step 4
$c \triangleright a \triangleright d \triangleright b$	$a \triangleright c \triangleright d \triangleright b$	$a \triangleright c \triangleright d \triangleright b$	$c \triangleright a \triangleright d \triangleright b$	$c \triangleright a \triangleright d \triangleright b$
$b \triangleright d \triangleright c \triangleright a$	$b \triangleright d \triangleright c \triangleright a$	$d \triangleright b \triangleright c \triangleright a$	$d \triangleright b \triangleright c \triangleright a$	$b \triangleright d \triangleright c \triangleright a$
$d \triangleright b \triangleright c \triangleright a$	$d \triangleright b \triangleright c \triangleright a$	$d \triangleright b \triangleright c \triangleright a$	$d \triangleright b \triangleright c \triangleright a$	$d \triangleright b \triangleright c \triangleright a$
$c \triangleright d \triangleright a \triangleright b$	$c \triangleright d \triangleright a \triangleright b$	$c \triangleright d \triangleright a \triangleright b$	$c \triangleright d \triangleright a \triangleright b$	$c \triangleright d \triangleright a \triangleright b$
b	a	d	c	b

This counterexample works for STV and Plurality with runoff. ■

5 Voting rules based on pair-wise comparisons

5.1 Copeland

We show by a simple counterexample that even a Nash equilibrium is not guaranteed to exist in the voting game when using the Copeland rule. Therefore, a considerate equilibrium may not exist.

Proposition 8 *A Nash equilibrium is not guaranteed to exist in the strategic voting game where $\mathcal{F} = \text{Copeland}$, even if the preferences are strict and single-peaked.*

Proof: Consider an instance where $N = \{1, 2\}$, $M = \{a, b, c\}$ and $a \triangleright b \triangleright c$. The profile of preferences is:

$$\begin{array}{l} 1 \quad a \succ b \succ c \\ 2 \quad c \succ b \succ a \end{array}$$

An alternative x wins against y in a pair-wise election if x is ranked before y in the two ballots. If a wins, then agent 2 deviates by placing a at the last position of her new ballot and placing on top of her new ballot the first ranked candidate in the ballot of agent 1 between b and c . If c wins, then it suffices for agent 1 to place c at the last position of her new ballot because c is ranked last in the tie-breaking. Finally, if b wins, then if c is not ranked first in agent 2's ballot, then agent 1 plays $a \succ c \succ b$; otherwise, agent 1 plays $a \succ b \succ c$. In conclusion, there is no Nash equilibrium in this instance. ■

5.2 Maximin

The voting game under Maximin always admits a Nash equilibrium, e.g. the profile where the best alternative w.r.t. \triangleright is placed on top of each ballot. However, a strong equilibrium is not guaranteed to exist as we can see with the profile given in Example 3. Nevertheless, a considerate equilibrium is guaranteed to exist in any instance.

Theorem 4 Every instance of the voting game where the players' preferences are in $\mathcal{L}(M)$ and $\mathcal{F}=\text{Maximin}$ admits a considerate equilibrium.

Proof: The proof follows the same idea as the proof for Plurality (Theorem 1). It suffices to show that under Maximin the set of powerful coalitions \mathcal{C}_p also corresponds to the set of coalitions with size strictly larger than $n/2$. Indeed, if a is the best alternative w.r.t \triangleright then a gets elected in the unanimous profile where a is placed on top of each ballot. From such a profile, only a coalition of size strictly larger than $n/2$ can change the outcome by placing another alternative y unanimously on top of the new ballot. If the coalition has only $n/2$ agents, it is not sufficient because a still has a maximin score of $n/2$ and no other candidate can have a strictly better score. ■

It is known that the strategic voting game under the Maximin rule is not guaranteed to converge to a Nash equilibrium with an arbitrary deterministic tie-breaking [14]. However, even with a deterministic tie-breaking which is a linear order, the game under Maximin is not guaranteed to converge, as we can see in the next proposition. Moreover, we do not use a general best reply but a restricted one.

Proposition 9 The dynamics associated with the Nash equilibrium is not guaranteed to converge in the voting game with $\mathcal{F}=\text{Maximin}$, even if the initial profile is truthful, the players' preferences are strict, single-peaked and single-crossing, and each move is a best reply minimizing the distance to the truthful ballot in terms of number of differences in pair-wise comparisons.

Proof: Consider an instance where $N = \{1, 2, 3, 4, 5\}$, $M = \{a, b, c, d\}$ and $a \triangleright b \triangleright c \triangleright d$. The preferences are single-peaked using the axis over the candidates (d, b, c, a) and single-crossing using the axis over the voters $(1, 4, 3, 2, 5)$. The next table gives a sequence of states where the steps 1 to 5 form a cycle. Deviations are marked with bold letters. Step 0 corresponds to the truthful profile. Each linear order is the ballot of a voter and the last line of the table specifies the winner at each step.

Step 0	Step 1	Step 2	Step 3	Step 4	Step 5
$d \succ b \succ c \succ a$	$d \succ b \succ c \succ a$	$c \succ d \succ b \succ a$	$c \succ d \succ b \succ a$	$d \succ b \succ c \succ a$	$d \succ b \succ c \succ a$
$a \succ c \succ b \succ d$	$a \succ d \succ c \succ b$	$a \succ d \succ c \succ b$	$a \succ d \succ c \succ b$	$a \succ d \succ c \succ b$	$a \succ d \succ c \succ b$
$c \succ b \succ a \succ d$	$c \succ b \succ a \succ d$	$c \succ b \succ a \succ d$	$c \succ b \succ a \succ d$	$c \succ b \succ a \succ d$	$c \succ b \succ a \succ d$
$b \succ d \succ c \succ a$	$b \succ d \succ c \succ a$	$b \succ d \succ c \succ a$	$b \succ d \succ c \succ a$	$b \succ d \succ c \succ a$	$b \succ d \succ c \succ a$
$a \succ c \succ b \succ d$	$a \succ c \succ b \succ d$	$a \succ c \succ b \succ d$	$a \succ b \succ d \succ c$	$a \succ b \succ d \succ c$	$a \succ c \succ b \succ d$
c	a	c	a	b	a

6 Discussion and perspectives

We proposed to explore voting games from a strategical and social point of view. The considerate equilibrium captures a stable outcome regarding every set of coalitions arising from a social network.

The next table summarizes the existence and convergence results for the different voting rules that we explored. They are classified by solution concepts, which allows to spot the gap between existence/non-existence and convergence/non-convergence since the different equilibria are related: \exists considerate equilibrium $\Rightarrow \exists$ partition equilibrium $\Rightarrow \exists$ Nash equilibrium.

		Plurality	Veto	k -approval	Borda
strong	Existence	×	✓(Prop.3)	×	×
	Convergence	×	×	×	×
considerate	Existence	✓(Th.1)	✓(Th.2)	×	×
	Convergence	×	×	×	×
partition	Existence	✓	✓	×	×
	Convergence	×(Prop.1)	×(Prop.4)	×	×
Nash	Existence	✓	✓	×(Prop.5)	×(Prop.6)
	Convergence	✓[16]	✓[14, 24]	×[14, 24]	×[14, 24]
		STV	PwRO	Copeland	Maximin
strong	Existence	×	×	×	×
	Convergence	×	×	×	×
considerate	Existence	✓(Th.3)	✓(Th.3)	×	✓(Th.4)
	Convergence	×	×	×	×
partition	Existence	✓	✓	×	✓
	Convergence	×	×	×	×
Nash	Existence	✓	✓	×(Prop.8)	✓
	Convergence	×(Prop.7)	×(Prop.7)	×	×(Prop.9)

We can remark that convergence under general best replies (even if they are refined as in Prop. 7 and 9) is difficult to achieve. We restricted ourselves to almost general best replies. A possible extension would be to analyze convergence to a considerate equilibrium for some restricted manipulation moves, as studied in [23, 11, 21] for Nash equilibria. Another possible way to achieve convergence is to focus on specific classes of graphs or coalition families, provided that it matches with an actual social structure.

On the positive side, we were able to prove the existence of a considerate equilibrium for a significant number of voting rules, namely Plurality, Antiplurality, Plurality with runoff, STV and Maximin. These results are encouraging because the notion of considerate equilibrium covers a large spectrum of families of coalitions.

As a balance, the assumption of consideration within this equilibrium — the fact no coalition harms its neighbors — is rather strong. Actually, without the consideration assumption, it is not possible to generalize the existence of such an equilibrium to every class of graph. As an example, take the complete graph which corresponds to a super strong equilibrium and see Example 6. This counterexample holds for every voting rule for which we proved the existence of a considerate equilibrium, showing the importance of the consideration assumption.

Nevertheless, the consideration assumption is relevant if we assume that agents are not fully selfish and that they care about their relatives. Moreover, it allows to integrate a social dimension into the game. If agents are connected in the social network, then they are reluctant to act in a way that harm their partners. If one wants to escape from the consideration assumption, then an option is to restrict to specific classes of graphs, or to specific families of coalitions. For example, with the partition equilibrium, our existence results hold without the consideration assumption. One can also think of *laminar structures* where there is either inclusion or empty intersection between every pair of coalitions. Another possibility is to relax the consideration assumption, e.g. a coalition C has consideration for an agent $i \notin C$ if i has at least a given number of neighbors in C .

REFERENCES

- [1] S. Airiau and U. Endriss, 'Iterated majority voting', in *Proceedings of the First International Conference on Algorithm Decision Theory (ADT-09)*, volume 5783 of LNCS, pp. 38–49, (October 2009).
- [2] I. Ashlagi, P. Krysta, and M. Tennenholtz, 'Social context games', in *Internet and Network Economics*, 675–683, Springer, (2008).
- [3] R. J. Aumann, 'Acceptable points in general cooperative n-person games', in *Contribution to the Theory of Games*, eds., A. W. Tucker and R. D. Luce, volume IV of *Annals of Mathematics Studies*, 40, pp. 287–324. Princeton Univ. Press, (1959).
- [4] B. D. Bernheim, B. Peleg, and M. D. Whinston, 'Coalition-proof nash equilibria i. concepts', *Journal of Economic Theory*, **42**(1), 1–12, (1987).
- [5] F. Brandt, V. Conitzer, U. Endriss, J. Lang, A.D. Procaccia, and H. Moulin, *Handbook of Computational Social Choice*, Cambridge University Press, 2016.
- [6] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra, 'Using complexity to protect elections', *Communications of the ACM*, **53**(11), 74–82, (2010).
- [7] P. Faliszewski and A. D. Procaccia, 'AIs war on manipulation: Are we winning', *AI Magazine*, **31**(4), 53–64, (2010).
- [8] A. M. Feldman and R. Serrano, *Welfare economics and social choice theory*, Springer Science & Business Media, 2006.
- [9] M. Feldman and M. Tennenholtz, 'Partition equilibrium', in *Algorithmic Game Theory, Second International Symposium, SAGT 2009, Paphos, Cyprus, October 18-20, 2009. Proceedings*, eds., M. Mavronicolas and V. Papadopoulou, volume 5814 of LNCS, pp. 48–59. Springer, (2009).
- [10] A. Gibbard, 'Manipulation of voting schemes: a general result', *Econometrica: journal of the Econometric Society*, 587–601, (1973).
- [11] U. Grandi, A. Loreggia, F. Rossi, K. B. Venable, and T. Walsh, 'Restricted manipulation in iterative voting: Condorcet efficiency and borda score', in *Algorithmic Decision Theory*, 181–192, Springer, (2013).
- [12] M. Hoefer, M. Penn, M. Polukarov, A. Skopalik, and B. Vöcking, 'Considerate equilibrium', in *22nd International Joint Conference on Artificial Intelligence (IJCAI)*, (July 2011).
- [13] M. O. Jackson et al., *Social and economic networks*, volume 3, Princeton university press Princeton, 2008.
- [14] O. Lev and J. S. Rosenschein, 'Convergence of iterative voting', in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 611–618. International Foundation for Autonomous Agents and Multiagent Systems, (2012).
- [15] E. Maskin, 'Nash equilibrium and welfare optimality*', *The Review of Economic Studies*, **66**(1), 23–38, (1999).
- [16] R. Meir, M. Polukarov, J. Rosenschein, and N. Jennings, 'Convergence to equilibria in plurality voting', in *Proc. of 24th Conference on Artificial Intelligence (AAAI-10)*, pp. 823–828, (2010).
- [17] M. Messner and M. K. Polborn, 'Strong and coalition-proof political equilibria under plurality and runoff rule', *International Journal of Game Theory*, **35**(2), 287–314, (2007).
- [18] H. Moulin, *Axioms of cooperative decision making*, Cambridge University Press, 1991.
- [19] R. B. Myerson and R. J. Weber, 'A theory of voting equilibria', *American Political Science Review*, **87**(01), 102–114, (1993).
- [20] J. Nash, 'Non-cooperative Games', *The Annals of Mathematics*, **54**(2), 286–295, (1951).
- [21] S. Obraztsova, E. Markakis, M. Polukarov, Z. Rabinovich, and N. R. Jennings, 'On the convergence of iterative voting: how restrictive should restricted dynamics be?', in *AAAI 2015: Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 993–999, (2015).
- [22] B. Peleg and H. Peters, *Strategic Social Choice: Stable Representations of Constitutions*, Springer-Verlag Berlin Heidelberg, 2010.
- [23] A. Reijngoud and U. Endriss, 'Voter response to iterated poll information', in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 635–644. International Foundation for Autonomous Agents and Multiagent Systems, (2012).
- [24] R. Reyhani and M. Wilson, 'Best-reply dynamics for scoring rules', in *20th European Conference on Artificial Intelligence*. IOS Press, (2012).
- [25] M. A. Satterthwaite, 'Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions', *Journal of economic theory*, **10**(2), 187–217, (1975).
- [26] M. Sertel and R. Sanver, 'Strong equilibrium outcomes of voting games are the generalized condorcet winners', *Social Choice and Welfare*, **22**(2), 331–347, (2004).
- [27] S. Sina, N. Hazon, A. Hassidim, and S. Kraus, 'Adapting the social network to affect elections', in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 705–713. International Foundation for Autonomous Agents and Multiagent Systems, (2015).
- [28] A. Tsang and K. Larson, 'The echo chamber: Strategic voting and homophily in social networks', to appear in *AAMAS 2016*, (2016).
- [29] M. Voorneveld, *Potential Games and Interactive Decisions with Multiple Criteria*, Ph.D. dissertation, Tilburg University, 1999.

The Complexity of Deciding Legality of a Single Step of Magic: The Gathering

Krishnendu Chatterjee and Rasmus Ibsen-Jensen¹

Abstract. Magic: the Gathering is a game about magical combat for any number of players. Formally it is a zero-sum, imperfect information stochastic game that consists of a potentially unbounded number of steps. We consider the problem of deciding if a move is legal in a given single step of Magic. We show that the problem is (a) coNP-complete in general; and (b) in P if either of two small sets of cards are not used. Our lower bound holds even for single-player Magic games. The significant aspects of our results are as follows: First, in most real-life game problems, the task of deciding whether a given move is legal in a single step is trivial, and the computationally hard task is to find the best sequence of legal moves in the presence of multiple players. In contrast, quite uniquely our hardness result holds for single step and with only one-player. Second, we establish efficient algorithms for important special cases of Magic.

1 Introduction

Magic: the Gathering (henceforth, Magic) is a collectible card game where each player has the role of a mighty wizard that can cast spells like fireballs or summon for instance dragons, angels, or demons. To do so the wizards uses mana they get from lands or in other ways. The objective of each player is to win over the other wizards by killing them.

Legality of a move in a single step. We are interested in establishing complexity bounds for deciding whether a move is legal in a single step of a Magic game. Formally, Magic is a stochastic, imperfect information, zero-sum game over an unbounded number of turns. Each turn is divided into some number of phases and each phase consists of some number of steps, in which typically only one player has a choice. The only exception to the turn-based nature is that some cards require all players to perform some action at the same time.

Cards. Each card in the game represents a specific spell, creature, or land, and has in general very different properties. We describe the typical properties of Magic relevant for this work, and there are many slight deviations which we omit. Each card has a name, a color or colors (or is colorless) and consists of a drawing, illustrating the effect of the card, and a text box stating what the card does. Whenever we mention a card in the text we will write its name in bold font and include a figure with the card. Each card can be in different *zones*. The zones important for this paper are:

1. Nearly all cards start in some **library** from which cards can be drawn. The cards in any library are not visible to any player.

2. The cards in a **hand** of a player consists of the cards drawn from the library but not yet played. Cards in a hand can be played for some cost. The cards in a hand is only visible to that player.
3. The **stack** consists of cards and effects that have been played or triggered but are waiting to take effect. Whenever a card or effect is put on the stack, each player has, in turn, the opportunity to play more cards or effects. Whenever no player wants to add more to the stack, the top card or effect on the stack takes effect. Alternately, if the stack is empty, the next phase or turn starts. The content of the stack is visible to all players.
4. The cards on the **battlefield** consists of the cards that have a direct influence on the game currently and are visible to all players.
5. The cards in the **graveyard** consists of cards that have been used and are visible to all players.

Each card also has a card type. The important card types are:

1. **Instant/sorcery** which, when taking effect, has some immediate or short-term effect on the game and immediately goes to the graveyard.
2. **Artifact/enchantment** which, when taking effect, enters the battlefield and have some long-term effect.
3. **Planeswalker** which, when taking effect, enters the battlefield, with some amount of loyalty. On each of their turns, the controller of the planeswalker can pick an effect, which typically includes an increase or decrease (but not below 0) of loyalty. Planeswalkers can also be attacked to decrease their loyalty. Whenever the loyalty of a planeswalker falls below 1, they go to the graveyard.
4. **Creature** which, when taking effect, enters the battlefield. Most creatures can attack and block in the combat phase and all creatures have a power x and a toughness y , denoted x/y on the cards. The power is the amount of damage the creature would do and the toughness is how much damage a typical creature can take before dying and going to the graveyard. Especially, a creature with 0 toughness dies immediately.
5. **Land** which does not use the stack, but can only be played when the stack is empty. Each player can only play lands in his own turn and only one land on each turn. Most lands are used to generate mana which is often used in the cost of the other card types.

Trivia. Magic is a card game initially published in 1993, played by around twenty million players worldwide and there exist 15.919 different Magic cards currently. Magic has the fourth largest real-life tournament, and other than Poker variants, it has the largest tournament. Magic is the first collectible card game, a category including quite a few different games nowadays. While Magic is a single game, there exists many variants of the game, called formats, with currently

¹ IST Austria, Austria, email: {Krishnendu.Chatterjee,RIbsen}@ist.ac.at

22 different formats receiving some official support. Different formats differs in which cards are legal, basic rules, number of players and their relations (i.e. adversarial or team) and even how the decks of cards played with are constructed. The value of cards can vary greatly from basically nothing to \$27.302 for a single card in regular print.

One player variant: Goldfish. None of the official formats in Magic are single-player, because the main purpose is real-world tournaments with multiple players. However, the relevance of one-player version is that any lower bound on this simple version represents a stronger hardness result. We consider the most well-known one-player variant:

- **Goldfish.** In goldfish, the lone player is playing against a “player” that starts with no cards, never makes any choice and generally never does anything unless forced to – a so called goldfish. Along with the relevance mentioned above, the variant is used to provide a benchmark on how fast a deck can win against a player that does nothing relevant.

Research on Magic. Previous research works have focused on various aspects of Magic, for instance, considering different strategies for collecting Magic-cards ([3]) or used online auctions for Magic cards to test revenue for various auction types ([16]). [5] have shown that for Magic games with at least 6 players, there exist decks of cards, such that for all Turing machines, there exists a polynomial-length sequence of actions by the players, such that after the sequence, a player eventually wins iff the Turing machine terminates². While previous results on complexity consider multiple steps of the game, the problem we consider (i.e., the complexity of a single step) has not been considered before.

Research on other real-life games. There has been some research on the complexity of other real-life games. For instance, [12, 4] considered the popular real-time strategy computer game Star Craft. Also [11, 2] considered Bridge. Complexity of various generalized games have also been considered. For instance, checkers is EXPTIME-complete [18] and Othello PSPACE-complete [15], when the games are generalized to n by n boards. See also [14] for a survey about games and complexity. For the game Phutball, [10] showed that deciding if a legal move that wins immediately exists is NP-complete.

The rules of Magic. Magic has a very complex set of rules: The simple introductory rules used for playing the first few games is 16 pages long, but this does not suffice for the purpose of this article. The full set of rules is 210 pages long and is without pictures ([19]). Therefore, explaining the rules of the game is not part of the goal of this article. Instead, we will explain informally the key rules relating to where the complexity is coming from.

Result of this paper. We present three main results.

1. *Classification of requirements and restrictions.* We classify the requirements and restrictions on Magic cards into a few classes. (Blockers must be declared such that all restrictions are satisfied and as few requirements as possible are broken, see Section 2).
2. *Complexity in general.* We show that the legality of a move in a single step can be decided in coNP in general and is coNP -hard

² The construction has later been improved, see [6, 7, 8], however these texts are posts on forums and less polished

even in *goldfish* games (the single-player variant) measured in the number of objects.

3. *Efficient algorithms for special cases.* For two relatively small and nearly disjoint sets of cards rarely used in tournaments, consisting of 0.5% and 0.3% of all cards respectively, we show that finding a legal move in a single step in Magic games *not* using those cards are in DL (deterministic log-space) and P (PTIME) respectively. It follows from our results that in these cases deciding the legality of a move in a single step also has the same complexity.

Technical contributions. The key technical aspects are:



Figure 1. Card making tokens: **Captain's Call** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Greg Staples. Image used with permission

1. *Non-trivial complexity.* Note that we consider games with a constant number of players with a fixed and finite deck. Such restriction in other games lead to trivial complexity (e.g., normal chess is trivial in the sense that only a finite number of configurations can be achieved and an optimal strategy can be found in a constant, but very large, amount of time). The reason this does not happen in Magic is the concept of tokens and the possibility of replaying cards. A token is an object that is not a card. For instance **Captain's Call** creates 3 creatures, though it is only a single card. It is possible to then get **Captain's Call** back to hand (without losing other cards) allowing it to be played again, using some combinations of cards. By playing it many times, any number of tokens can be created.

2. *Relationship with classical graph algorithm.* A key technical contribution is that we establish a close relationship between the problem we consider and the classical graph algorithm of *min-cost-flow*. While the min-cost-flow problem can be solved in P, we show that the problem we consider requires a natural generalization: along with the standard capacity constraints (which specify that the flow is at most the capacity), we require new constraints that specify that the flow is either 0 or at least a given threshold. Our result shows that this natural generalization makes the problem coNP -complete. On the other hand, if certain cards are not used, we present a many-to-one reduction from our problem to the min-cost-flow problem.

Significance. The main significant implications are:

1. *Magic-specific result.* There exists an online tool for playing Magic, Magic Online ([20]), that checks the legality of a step. However, the tool considers a special case of Magic where at most 200 tokens are allowed per player (i.e., it gives an algorithm for the special case when the size of the input problem is small). In contrast, we first establish complexity of the general problem, showing that it is coNP -hard (even for single-player) and no efficient (polynomial-time) algorithm exists in general (unless $\text{coNP} = \text{P} = \text{NP}$). Second, we present efficient (polynomial-time) algorithms for two special cases, where a small number of cards are not used, but there is no restriction on the number of tokens. Thus we present efficient solution to orthogonal special cases as compared to the online tool.
2. *Uniqueness of complexity.* We consider games with a constant number of players where each has a *fixed and finite* deck. Moreover, we consider the legality of a single step. In most real-life games, either of the above restrictions would lead to trivial computational complexity. The first restriction typically means that the problem size is constant. The second restriction implies that we do not consider strategic choices over multiple steps. In most real-life games the complexity comes from strategic choices (i.e., that maximizes the probability to win over several steps). Quite uniquely our lower bound is for legality of a single-step in a single-player variant of Magic, and the complexity we obtain is neither from the fact that the game is stochastic or imperfect information.
3. *General graph algorithm.* Our technical contribution considers a generalization of the classic min-cost-flow problem on graphs with additional constraints that either the flow is zero or at least a threshold value. We show that such problem is coNP -complete, and show that for real-life game problems (such as special cases of Magic) the classical min-cost-flow algorithm can be used.

2 The complexity of a single step

In this section we consider the complexity of a single step of a Magic game. Concretely we consider the complexity of deciding if in a given step, a given choice is legal. For most types of steps, besides the declare blockers step of the combat phase, this can easily be done in log-space (hence in polynomial time). The reason for the log-space complexity is that the legality of a choice (other than declaration of blockers) is *local*, in the sense that legality can be decided by only considering pairs of objects (i.e. a card c might target another card c' and is legal iff c' can be targeted by c) and counts (i.e. a card c might target k cards, which would be legal if c can have k targets). This can all be done in DL (deterministic log-space). We will thus focus on the declaration of blockers and argue that it is coNP -complete to find a legal declaration of blockers. We describe the problem using the min-cost-flow terminology (since min-cost flow is in P , see [9], and our problem is not, some parts require generalization of min-cost flow).

Min-cost flow. The min-cost flow problem consists of a directed graph $G = (V, E)$, a *capacity function* $c : V \cup E \rightarrow \mathbb{N}$ and a weight function $w : E \rightarrow \mathbb{Z}$. Also, there is a designated *source state* s and a designated *sink state* t . For a map $M : E \rightarrow \mathbb{N}$, let $\Delta_M : V \rightarrow \mathbb{Z}$ be $\Delta_M(v) = \sum_{(u,v) \in E} M(u,v) - \sum_{(v,u) \in E} M(v,u)$. A *flow* is a map $f : E \rightarrow \mathbb{N}$ such that (1) $\Delta_f(s) \geq 0$; (2) $\Delta_f(t) \leq 0$; and (3) for all $v \in (V \setminus \{s, t\})$ the number $\Delta_f(v)$ is 0 (intuitively, there

is a flow from s to t and for other vertices the incoming flow matches the outgoing flow). A flow f is *feasible* if (1) for all $v \in (V \setminus \{s, t\})$ we have $\sum_{(u,v) \in E} f(u,v) \leq c(v)$; and (2) for all $e \in E$ we have $f(e) \leq c(e)$. The *value* of a flow f is $\text{val}(f) = \sum_{e \in E} f(e) \cdot w(e)$. The solution to a min-cost flow problem is a feasible flow with maximum value among all feasible flows. This problem is in P ([9]).

The declaration of blockers step. The declaration of blockers step is a part of the combat phase of a turn. In the combat phase of a turn first a set of creatures A , controlled by P , is declared as *attacking* P' by P (in general with more players each of the creatures can attack different players other than P). For the goldfish format, the player P is the goldfish player (in our lower bound), and there exist cards that ensure that all creatures must be declared attacking (this is the only option for the goldfish player). The creatures controlled by P' is the set B . After this step, if A is not empty, the defending player P' (in general, a player is defending if he is attacked by some creature) can, for each creature $a \in A$ that attacks him, declare that some, perhaps empty, subset $B^a \subseteq B$ is going to block a . We will let B^A be the set $\bigcup_{a \in A} B^a$ of blocking creatures. For any $b \in B$, we also define b^A as $\{a \mid b \in B^a\}$, i.e. the creatures b blocks. The complexity of finding a legal declaration of blockers comes from rule 509.1c ([19]), which states, paraphrased, that a declaration of blockers is legal if it satisfies all *restrictions*, while maximizing the number of satisfied *requirements*.

The base case: No restrictions or requirements. If there are no requirements or restrictions involved, each creature in A can be blocked by any number of creatures in B , but each creature in B can only block one creature of A and any such choice of blockers is legal. Using terminology from min-cost flow, the graph G is a bipartite graph, with the source on the right side and the sink on the left. Each state (besides the sink) on the left side corresponds to a creature in B and each state (besides the source) on the right side corresponds to a creature in A , and the states on the left side has capacity 1 (besides the sink, which has infinite capacity) while the states on the right have infinite capacity. The source has an outgoing edge to each state in B , each state in B has an outgoing edge to each state in A and each state in A has an outgoing edge to the sink. The edges between A and B each have capacity 1 and the edges to/from the sink/source have each infinite capacity. Also, each edge has weight 0.

Restrictions and requirements. The complexity of the problem comes from the requirements and restrictions. We will explain requirements and restrictions in how they modify the min-cost flow instance from the base case.

Restrictions. Rule 509.1b ([19]) states that for a declaration of blockers to be legal, every restriction must be satisfied. A restriction is statement that says that a specific creature cannot block/be blocked unless some condition is met. This, in itself, does not cause any complexity, since for a fixed declaration of blockers, checking if a fixed restriction is satisfied is easy. We now consider the following five types of restrictions³:

1. **Local restrictions.** Local restrictions are restrictions of the form: $a \in A$ cannot be blocked by $b \in B$ (i.e. a local restriction is satisfied if it is satisfied for each pair of affected creatures). Concretely, a could have flying and b could have neither reach or flying. A local restriction between $a \in A$ and $b \in B$ can be modeled in our min-cost flow instance by removing the edge (a, b) . For instance,

³ To our knowledge, all restrictions fit into precisely one type



Figure 2. The cards are sorted from left to right. First row - cards with various types of non-capacity restrictions: (1) Local restriction: **Razortooth Rats** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Carl Critchlow. Image used with permission; (2) Local counting restriction $k = 2$: **Boggart Brute** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Igor Kieryluk. Image used with permission; (3) Local counting restriction $k = 3$: **Guile** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Zoltan Boros & Gabor Szikszai. Image used with permission; Second row - continued: (4) Local counting restriction $k = |B|$: **Tromokratis** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Matt Stewart. Image used with permission; (5) Global counting restriction $k = 1$: **Silent Arbiter** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Mark Zug. Image used with permission; (6) Global counting restriction $k = 2$: **Caverns of Despair** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Harold McNeill. Image used with permission.

Razortooth Rats creates a local restriction with all non-artifact non-black creatures.

2. **Local counting restrictions.** Local counting restrictions are restrictions of the form: $a \in A$ cannot be blocked except by k or more creatures. Here, concretely, there exists cards for which k is 2 (i.e. cards with the ability **Menace**, like **Boggart Brute**), 3 (there are six cards, for instance **Guile**) and $|B|$ (the card **Tromokratis**). Local counting restrictions have no nice interpretation in min-cost

flow terminology (note that there must be something for the problem to be coNP-complete, since min-cost flow can be solved in P, see [9]). A local counting restriction on $a \in A$ corresponds to saying that a have either flow 0 or flow greater than k in min-cost flow.

3. **Global counting restrictions.** Global counting restrictions are restrictions of the form: no more than k creatures can block. Here, concretely, there exists cards for which k is 1 (the cards **Silent**



Figure 3. Two first cards of first row - cards that can create capacity restrictions:

(1) Increase blocker capacity by 1: **Iona's Blessing** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by David Gaillet. Image used with permission;

(2) Set capacity of attacker to 1:

Alpha Authority ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Ron Spencer. Image used with permission.

Last card on first row and cards in second row - cards with various kinds of requirements:

(1) Generic attacker requirement: **Irresistible Prey** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Jesper Ejsing. Image used with permission;

(2) Generic blocker requirement: **Culling Mark** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Tomasz Jedruszek. Image used with permission;

(3) Special generic requirement: **Nacatl War-Pride** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by James Kei. Image used with permission;

(4) Specific requirement: **Hunt Down** ©2016 Wizards of the Coast LLC in the USA & other countries. Illustration by Christopher Moeller. Image used with permission.

Arbiter and **Dueling Grounds**) and 2 (the card **Caverns of Despair**). While there is no straightforward interpretation in min-cost flow, the global counting restrictions are easy to handle, by simply considering each creature or pair of creatures in B (depending on whether k is at least 1 or 2) as the set of creatures that can block and then solving the problem with only those creatures in B (except that any **Tromokratis** cannot be blocked in this case, assuming that $|B| > 2$ initially).

4. **Blocker capacity restriction.** Blocker capacity restrictions are restrictions of the form: $b \in B$ must be such that $|b^A| \leq k$. By default there is a blocker capacity restriction on each creature for $k = 1$, but k can be changed to any natural number⁴, using for instance, a sufficiently large number of **Iona's Blessing**. A blocker capacity restriction on $b \in B$ with $k \in \mathbb{N}$ is modeled in min-cost

⁴ It can also be infinite. However, there is no difference between infinite and $k \geq |A|$

flow by having a capacity of k on b .

5. **Attacker capacity restriction.** Attacker capacity restrictions are restrictions of the form: $a \in A$ must be such that $|B^a| \leq k$. Currently k can only be 1, which can be achieved by for instance using **Alpha Authority**. A attacker capacity restriction on $a \in A$ with $k \in \mathbb{N}$ can be modeled in min-cost flow with capacity of k on a .

Requirements. Rule 509.1c ([19]) states that the declaration of blockers must maximize the number of requirements satisfied, while not breaking any restrictions. A requirement is an ability that says that a specific creature must block/be blocked (perhaps by a specific set of creatures). In min-cost flow the requirements can be modeled as weights on edges⁵. We focus on three requirements⁶:

1. **Generic attacker/blocker requirements.** Generic attacker (resp. blocker) requirements are parameterised by an attacker $a \in A$ (resp. blocker $b \in B$) and are satisfied if a is blocked (resp. if b blocks). Generic attacker (resp. blocker) requirements can be modeled in min-cost flow by having two edges pointing to a from the source (resp. away from b to the sink), one with unbounded capacity and 0 weight and another with capacity 1 and weight equal to the number of times the creature is affected by the requirement. The requirement could for instance be caused by **Irresistible Prey** (resp. **Culling Mark**).
2. **Special generic requirements** are parametrised by an attacker $a \in A$ and are satisfied if a is blocked by exactly one creature. The requirement has no simple interpretation in min-cost flow, but any flow that has a flow of 1 through a satisfies the requirement. The requirement is only in effect if a is a **Nacatl War-Pride**.
3. **Specific requirements** are parametrised by an attacker $a \in A$ and blocker $b \in B$ and are satisfied if $b \in B^a$. Specific requirements could for instance be caused by **Hunt Down**. We can model k specific requirements between $a \in A$ and $b \in B$ by having a weight of k on the edge from a to b in min-cost flow.

Remark 1 Finding the categories. *Categorizing the requirements and restrictions of all cards in Magic is a demanding task (because of the number of cards). To do so we made a case analysis with many thousands of cases. We do not explicitly include the very technical case analysis, but only the outcome, because of the huge size of the case analysis.*

2.1 Complexity bound in general case

Inclusion in coNP. For a fixed declaration of blockers, it is easy to check that all restrictions are satisfied. In case any restriction is not satisfied, it is easy to find a declaration of blockers satisfying all (concretely, not declaring any blockers always satisfies all restrictions). It is also easy to count the number of requirements satisfied. Hence, the given declaration of blockers is legal precisely if no declaration can be found that satisfies more requirements while satisfying all restrictions. Note that a declaration of blockers can be described as the sets B^a for each a . This can be done in $O(|A| \cdot |B|)$ space, giving a polynomial sized witness. Note that even in case of many players,

⁵ Except for the requirement caused by **Nacatl War-Pride**, which does not have a simple interpretation in min-cost flow.

⁶ To our knowledge, all requirements fit into precisely one type

since the declaration of blockers is a turn-based step, the coNP upper bound holds.

coNP-hardness. Our reduction will be from exact cover by 3-sets. The exact cover by 3-sets problem is as follows: A set of elements E and a set $S \subseteq 2^E$ of sets of elements of E is given, such that $|s| = 3$ for each $s \in S$, and the problem is to decide if there exists a subset S' of S such that $\bigcup_{s \in S'} s = E$ and $s \cap s' = \emptyset$ for all $s, s' \in S'$. The exact cover by 3-sets problem is NP-complete ([13]).

Consider an instance of the exact cover by 3-sets problem and we will construct a combat instance and a declaration of blockers which is not legal iff there exists a satisfying set S' for the exact cover by 3-sets instance. There is a special attacker **Tromokratis**⁷, which has, together with some arbitrary creature $b \in B$, been targeted by **Hunt Down** $|E| - 1$ times (hence, blocking it with all creatures in B will satisfy $|E| - 1$ requirements). Besides that the attackers will play the role of the sets of S and the blockers the elements of E . Each attacker a , besides the **Tromokratis**, is a **Guile**⁸ and will correspond to a set $(e_1, e_2, e_3) \in S$ and **Hunt Down** has been cast on a and the blocker b_1 corresponding to e_1 (so that there is a specific requirement between a and b_1). Similarly for a and e_2 and a and e_3 . Hence, blocking a with the creatures corresponding to e_1, e_2 and e_3 will satisfy 3 requirements. Each blocker can block only 1 creature (as is default) and each attacker can be blocked by any number of creatures (as is default). Blocking **Tromokratis** with all creatures is not legal iff there exists an exact cover by 3-sets.

Note that we are giving creatures to the goldfish player. This is sometimes done even in real-life tournament Magic games, for instance it often happens that a player of the Oath of Druids deck will give creatures to his opponent.

Remark 2 How to make the setup in a concrete game of Magic: *This remark is meant for people familiar with the rules of Magic and we do not include illustrations of the cards used. Play **Imperious Perfect**, **Intruder Alarm**, **2 Birds of Paradise** (these first cards gives an arbitrary amount of mana), **Djinn Illuminatus**, **Tromokratis**, **Concordant Crossroads**, **Vedalken Orrey**, **Fumiko the Lowblood** and **Guile**. **Djinn Illuminatus** allows us to copy spells an arbitrary number of times. On the opponents turn use **Spitting Image** to create enough **Guiles**, **Hunt Down** to make the requirements and **Donate** to give the attackers away (the cards can be played at that point because of **Vedalken Orrey**). **Fumiko the Lowblood** and **Concordant Crossroads** makes the creatures attack, without choice.*

Theorem 1 *The complexity of a single step in a Magic game is coNP-complete, and the hardness result holds even in the special case of goldfish.*

2.2 Special cases with better complexities

In this section we consider several special cases, and show that they have much better complexity. Note that the upper bounds we get in this section is for any number of players because the declare blockers step is turn-based.

Restriction 1: Neither local counting restrictions (except for Tromokratis) nor Nacatl War-Pride. As argued above besides lo-

⁷ which must either be left unblocked or blocked by all

⁸ which must either be left unblocked or blocked by 3 or more creatures

cal and global counting restrictions and **Nacatl War-Pride**, every requirement and restriction can be encoded in the standard min-cost flow problem. We explain how to handle **Tromokratis** and global counting restrictions.

- **Handling Tromokratis.** For any number $k \leq |A|$, we can find the optimal solution such that precisely k **Tromokratis** are blocked, because of the following: For each **Tromokratis** $a \in A$, the number of requirements satisfied by blocking a with all creatures in B , is independent of how else the creatures in B block. Therefore, we can sort the **Tromokratis** after how many requirements are satisfied, block the first k with all creatures in B , and then solve the sub-problem where the set of attackers is A , except for the **Tromokratis**, and each creature in B can block k less creatures.
- **Handling global counting restrictions.** We can handle global counting restrictions parametrised with k (even in the presence of local counting restrictions) straightforwardly since $k \in \{1, 2\}$ (it is not clear if the problem is in polynomial time for k as a parameter, but k is at most 2 for global counting restrictions in Magic). That is, we simply guess the creatures in B^A and then solve the problem with only those creatures. We can, in the presence of global counting restrictions also handle local counting restrictions easily.

For an in depth complexity analysis see Section 2.3, where we show the complexity is $O(nm^2 \log n + mn^2 \log^2 n + n^2 \log C)$ time, where, in our min-cost flow instance, n is the number of states and m the number of edges (i.e. $n = 2 + |A| + |B|$ and $m \leq |A| + |B| + |A| \cdot |B|$), and C is the greatest weight on an edge (and thus C is at most the number of requirements). Hence, the problem is in polynomial time if none of the 37 cards (of which 31 have or grants local counting restriction 2, i.e. Menace) that have or grants a local counting restriction, besides **Tromokratis**, but also including **Nacatl War-Pride**, is in the game. None of these 37 cards, which is less than 0.3% of the distinct Magic cards, are heavily used. Menace is a new ability that might be used in the future.

Restriction 2: Removing requirements. Another simple way to make the problem easier is to remove the cards that have or grants requirements. There are 63 cards, which is less than 0.5% of all distinct Magic cards, that causes requirements on blocking, none of which is used often. Note that only the card **Nacatl War-Pride** is in the 63 cards removed from this special case as well as the 37 cards removed in the above considered special case. A declaration of blockers is then legal iff it satisfies all restrictions. For each type of restriction, as describe above, it is easy to check in DL if the restriction is satisfied. Hence, this special case is in $DL \subseteq P$.

Restriction 3: Removing tokens. Another simple way is to remove all roughly 830 producers of tokens (i.e. creatures which are not cards), which corresponds to less than 6.1% of all distinct Magic cards, some of which are used heavily. At this point, the number of creatures in play is bounded by the number of Magic cards in existence and the problem can be solved by considering each possible declaration of blockers. Since the problem is bounded, it is trivial. (We consider this restriction, since it is a reasonably obvious way to make the problem simpler – it does not lead to an efficient algorithm though). A variant of Restriction 3 is used in the online implementation of Magic ([20]) where at most 200 tokens can be created per player (after which no more tokens appear when one tries to make any).

Theorem 2 *The complexity of a single step in a Magic game with any of the Restriction 1-3, no matter the number of players, can be solved in polynomial time.*

Also note that for Restriction 1-3 it follows from our results that the legality of a given move in a single step can be decided in the same complexity as mentioned above.

2.3 In depth analysis of Restriction 1

We will, in this section, give a more in depth complexity analysis, in case there are neither local counting restrictions (except for **Tromokratis**) nor **Nacatl War-Pride** (that is, of Restriction 1). To get a better complexity bound we consider the min-cost flow instance created in this case in Section 2.2. Observe that it has $n = |A| + |B|$ states. Also, there is an edge from $a \in A$ to $b \in B$ unless there is a local restriction between them. Hence, $m = |A| \cdot |B| - c$, where c is the number of local restrictions between different pairs. Parametrising like this, Orlin's strongly polynomial algorithm for minimum cost flow ([17]) runs in time $O(m^2 \log n + mn \log^2 n)$.

We consider three cases:

1. If there is a global counting restriction of k (which is either 1 or 2), we can as explained in Section 2.2, consider each creature or pair of creatures in B in turn and solve the resulting instance when B only consists of those creatures. This takes n^2 times the time for the case when B consists of two creatures.

Remark 3 *Observe that if the case where $|B| = 2$ can be solved in $O(n \cdot f(n) + g(n))$ time, where $f(n)$ is a sum of terms, each of which is at least constant, then this case can be solved in time $O(nm \cdot f(n) + n^2 g(n))$. This is because the time for a pair of states u, v is then $O((d_u + d_v) \cdot f(d_u + d_v) + g(d_u + d_v))$, where d_u and d_v is the degree of u and v respectively. Hence,*

$$\begin{aligned} & \sum_{u,v} O((d_u + d_v) \cdot f(d_u + d_v) + g(d_u + d_v)) \\ & \leq O(n^2 g(n)) + f(n) \cdot \sum_{u,v} O((d_u + d_v)) \\ & = O(n^2 g(n)) + f(n) \cdot \sum_u O(n \cdot d_u + m) \\ & = O(nmf(n) + n^2 g(n)) \end{aligned}$$

This shows that using Orlin's method ([17]), this case would use $O(m^2 n^2 \log n + m^2 n \log^2 n) = O(m^2 n^2 \log n)$ time.

2. If **Tromokratis** is not in the game and there are no global counting restrictions, then it is easy to solve it in time $O(m^2 \log n + mn \log^2 n)$, since it is a standard flow problem.
3. Otherwise, if there is no global counting restrictions and **Tromokratis** is in play then we can guess the number Δ of **Tromokratis** each creature should block (observe that this is bounded by the number of creatures in A), by simply trying all options. We can then find the Δ **Tromokratis** that, if blocked, will give the most satisfied requirements, by sorting them after how many requirements will be satisfied (note that the number of satisfied requirements is independent of which other creatures a creature block, and thus doing it greedily will find the best set). We can then reduce the capacity of each blocker by Δ , remove the **Tromokratis** and solve the resulting instance in

$O(m^2 \log n + mn \log n)$ using Orlin's algorithm ([17]). This requires $O(nm^2 \log n + mn^2 \log^2 n)$ time.

Observe that the worst case time, if we use Orlin's algorithm ([17]) in the first case, will be the time for the first step. That said, there exists specialized algorithms for this case, where one side is of constant size, see [1], using time $O(n + \log C)$ where C is the greatest weight. Specifically, C is in this case the maximum number of times any single state (resp. pair of states) are effected by a generic (resp. specific) requirement. This will ensure that the time for case 1 is then $O(nm + n^2 \log C)$ and the total time is $O(nm^2 \log n + mn^2 \log^2 n + n^2 \log C)$.

3 Conclusion

In this work we establish the complexity of deciding the legality of a move in a single step of Magic. In sharp contrast to existing real-life games, where legality of a move in single step is trivial, we establish coNP-completeness for legality of a move in a single-step of a single-player variant of Magic, which is quite unique and should be of wide interest. Moreover, our result is established by showing a close connection with a generalization of the min-cost-flow problem, which is also of independent and general interest. While we show that in general the legality of a move in a single-step is coNP-complete, we present efficient algorithms for special cases which are different from existing tools but these special cases are widely used. Hence our algorithms are also practically relevant for automated tools to analyzed important special cases in Magic.

References

- [1] R. K. Ahuja, J. B. Orlin, and C. Stein. Improved Algorithms for Bipartite Network Flow. *SIAM J. Comput.*, 23(5):906–933, 1994.
- [2] E. Bonnet, F. Jamain, and A. Saffidine. On the Complexity of Trick-Taking Card Games. In *IJCAI 2013*, 2013.
- [3] R. A. Bosch. Optimal Card-Collecting Strategies for Magic: The Gathering. *The College Mathematics Journal*, 31(1):pp. 15–21, 2000.
- [4] M. Buro and A. Kovarsky. Concurrent Action Execution with Shared Fluents. In *AAAI 2007*, pages 950–955, 2007.
- [5] A. Churchill. Magic: the Gathering is Turing Complete. <http://tinyurl.com/olg28d5>, 2012.
- [6] A. Churchill et. al. Magic: the Gathering is Turing Complete (forum page 1). <http://tinyurl.com/pv3n2lg>, 2012.
- [7] A. Churchill et. al. Magic: the Gathering is Turing Complete (forum page 2). <http://tinyurl.com/naq9jmc>, 2012.
- [8] A. Churchill et. al. Magic: the Gathering is Turing Complete (forum page 3). <http://tinyurl.com/ow7obh6>, 2014.
- [9] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.
- [10] E. D. Demaine, M. L. Demaine, and D. Eppstein. Phutball Endgames are Hard. *CoRR*, cs.CC/0008025, 2000.
- [11] I. Frank and D. A. Basin. A theoretical and empirical investigation of search in imperfect information games. *Theor. Comput. Sci.*, 252(1-2):217–256, 2001.
- [12] T. Furtak and M. Buro. On the Complexity of Two-Player Attrition Games Played on Graphs. In *AIIDE 2010*, 2010.
- [13] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
- [14] R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation*. A. K. Peters, Ltd., Natick, MA, USA, 2009.
- [15] S. Iwata and T. Kasai. The Othello game on an $n \times n$ board is PSPACE-complete. *Theoretical Computer Science*, 123(2):329 – 340, 1994.
- [16] D. Lucking-Reiley. Using Field Experiments to Test Equivalence between Auction Formats: Magic on the Internet. *The American Economic Review*, 89(5):pp. 1063–1080, 1999.
- [17] J. Orlin. A Faster Strongly Polynomial Minimum Cost Flow Algorithm. In *STOC '88*, pages 377–387, 1988.
- [18] J. M. Robson. N by N Checkers is Exptime Complete. *SIAM Journal on Computing*, 13(2):252–267, 1984.
- [19] Wizards of the Coast. Magic: the Gathering Comprehensive Rules. <http://tinyurl.com/qf6hlx4>, 2015.
- [20] Wizards of the Coast. Magic: the Gathering Online. <http://tinyurl.com/h95z67n>, 2016.

Description Logics Reasoning w.r.t. General TBoxes Is Decidable for Concrete Domains with the EHD-Property

Claudia Carapelle and Anni-Yasmin Turhan¹

Abstract. Reasoning for Description logics with concrete domains and w.r.t. general TBoxes easily becomes undecidable. However, with some restriction on the concrete domain, decidability can be regained. We introduce a novel way to integrate concrete domains \mathcal{D} into the well-known description logic \mathcal{ALC} , we call the resulting logic $\mathcal{ALC}^P(\mathcal{D})$. We then identify sufficient conditions on \mathcal{D} that guarantee decidability of the satisfiability problem, even in the presence of general TBoxes. In particular, we show decidability of $\mathcal{ALC}^P(\mathcal{D})$ for several domains over the integers, for which decidability was open. More generally, this result holds for all negation-closed concrete domains with the EHD-property, which stands for ‘the existence of a homomorphism is definable’. Such technique has recently been used to show decidability of CTL^* with local constraints over the integers.

1 Introduction

Description Logics (DLs) are a collection of knowledge representation formalisms with well-founded semantics. Most DLs are (decidable) fragments of First Order Logic (FO). They are employed nowadays in a range of application areas such as the bio-medical field or the semantic web, and are the foundations of the web ontology language OWL 2 [20]. DLs are an excellent tool to represent abstract knowledge and to reason over it, but practical applications often require concrete properties with values from a fixed domain, such as integers or strings and to support built-in predicates.

In [1], DLs were extended with *concrete domains*, where partial functions map objects of the abstract domain to values of the concrete domain. The resulting logic $\mathcal{ALC}(\mathcal{D})$ extends the standard DL \mathcal{ALC} by *concrete domain restrictions* over a concrete domain \mathcal{D} . Concrete domain restrictions can be used for building complex concepts based on concrete qualities of their instances such as the age, temperature or even measured values. For instance, the following GCI of $\mathcal{ALC}(\mathcal{D})$ -concepts:

$$\text{motor-vehicle-driver} \sqsubseteq \text{Person} \sqcap \exists \text{has-age.} \geq 18$$

requires that drivers of a motor vehicle are at least 18 years old. A concrete domain restriction can connect several abstract objects via *feature-paths*, i.e. paths of functional roles, and assert a predicate of arbitrary arity for concrete quantities of those objects. Concrete domains are incorporated in a weakened form in OWL as data-types for which only unary predicates are admitted [20].

If definitorial, acyclic TBoxes are used, then reasoning for \mathcal{ALC} extended by concrete domains that are *admissible* is decidable

[1]. Reasoning can become undecidable in the presence of general TBoxes [11, 15] for such DLs. There have been several attempts to regain decidability for reasoning in $\mathcal{ALC}(\mathcal{D})$ with general TBoxes. Some approaches simply restrict concrete domain restrictions to unary predicates [10] or to feature-paths of length 1 [9]. These restrictions limit the modelling capabilities severely. Lutz and Miličić took a different approach and showed that if a concrete domain respects a criterion called *ω -admissibility*, then satisfiability for $\mathcal{ALC}(\mathcal{D})$ with general TBoxes is decidable [16]. The condition of *ω -admissibility* essentially allows to lift local satisfiability of (connected) concrete domain parts to global satisfiability by requiring compactness and that the concrete domain parts need to conform on the predicates asserted for the shared objects. This condition indicates decidability of DL reasoning for some concrete domains, for instance, the RCC8 relations and the Allen relations over the real numbers [16]. However, several interesting domains do not satisfy *ω -admissibility*, for instance, the ones based on non-dense numerical sets, as the integers or the natural numbers. In [14] Lutz considers a concrete domain over the rational numbers, and proves that reasoning w.r.t. general TBoxes is decidable. Such domain can, however, not be used to reasonably represent some situations: certain concrete features, such as ‘number of children’, cannot possibly be fractions.

In this paper we devise a new criterion for concrete domains that guarantees decidability of the satisfiability problem in the presence of general TBoxes. This criterion holds also for some concrete domains that are known to be not *ω -admissible*, such as the integers. To this end we introduce the new DL $\mathcal{ALC}^P(\mathcal{D})$ that uses path constraints instead of concrete domain restrictions. Unlike the latter, which only allow feature-paths to connect an individual and a concrete value, path constraints can use the full expressiveness of role-paths. This enables to model for instance ‘person who only has younger siblings’, as an individual whose age is greater than that of all his siblings, where the sibling relation need not be functional. Furthermore, $\mathcal{ALC}^P(\mathcal{D})$ admits Boolean combinations of concrete domain predicates in path constraints.

We show decidability of the satisfiability problem of $\mathcal{ALC}^P(\mathcal{D})$ -concepts w.r.t. general TBoxes if \mathcal{D} (1) is *negation-closed*, which requires that the complement of each (atomic) relation is effectively definable by a positive existential first-order formula, and (2) has the EHD-property, which stands for ‘the existence of a homomorphism is definable’, expressing the ability of a certain logic L to distinguish between those structures which can be mapped to \mathcal{D} by a homomorphism and those who cannot. Our approach to show decidability of $\mathcal{ALC}^P(\mathcal{D})$ with concrete domains that fulfill the above conditions is an adaptation of the EHD-method, used in [6, 7] for CTL^* and ECTL^* . This, in turn, uses a recent decidability result by Bojańczyk and Toruńczyk for $\text{WMSO}+\text{B}$ over infinite trees, an extension of

¹ Technische Universität Dresden, Institute for Theoretical Computer Science, SFB HAEC; email: firstname.lastname@tu-dresden.de

weak monadic second order logic by the bounding quantifier B [3].

The idea for testing satisfiability of an $\mathcal{ALCC}^P(\mathcal{D})$ -concept C w.r.t. an $\mathcal{ALCC}^P(\mathcal{D})$ -TBox \mathcal{T} is to proceed in two steps. First, an ordinary \mathcal{ALC} interpretation is built that satisfies an abstracted version of C and \mathcal{T} , where each path constraint is replaced by a fresh concept name. Second, this interpretation is used to generate a so-called constraint graph, which is a structure for storing the contribution of the constraints that were abstracted away. We show that deciding whether such a constraint graph allows a homomorphism to the concrete domain is enough to guarantee that the constraints are satisfied. In contrast to the mentioned CTL variants, $\mathcal{ALCC}^P(\mathcal{D})$ is multi-modal and uses features, i.e. functional roles, which required some adaptation to apply the techniques from [6, 7].

By the newly established criterion for decidability of $\mathcal{ALCC}^P(\mathcal{D})$ w.r.t. TBoxes, we confirm what the authors of Lutz and Miličić have conjectured: that ω -admissibility is a sufficient, but not necessary condition for decidability. We show, in fact, that reasoning with non ω -admissible concrete domains over the natural numbers and the integers w.r.t. general TBoxes is decidable. We also show that it is possible to consider an extension of the concrete domain over the rational numbers presented in [12, 11], that allows to test whether a certain concrete value is an integer.

This paper is structured as follows. In the next two sections we give some preliminary notions and introduce the DL $\mathcal{ALCC}^P(\mathcal{D})$ with some of its properties. Section 4 explains the EHD-method and shows decidability of DL reasoning for $\mathcal{ALCC}^P(\mathcal{D})$ with negation-closed concrete domains that have the EHD-property. As customary, the paper ends with conclusions and future work. All omitted or shortened proofs can be found in full detail in [8].

2 Preliminaries

Before we define our DL $\mathcal{ALCC}^P(\mathcal{D})$, we introduce some basic notions needed later on in the technical constructions. A (relational) signature $\sigma = \{R_1, R_2, \dots\}$ is a countable (finite or infinite) set of relation symbols. Every relation symbol $R \in \sigma$ has an associated arity $\text{ar}(R) \geq 1$. A σ -structure is a tuple $\mathcal{A} = (A, R_1^A, R_2^A, \dots)$, where A is a non-empty set and for each $R \in \sigma$, $R^A \subseteq A^{\text{ar}(R)}$ is the interpretation of the relation symbol R in \mathcal{A} , that is an $\text{ar}(R)$ -ary relation over A .

Example 1. A simple example of a $\{=, <\}$ -structure is $\mathcal{Z} = (\mathbb{Z}, =^{\mathbb{Z}}, <^{\mathbb{Z}})$, where $=^{\mathbb{Z}}$ and $<^{\mathbb{Z}}$ are defined as expected, namely as $\{(a, b) \in \mathbb{Z}^2 \mid a = b\}$ and $\{(a, b) \in \mathbb{Z}^2 \mid a < b\}$, respectively.

We often identify the relation R^A with the relation symbol R . In the example above, then, we would simply write $(\mathbb{Z}, =, <)$. For a σ -structure \mathcal{A} and a τ -structure \mathcal{B} such that $\tau \subseteq \sigma$, a homomorphism from \mathcal{B} to \mathcal{A} is a mapping $h : B \rightarrow A$ such that for all $R \in \tau$ and all tuples $(b_1, \dots, b_{\text{ar}(R)}) \in B^{\text{ar}(R)}$ we have

$$(b_1, \dots, b_{\text{ar}(R)}) \in R^{\mathcal{B}} \Rightarrow (h(b_1), \dots, h(b_{\text{ar}(R)})) \in R^{\mathcal{A}}.$$

We write $\mathcal{B} \preceq \mathcal{A}$ if there is a homomorphism from \mathcal{B} to \mathcal{A} . Note that we do not require this homomorphism to be injective.

We shortly introduce MSO and WMSO+B, for a more detailed introduction we refer the reader to [3, 18]. We fix countably infinite sets V_e and V_s of element variables and set variables, respectively. Monadic second-order logic (MSO) is the extension of first-order logic (FO) where also quantification over sets is allowed. MSO-formulas over a signature σ are defined by the following grammar, where $R \in \sigma$, $x, y, x_1, \dots, x_{\text{ar}(R)} \in V_e$ and $X \in V_s$:

$$\varphi := R(x_1, \dots, x_{\text{ar}(R)}) \mid x = y \mid x \in X \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \exists x\varphi \mid \exists X\varphi.$$

Using negation we can obtain disjunction \vee , universal quantification $\forall x$ and $\forall X$, and implication \rightarrow . MSO-formulas are evaluated on σ -structures, where element and set variables range respectively over elements and subsets of the domain. Weak monadic second-order logic (WMSO) has the same syntax as MSO, but second-order variables are interpreted as finite subsets of the underlying universe.

WMSO+B is the extension of WMSO by the bounding quantifier $BX\varphi$ for $X \in V_s$. The semantics of $BX\varphi$ on a structure \mathcal{A} with universe A is defined as follows: $\mathcal{A} \models BX\varphi(X)$ if and only if there is a bound $b \in \mathbb{N}$ such that whenever $\mathcal{A} \models \varphi(B)$ for some finite subset $B \subseteq A$, then $|B| \leq b$.

Finally, let BMWB denote the set of all Boolean combinations of MSO-formulas and (WMSO+B)-formulas.

Example 2. Given a graph $\mathcal{G} = (V, E)$, WMSO can express reachability in \mathcal{G} . We define the WMSO-formula $\text{reach}(x_1, x_2)$ to be

$$\exists Z x_1 \in Z \wedge \forall Y \subseteq Z [(x_1 \in Y \wedge \text{scl}(Y)) \rightarrow x_2 \in Y],$$

where $\text{scl}(Y) = \forall y \forall z (y \in Y \wedge z \in Z \wedge E(y, z)) \rightarrow z \in Y$ says that the set Y is successor-closed. The semantics of reach seen as an MSO-formula or a WMSO-formula are the same because b is reachable from a in the graph \mathcal{G} if and only if it is in some finite subgraph of \mathcal{G} .

In [3] Bojańczyk and Toruńczyk show that satisfiability for WMSO+B over binary trees is decidable. This result can be extended to BMWB over trees of branching degree n (n -trees):

Theorem 3 (cf. [3, 7]). *One can decide whether for a given formula $\varphi \in \text{BMWb}$ there exists an n -tree \mathcal{T}_n such that $\mathcal{T}_n \models \varphi$.*

3 The Description Logic $\mathcal{ALCC}^P(\mathcal{D})$

We introduce now the new DL $\mathcal{ALCC}^P(\mathcal{D})$ and some basic notions on DLs in general. We start with the (concrete domain) constraints.

Let us fix for the rest of this section a countably infinite set of register variables Reg , a relational signature σ , and an arbitrary σ -structure $\mathcal{D} = (D, R_1, R_2, \dots)$, called the concrete domain.

Definition 4. We define a constraint $c(x_1, \dots, x_k)$ of arity k over \mathcal{D} as a Boolean combination of atomic constraints $R(x_{i_1}, \dots, x_{i_{\text{ar}(R)}})$, where $R \in \sigma$ and $i_j \in \{1, \dots, k\}$. We write $\mathcal{D} \models c(a_1, \dots, a_k)$ if the constraint is satisfied in \mathcal{D} by the assignment $x_i \mapsto a_i$.

Example 5. Consider as concrete domain $\mathcal{Z} = (\mathbb{Z}, <, =)$, the relational structure introduced in Example 1. Using infix notation for the relations, $c(x, y, z) = [(x < y \vee x = y) \wedge \neg y < z]$ is a constraint of arity 3 over \mathcal{Z} , and $\mathcal{Z} \models c(0, 1, 0)$.

Let us fix two countably infinite sets N_C and N_R of concept names and role names respectively. Let then $N_F \subseteq N_R$ be the set of features, i.e. roles that are interpreted as partial functions. We call a finite sequence $P = r_1 \dots r_n$ of role names a role-path of length n .

Definition 6. We recursively define $\mathcal{ALCC}^P(\mathcal{D})$ -concepts as follows

$$C := A \mid \neg C \mid (C \sqcap C) \mid \exists r.C \mid \exists P.c(S^{i_1}x_1, \dots, S^{i_k}x_k)$$

where $A \in N_C$, $r \in N_R$, P is a role-path of length $n \geq 0$, c is a constraint of arity k , $x_1, \dots, x_k \in \text{Reg}$, and $i_1, \dots, i_k \leq n$. We call $\exists P.c(S^{i_1}x_1, \dots, S^{i_k}x_k)$ a path constraint. The symbol S appearing in the path constraints stands for successor, as the term $S^{i_k}x$ points at the register variable x in the i -th position of the path P .

\mathcal{ALC} is the fragment of $\mathcal{ALCC}^P(\mathcal{D})$ without path constraints. As usual, a general concept inclusion (GCI) is an expression of the form $C \sqsubseteq D$, where C and D are concepts. A TBox is a finite set of GCIs.

Definition 7. A \mathcal{D} -interpretation \mathcal{I} is a tuple $(\Delta, \cdot^{\mathcal{I}}, \gamma)$, where Δ is a set called *the domain*, $\cdot^{\mathcal{I}}$ is the *interpretation function*, and $\gamma : \Delta \times \text{Reg} \rightarrow D$ is the *valuation function*, assigning a value from the concrete domain to each register variable in each element of the interpretation domain. The interpretation function maps each concept name $A \in \mathbf{N}_C$ to some $A^{\mathcal{I}} \subseteq \Delta$, each role name $r \in \mathbf{N}_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta \times \Delta$, with the condition that if $r = f \in \mathbf{N}_F$ the binary relation $f^{\mathcal{I}}$ has to be functional, i.e. for all $a, b, c \in \Delta$, $(a, b), (a, c) \in f^{\mathcal{I}}$ implies $b = c$. It is then extended to $\neg C, C \sqcap D, \exists r.D$ as usual, and to a role-path $P = (r_1, \dots, r_n)$ as:

$$P^{\mathcal{I}} := \{(v_0, \dots, v_n) \in \Delta^{n+1} \mid (v_{i-1}, v_i) \in r_i^{\mathcal{I}} \text{ for } i = 1, \dots, n\}.$$

Finally, if P has length n , we define $(\exists P.c(S^{i_1}x_1, \dots, S^{i_k}x_k))^{\mathcal{I}}$ as

$$\{v \in \Delta \mid \exists (v_0, \dots, v_n) \in P^{\mathcal{I}} \text{ s.t. } v_0 = v, \\ \text{and } \mathcal{D} \models c(\gamma(v_{i_1}, x_1), \dots, \gamma(v_{i_k}, x_k))\}.$$

So the fact that an element $v \in \Delta$ belongs to the interpretation of a path constraint $\exists P.c(S^{i_1}x_1, \dots, S^{i_k}x_k)$ means that there exists an instance of the path $P^{\mathcal{I}}$ starting in v , namely some $(v_0, v_1, \dots, v_n) \in P^{\mathcal{I}}$ with $v_0 = v$, such that the assignment $y_j \mapsto \gamma(v_{j_i}, x_j)$ satisfies the constraint $c(y_1, \dots, y_k)$. A term S^i inside the constraint is used to point at the i -th element of the path $P^{\mathcal{I}}$. Note that the requirement that $i_1, \dots, i_k \leq n$ ensures that such element is well-defined.

Note, also that an atomic constraint $R(S^{i_1}x_1, \dots, S^{i_k}x_k)$ is *local* in the sense that it involves only nodes in a fixed *neighborhood* of the position at which they are evaluated. We call $d := \max\{i_1, \dots, i_k\}$ the *depth* of R . By extension, the depth of a constraint c is the maximum depth of all the atomic constraints which appear in c .

Let $\text{Reg}_{C, \mathcal{T}}$ denote the set of register variables that occur in C and \mathcal{T} . Obviously, the relevance of the valuation function γ is limited to the domain $(\Delta \times \text{Reg}_{C, \mathcal{T}})$.

Definition 8. A \mathcal{D} -interpretation \mathcal{I} is a *model* of a TBox \mathcal{T} ($\mathcal{I} \models \mathcal{T}$) if and only if every GCI $C \sqsubseteq D \in \mathcal{T}$ is satisfied, that is, if and only if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Given a concept C and a TBox \mathcal{T} , we say C is *satisfiable with respect to \mathcal{T}* if and only if there exists a model \mathcal{I} of \mathcal{T} such that $C^{\mathcal{I}} \neq \emptyset$. We write $\mathcal{I} \models_{\mathcal{T}} C$.

We define some usual abbreviations: $C \sqcup D := \neg(\neg C \sqcap \neg D)$, $\forall r.C := \neg \exists r. \neg C$, $\forall P.c := \neg \exists P. \neg c$, $\exists P.C := \exists r_1. \exists r_2. \dots \exists r_n. C$, where $P = r_1 \dots r_n$, and special concept $\top := A \sqcup \neg A$

Using this extended set of operators and DeMorgan's laws we can, given an $\mathcal{ALC}^P(\mathcal{D})$ -concept C , obtain an equivalent concept in negation normal form $\text{nnf}(C)$, where negation only appears before concept names or atomic constraints.

The *TBox-concept* of a TBox \mathcal{T} is $C_{\mathcal{T}} := \bigcap_{C \sqsubseteq D \in \mathcal{T}} (\neg C \sqcup D)$. Note that it is equivalent to ask that an interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}}, \gamma_{\mathcal{I}})$ satisfies all GCIs from \mathcal{T} and to ask that the $C_{\mathcal{T}}$ is globally satisfied, i.e. $(C_{\mathcal{T}})^{\mathcal{I}} = \Delta$. Vice-versa, any globally satisfied concept C can be seen as the GCI $\top \sqsubseteq C$. For technical reasons, it is convenient for us to adopt this view, and from now on we will always assume that a TBox consists of a single concept $C_{\mathcal{T}}$ that needs to be globally satisfied. We say a TBox \mathcal{T} is in negation normal form if so is $C_{\mathcal{T}}$.

Example 9. Take again $\mathcal{Z} = (\mathbb{Z}, <, =)$ as concrete domain and consider the following TBox: $\mathcal{T} = \{\exists \text{neighbor}.(\text{green grass} < \text{Sgreen grass}), \neg \text{GreenThumb} \sqcup (\text{alive plants} = \text{plants})\}$ ². Here we consider three register variables: *green grass* measures the degree

² Here the absence of a path quantifier before $(\text{alive plants} = \text{plants})$ means that we are referring to a 'path of length zero'.

of 'greenness' of an individual's lawn, while *plants* and *alive plants* count the number of plants (total or alive) of an individual. In any model of \mathcal{T} , every individual has a neighbor whose grass is greener, and individuals with a green thumb keep all their plants alive.

In Example 9, there cannot exist a model for \mathcal{T} with a finite underlying domain, as the degree of greenness of neighboring lawns is strictly increasing. This is never the case for ordinary \mathcal{ALC} , which enjoys the finite model property.

In the literature on description logics with concrete domains (for instance in [1, 16]) one finds constraints of the kind $\exists R(P_1x_1, \dots, P_kx_k)$, where R is a relation from the concrete domain and each P_i is a path composed of features only. The constraint is satisfied by an element d if there exist k elements, $d_1 \dots d_k$, reachable from d via the feature-paths $P_1 \dots P_k$, such that the tuple $(\gamma(d_1, x_1), \dots, \gamma(d_k, x_k))$ belongs to the relation R in the concrete domain. Nonetheless, in many interesting cases this kind of constraint can be replaced with path constraints by introducing some additional register variables. For example $\exists(P_1x_1 < P_2x_2)$ can be expressed as $\exists P_1.(S^{|P_2|}x_1 < z) \sqcap \exists P_2.(z \leq S^{|P_2|}x_2)$, where z is a fresh register variable. Also $\forall(P_1x_1 < P_2x_2)$ can be replaced by $\neg(\exists P_1. \top \sqcap \exists P_2. \top) \sqcup (\exists P_1.(S^{|P_1|}x_1 < z) \sqcap \exists P_2.(z \leq S^{|P_2|}x_2))$.³

On the other hand, our constraints can use role-paths of arbitrary length, which—to the best of our knowledge—is not allowed in the previously existing literature, where they are limited in length or disallowed completely in favor of feature-paths. Therefore, although generally incomparable in expressiveness, path constraints are strictly more expressive on interesting concrete domains.

Note also that for each individual v of the abstract domain, the value $\gamma(v, x)$ is defined for all $x \in \text{Reg}$. This is essentially the same as saying that each $\gamma(\cdot, x)$, in literature commonly called *concrete feature*, is interpreted as a *total* function, more in the style of the *attributes* used by Toman and Weddell (see [19]).

3.1 $\mathcal{ALC}^P(\mathcal{D})$ has the Tree Model Property

Definition 10. Let $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}}, \gamma)$ be a \mathcal{D} -interpretation and define $\rightarrow := \bigcup_{r \in \mathbf{N}_R} r^{\mathcal{I}}$. We say \mathcal{I} is a *tree-shaped \mathcal{D} -interpretation* if and only if (Δ, \rightarrow) is a tree, that is:

- $\Delta \subseteq \Sigma^*$ is (isomorphic to) a prefix-closed set of strings over some alphabet Σ , and
- for all $u, v \in \Delta$, $u \rightarrow v$ if and only if $v = ua$ for some $a \in \Sigma$.

We call \mathcal{I} an *n -tree \mathcal{D} -interpretation* if $\Delta = [1, n]^*$ for some $n \in \mathbb{N}$, where $[1, n]$ denotes the closed interval $\{1, \dots, n\}$.

A logic has the *tree model property*, if for every concept C and every TBox \mathcal{T} , C is satisfiable w.r.t. \mathcal{T} iff there exists a tree-shaped \mathcal{D} -interpretation \mathcal{J} such that $\mathcal{J} \models_{\mathcal{T}} C$.

We show that $\mathcal{ALC}^P(\mathcal{D})$ has a strong version of the tree model property, in which we are able to give a bound on the branching degree. Let \mathcal{T} and C be an $\mathcal{ALC}^P(\mathcal{D})$ TBox and concept, respectively. We denote by $\text{Sub}(\mathcal{T}, C)$ the set of all concepts which appear in \mathcal{T} and C . If d is the maximum depth of an existential path constraint occurring in \mathcal{T} and C , and e is the number of existentially quantified subconcepts in $\text{Sub}(\mathcal{T}, C)$ we can prove the following:

Lemma 11. C is satisfiable w.r.t. \mathcal{T} iff there exists an n -tree \mathcal{D} -interpretation \mathcal{I} such that $\mathcal{I} \models_{\mathcal{T}} C$, where $n = d \cdot e$.

³ Such translations must be applied after the concepts are converted to strong negation normal form (see Sec. 3.2) because they preserve satisfiability but are not necessarily closed under negation.

Sketch of proof. First we show the normal tree model property, obtained by unraveling. We need to deal with the constraints, but this does not create particular problems. Successively we *prune* the tree, going top bottom and leaving only those nodes that are necessary to satisfy the existentially quantified formulas. These are at most e in each node, and are witnessed at most by a path of d nodes. Once the pruning is finished we have a tree-shaped interpretation \mathcal{I} with branching degree at most n . We can obtain an n -tree: for each node we introduce the needed number of s -successors, where s is a role that does not appear in C or \mathcal{T} , and attach a copy of \mathcal{I} to it until we obtain an n -tree. This guarantees that the freshly introduced nodes respect the TBox-concept. \square

Given this result, from now on we can restrict ourselves to \mathcal{D} -interpretations of the form $\mathcal{I} = ([1, n]^*, \cdot^{\mathcal{I}}, \gamma_{\mathcal{I}})$ where for each $u, v \in [1, n]^*$ there exists $r \in \mathbb{N}_R$ such that $(u, v) \in r^{\mathcal{I}}$ if and only if there exists $i \in [1, n]$ such that $v = ui$.

3.2 Strong Negation Normal Form

We show now how, requiring that the concrete domain satisfies a property called *negation closure*, we can obtain a *strong negation normal form*, where negation only appears in front of concept names.

Definition 12. We call a σ -structure $\mathcal{D} = (D, R_1^{\mathcal{D}}, R_2^{\mathcal{D}}, \dots)$ *negation-closed* if for every $R \in \sigma$ the complement of $R^{\mathcal{D}}$ is effectively definable by a positive existential first-order formula, i.e., if there is a computable function that maps each relation symbol $R \in \sigma$ to a positive existential first-order formula $\varphi_R(x_1, \dots, x_{\text{ar}(R)})$ (a formula that is built up from relations of σ using \wedge, \vee , and \exists) such that

$$D^{\text{ar}(R)} \setminus R^{\mathcal{D}} = \{(a_1, \dots, a_{\text{ar}(R)}) \mid \mathcal{D} \models \varphi_R(a_1, \dots, a_{\text{ar}(R)})\}.$$

Example 13. Let $=_a := \{a\}$ be the unary predicate which holds only for a , and $\equiv_{a,b} := \{a + kb \mid k \in \mathbb{Z}\}$ be a unary predicate expressing that some number is congruent to a modulo b . Consider the structure $(\mathbb{Z}, <, =, (=_{a,b})_{a \in \mathbb{Z}}, (\equiv_{a,b})_{0 \leq a < b})$. Such a structure is negation-closed, we have in fact:

- $\neg x = y$ if and only if $x < y \vee y < x$,⁴
- $\neg x < y$ if and only if $x = y \vee y < x$,
- $\neg x = a$ if and only if $\exists y (y = a \wedge (x < y \vee y < x))$, and
- $\neg x \equiv a \pmod{b}$ if and only if $x \equiv c \pmod{b}$ for some $0 \leq c < b$ with $a \neq c$:

$$\bigvee_{\substack{0 \leq c < b \\ a \neq c}} x \equiv c \pmod{b}.$$

Definition 14. We say that an $\mathcal{ALCP}(\mathcal{D})$ -concept φ is in *strong negation normal form* if it is in negation normal form and if, additionally, all constraints $c(x_1, \dots, x_k)$ do not contain any negation. Consequently we say that a TBox \mathcal{T} is in strong negation normal form if so is the TBox-concept $C_{\mathcal{T}}$.

Lemma 15. If $\mathcal{D} = (D, R_1^{\mathcal{D}}, R_2^{\mathcal{D}}, \dots)$ is negation-closed, given a concept C and a TBox \mathcal{T} , one can compute \widehat{C} and $\widehat{\mathcal{T}}$ in strong negation normal form such that C is satisfiable with respect to \mathcal{T} if and only if \widehat{C} is satisfiable with respect to $\widehat{\mathcal{T}}$.

Sketch of proof. The idea is the following: Given a negated atomic constraint $\neg R(\dots)$ of depth d , we want to substitute it with a boolean combination of positive ones. We use the fact that the complement of R is definable by an existential first order sentence φ_R .

⁴ We write $x = y$ instead of $=(x, y)$, $x = a$ instead of $=_a x$, and so on.

We deal with the variables existentially quantified in φ_R by adding fresh register variables. These new registers are ‘placed’ at depth d , so that (considering the tree-like structure of an $\mathcal{ALCP}(\mathcal{D})$ model) they are unique for the path used to evaluate R . \square

Example 16. Consider the concrete domain $(\mathbb{Z}, <, =, (=_{a,b})_{a \in \mathbb{Z}})$ and the concept $C = \exists r s. [S^1 x < S^2 x \wedge \neg S^2 x = 3]$. An individual v which belongs to an interpretation of C must necessarily have an r -successor v_1 which has an s -successor v_2 , such that the value of x in v_1 is smaller than the value of x in v_2 , which in turn must be different than 3. As one can see from Example 13, \mathcal{D} is negation-closed, and we can find an existentially quantified positive first order formula, namely $\psi(a) = \exists z (z = 3 \wedge (a < z \vee z < a))$, such that $\neg x = 3$ if and only if $\psi(x)$ holds. The strong negation normal form of C is $\widehat{C} = \exists r s. [S^1 x < S^2 x \wedge S^2 y = 3 \wedge (S^2 x < S^2 y \vee S^2 y < S^2 x)]$. As you can see we have introduced a new register variable y and placed it at depth 2 inside the constraint to hold the value that was existentially quantified in ψ .

Now that we have successfully eliminated negation from inside the constraints, there is one last step to do, in order to obtain a normal form that will be useful in the next section. Observe that if a constraint $c(x_1, \dots, x_k)$ does not contain negation, it is possible to apply distributivity repeatedly and obtain an equivalent constraint in DNF or in CNF⁵ which still does not contain negation. Therefore we can assume that all path constraints of the form $\exists P.c$ (respectively $\forall P.c$) are such that the constraint c is in DNF (resp. CNF). Using then the fact that universal quantification commutes with conjunction and that existential quantification commutes with disjunction, we can easily prove the following facts:

$$\begin{aligned} \exists P. \bigvee_{i=1}^n (a_1^i \wedge \dots \wedge a_{m_i}^i) &\equiv \bigwedge_{i=1}^n \exists P. (a_1^i \wedge \dots \wedge a_{m_i}^i), \text{ and} \\ \forall P. \bigwedge_{i=1}^n (a_1^i \vee \dots \vee a_{n_i}^i) &\equiv \bigwedge_{i=1}^n \forall P. (a_1^i \vee \dots \vee a_{n_i}^i), \end{aligned}$$

where each a_j^i is an atomic constraint. Therefore, given a concept C in strong negation normal form, and applying the above described transformations, we can obtain a new concept C' which is still in strong negation normal form, and is such that all path constraints are of the kind $\exists P.c$ (or $\forall P.c$) where c is a conjunction (resp. disjunction) of atomic constraints. We call this the *constraint normal form* of C .

4 The EHD Method

Following the approach in [6, 7] for CTL* and ECTL*, we can reduce the satisfiability problem of $\mathcal{ALCP}(\mathcal{D})$ to the satisfiability problem for BMWB over n -trees, provided the concrete domain has right properties.

4.1 The EHD-Property

The central notion used in the decidability proof is the EHD-property. EHD stands for ‘the existence of a homomorphism is definable’. It is a property of a relational structure \mathcal{A} , expressing the ability of a logic L to distinguish between those structures \mathcal{B} which can be mapped to \mathcal{A} by a homomorphism ($\mathcal{B} \preceq \mathcal{A}$) and those that cannot.

⁵ Disjunctive or conjunctive normal form.

Definition 17. Let L be a logic. A σ -structure \mathcal{A} has the EHD(L)-property, if there is a computable function that maps every finite subsignature $\tau \subseteq \sigma$ to an L -sentence φ_τ s.t. for every countable τ -structure \mathcal{B} holds: $\mathcal{B} \preceq \mathcal{A} \Leftrightarrow \mathcal{B} \models \varphi_\tau$.

We will see that for a negation-closed domain \mathcal{D} with the EHD(BMWB)-property, satisfiability of $\mathcal{ALCC}^P(\mathcal{D})$ is decidable. For this reason, we are mainly interested in structures with EHD(L)-property, where L is BMWB or some fragment of this logic. In this case, we might omit L and simply write EHD.

Remark 18. In [6, 7, 5] several (classes of) relational structures are investigated. Some of them that enjoy the property EHD are:

- the integers with equality-, order-, constants-, and modulo-constraints: $(\mathbb{Z}, =, <, (=_{a \in \mathbb{Z}}, (=_{a,b})_{a < b})$,
- the natural numbers with the same relational signature: $(\mathbb{N}, =, <, (=_{a \in \mathbb{N}}, (=_{a,b})_{0 \leq a < b})$,
- the class of all semi-linear orders (see [5]),
- the class of all trees of height h for some fixed $h \in \mathbb{N}$,
- $(\mathbb{Z}^n, <_{\text{lex}}, =)$ where $<_{\text{lex}}$ is the lexicographic order,
- $\text{Allen}_{\mathbb{Z}}$: the set of intervals over the integers together with Allen's relations, which allow to describe their relative positioning.

It was shown in [6] that $(\mathbb{Z}, <)$ has the EHD-property. Consider any countable $\{<\}$ -structure $\mathcal{A} = (A, <)$. For $x, y \in A$ we write $x <^* y$ if there exist x_1, \dots, x_n s.t. $x < x_1 < \dots < x_n < y$ in \mathcal{A} and call $\{x, x_1, \dots, x_n, y\}$ a $<$ -path from x to y . It is proved that $\mathcal{A} \preceq (\mathbb{Z}, <)$ iff

- \mathcal{A} is *acyclic*: there are no two elements x, y s.t. $x <^* y < x$, and
- for every two elements $x, y \in A$, there exists a bound n such that all $<$ -path from x to y have at most n elements.

This can be expressed by the following BMWB-formulas: $\neg \exists x, y (\text{reach}_{<}(x, y) \wedge y < x)$ and $\forall x \forall y \text{BXPath}(X, x, y)$. Here $\text{reach}_{<}$ is the same as in Example 2, but with the edge relation E replaced by $<$, and $\text{Path}(X, x, y)$ is a formula indicating that the set X is a $<$ -path from x to y (see [7, Ex. 2]). Here the bounding quantifier limits the length of all paths between any two elements.

In [12, 13] concrete domains over the rationals are considered for the logics \mathbb{Q} -SHIQ and TDL. The difference between TDL and $\mathcal{ALCC}^P(\mathcal{D})$ is that the TDL only allows feature-paths as connectors to the concrete domain. For \mathbb{Q} -SHIQ and TDL, it is stated that adding a unary predicate int , expressing that a certain concrete value has to be an integer, would be extremely useful. Decidability of reasoning in these logics under this addition remained an open problem. We show that the domain $\mathcal{Q} = (\mathbb{Q}, <, \text{int}, \overline{\text{int}})$, where $\text{int} = \mathbb{Z}$ and $\overline{\text{int}} = \mathbb{Q} \setminus \mathbb{Z}$, has the EHD-property. In [7, Lem. 38] it is shown that, if a domain \mathcal{D} has the EHD-property, then so does $\mathcal{D}^=$, obtained by adding equality. This proves that $\mathcal{Q}^= = (\mathbb{Q}, =, <, \text{int}, \overline{\text{int}})$ (which is also negation-closed) has the EHD-property.

Proposition 19. $\mathcal{Q} = (\mathbb{Q}, <, \text{int}, \overline{\text{int}})$ has the EHD-property.

Sketch of proof. Let $\mathcal{A} = (A, <, \text{int}^A, \overline{\text{int}}^A)$ be an arbitrary countable structure. We prove that \mathcal{A} allows a homomorphism to \mathcal{Q} iff

- H1 \mathcal{A} is acyclic,
- H2 there exists no x s.t. $x \in \text{int}^A \cap \overline{\text{int}}^A$,
- H3 given any two elements $x, y \in A$, there exists a bound n s.t. each $<$ -path from x to y contains at most n elements from int^A .

In this setting, it is only the number of elements of int^A that needs to be bounded on all paths between two elements. The reason is that, being \mathbb{Q} dense, we can accommodate any countable amount of numbers

in any interval, provided that they are not forced to be integers. Properties H1-H3 are easily defined in BMWB: acyclicity is expressed as above using $\text{reach}_{<}$, H2 is given by $\neg \exists x (\text{int}(x) \wedge \overline{\text{int}}(x))$ and H3 by $\forall x, y \text{BX}[X \subseteq \text{int}^A \wedge \exists Z (X \subseteq Z \wedge \text{Path}(Z, x, y))]$. \square

4.2 Satisfiability of $\mathcal{ALCC}^P(\mathcal{D})$

We are now ready to state our main result:

Theorem 20. *If a concrete domain \mathcal{D} is negation-closed and has the property EHD(BMWB), the satisfiability problem for $\mathcal{ALCC}^P(\mathcal{D})$ is decidable.*

This theorem classifies all the concrete domains listed in Remark 18 and the new one from Proposition 19 positively, yielding a good number of decidability results for $\mathcal{ALCC}^P(\mathcal{D})$ w.r.t. general TBoxes, which strictly improves what was known so far.

The idea behind the proof of this theorem is to separate the search of a \mathcal{D} -interpretation for a concept C w.r.t. a TBox \mathcal{T} into two parts: In a first step look for an ordinary \mathcal{ALCC} interpretation (i.e., without the valuation function) that is a model for an *abstracted* version of C and \mathcal{T} . That is, replace each atomic constraint appearing in C and \mathcal{T} with a fresh concept name B and obtain a classical \mathcal{ALCC} -concept C_a and TBox \mathcal{T}_a , where the a stands for 'abstracted'. The fact that C_a is satisfiable w.r.t. \mathcal{T}_a is clearly not enough to guarantee that C is satisfiable w.r.t. \mathcal{T} . For instance, the $\mathcal{ALCC}^P(\mathcal{D})$ -concept $\exists r.(x < Sx \wedge Sx < x)$ is unsatisfiable, while its abstraction $\exists r.(B_1 \sqcap B_2)$ is not. To avoid this effect, the second step creates from the model of the abstracted concept a so-called *constraint graph*, a structure for storing the information from the constraints that were abstracted away. It turns out that if such constraint graph allows a homomorphism to our concrete domain, then this guarantees that the constraints are satisfied.

For the rest of this section let us fix a signature σ , a negation-closed σ -structure \mathcal{D} as concrete domain with the EHD-property, and an $\mathcal{ALCC}^P(\mathcal{D})$ -concept C and TBox \mathcal{T} , both in constraint normal form, in which only the atomic constraints $\theta_1, \dots, \theta_n$ occur. Let d_i be the depth of each θ_i , and let $B_1, \dots, B_n \in \text{Nc} \setminus \text{Sub}(\mathcal{T}, C)$.

Definition 21. Let $P = r_1 \dots r_p$ and c be a conjunction of the atomic constraints $\theta_1, \dots, \theta_m$ with $m \leq n$ with depths s.t. $0 =: d_0 \leq d_1 \leq \dots \leq d_m \leq d_{m+1} := p$ (if this is not the case, it suffices to reorder the constraints). Define the *abstraction of an existential path constraint* $E = \exists P.c(S^{i_1} x_1, \dots, S^{i_k} x_k)$, as

$$E_a = \exists P_1.(B_1 \sqcap \exists P_2.(B_2 \sqcap \dots \exists P_m.(B_m \sqcap \exists P_{m+1}.\top) \dots)), \quad (1)$$

where $\exists P_i$ is short for $\exists r_{d_{i-1}+1} \dots \exists r_{d_i}$. If $d_i = d_{i+1}$, then $\exists P_{i+1}$ is empty. Let c' be a disjunction of atomic constraints containing $\theta_1, \dots, \theta_m$ with $0 =: d_0 \leq d_1 \leq \dots \leq d_m \leq d_{m+1} := p$. Define the *abstraction of a universal path constraint* $E = \forall P.c'(S^{i_1} x_1, \dots, S^{i_k} x_k)$ as

$$E_a = \forall P_1.(B_1 \sqcup \forall P_2.(B_2 \sqcup \dots \forall P_m.(B_m \sqcup \forall P_{m+1}.\perp) \dots)). \quad (2)$$

We define C_a and \mathcal{T}_a as the \mathcal{ALCC} -concept and TBox obtained by C and \mathcal{T} by replacing every occurrence of a path constraint E by its abstraction E_a .

Let us consider $C = \exists r_1 r_2 r_3 (x = y \wedge x < S^2 x \wedge S^1 y = S^2 x)$, its abstraction C_a is $B_1 \sqcap \exists r_1. \exists r_2. (B_2 \sqcap B_3 \sqcap \exists r_3. \top)$, where the new concept names are assigned to the atomic constraints in order of appearance. Notice how we use the locality of constraints from $\mathcal{ALCC}^P(\mathcal{D})$ to individuate the 'lower' node involved in the constraint

(the one at depth d_i) and mark it as belonging to the fresh concept B_i . This way, when considering a tree-model of the abstracted concept C_a w.r.t. \mathcal{T}_a , all paths of length d_i that end in a node marked with B_i should satisfy the constraint θ_i .

Definition 22. Given an n -tree \mathcal{D} -interpretation $\mathcal{I} = ([1, n]^*, \cdot^{\mathcal{I}}, \gamma_{\mathcal{I}})$ s.t. $B_1^{\mathcal{I}} = \dots = B_m^{\mathcal{I}} = \emptyset$, we define the *abstraction of \mathcal{I}* as the interpretation $\mathcal{I}_a = ([1, n]^*, \cdot^{\mathcal{I}_a})$, where $\cdot^{\mathcal{I}_a}$ is defined as

- $A^{\mathcal{I}_a} = A^{\mathcal{I}}$ for all $A \in (\text{Nc} \setminus \{B_1, \dots, B_m\})$,
- $r^{\mathcal{I}_a} = r^{\mathcal{I}}$ for all $r \in \text{Nr}$,
- if $\theta_j = R(S^{i_1}x_1, \dots, S^{i_k}x_k)$ has depth d_j , then $u \in B_j^{\mathcal{I}_a}$ iff
 - $u = wv$ for some $w, v \in [1, n]^*$ with $|v| = d_j$, and
 - $(\gamma(wv_1, x_1), \dots, \gamma(wv_k, x_k)) \in R^{\mathcal{D}}$,

where v_t denotes the prefix of v of length i_t .

Hence, the fact that an element wv with $|v| = d_j$ belongs to the interpretation of B_j means that the atomic constraint θ_j is satisfied along every path that starts in node w and descends in the tree via wv . Now let $\text{Reg}_{C, \mathcal{T}}$ be the set of register variables occurring in C and \mathcal{T} .

Definition 23. Take an n -tree interpretation $\mathcal{J} = ([1, n]^*, \cdot^{\mathcal{J}})$ where $B_1^{\mathcal{J}}, \dots, B_m^{\mathcal{J}}$ can be non-empty. We define the *constraint graph $\mathcal{G}_{\mathcal{J}}$* of \mathcal{J} as a countable σ -structure $\mathcal{G}_{\mathcal{J}} = (([1, n]^* \times \text{Reg}_{C, \mathcal{T}}), R_1^{\mathcal{G}}, R_2^{\mathcal{G}}, \dots)$ as follows: The interpretation $R^{\mathcal{G}}$ of the relation $R \in \sigma$ contains all k -tuples $((wv_1, x_1), \dots, (wv_k, x_k))$, where $k = \text{ar}(R)$, for which there are $1 \leq j \leq m$ and $v \in [1, n]^{d_j}$ s.t. $wv \in B_j^{\mathcal{J}}$, and $\theta_j = R(S^{i_1}x_1, \dots, S^{i_k}x_k)$, where v_t still denotes the prefix of v of length i_t .

The domain of $\mathcal{G}_{\mathcal{J}}$ has one element for each pair (v, x) where v is a member of the domain of \mathcal{J} and x is a register variable appearing in C . When we abstract an atomic constraint θ_i we replace it with its *placeholder* B_i , but any occurrence of B_i marks a path where θ_i needs to hold. Such information is stored in the relations of $R^{\mathcal{G}}$.

Example 24. Let \mathcal{D} be a concrete domain having $<$ and $=$ in its signature. Suppose the concept names B_1 and B_2 are used to replace the atomic constraints $\theta_1 = (x = y)$ and $\theta_2 = (x < Sx)$ of depth $d_1 = 0$ and $d_2 = 1$, respectively. Figure 1 depicts the constraint graph associated with an ordinary 2-tree interpretation \mathcal{J} .

In the next theorem, we illustrate the connection between the satisfiability of an $\mathcal{ALC}^P(\mathcal{D})$ -concept w.r.t. a TBox, and the satisfiability of its abstraction. We denote by $\#_E(\mathcal{T}, C)$ the number of existentially quantified subconcepts that occur in $\text{Sub}(\mathcal{T}, C)$. Let C be an $\mathcal{ALC}^P(\mathcal{D})$ -concept and \mathcal{T} a TBox—both in constraint normal form—and let $n = d \cdot \#_E(C, \mathcal{T})$ where d is the maximum depth of all constraints appearing in $\text{Sub}(\mathcal{T}, C)$. Then the following holds:

Theorem 25. C is satisfiable w.r.t. \mathcal{T} iff there exists an ordinary n -tree interpretation $\mathcal{I} = ([1, n]^*, \cdot^{\mathcal{I}})$ s.t. $\mathcal{I} \models_{\mathcal{T}_a} C_a$ and s.t. $\mathcal{G}_{\mathcal{I}} \preceq \mathcal{D}$.

Proof. Let $\theta_1, \dots, \theta_m, d_1, \dots, d_m$ and $\text{Reg}_{C, \mathcal{T}}$ be as before.

(\Rightarrow) W.l.o.g. assume that $\mathcal{I} = ([1, n]^*, \cdot^{\mathcal{I}}, \gamma_{\mathcal{I}})$ is an n -tree \mathcal{D} -interpretation s.t. $\mathcal{I} \models_{\mathcal{T}} C$. Our first claim is that $\mathcal{I}_a \models_{\mathcal{T}_a} C_a$, which we show by induction on the structure of C , proving that for all $v \in [1, n]^*$ and for all subconcepts $E \in \text{Sub}(\mathcal{T}, C)$, $u \in E^{\mathcal{I}}$ implies $u \in E_a^{\mathcal{I}_a}$. Due to space limitations, we show only some cases. The remaining ones are shown in [8].

- If $E \in \text{Sub}(\mathcal{T}, C)$ is a concept name, then $E_a = E$.
- If $E = F \sqcap G$, then $u \in E^{\mathcal{I}}$ implies $u \in F^{\mathcal{I}}$ and $u \in G^{\mathcal{I}}$. By induction hypothesis we have that $u \in F_a^{\mathcal{I}_a}$ and $u \in G_a^{\mathcal{I}_a}$ which yields $u \in (F_a \sqcap G_a)^{\mathcal{I}_a} = E_a^{\mathcal{I}_a}$.

- If $E = \exists r.F$ and $u \in E^{\mathcal{I}}$, then there exists an element $v \in [1, n]^*$, s.t. $(u, v) \in r^{\mathcal{I}}$ and $v \in F^{\mathcal{I}}$. Then $(u, v) \in r^{\mathcal{I}_a}$ by definition of \mathcal{I}_a and $v \in F_a^{\mathcal{I}_a}$ by induction hypothesis. Together we obtain $u \in (\exists r.F_a)^{\mathcal{I}_a} = E_a^{\mathcal{I}_a}$.
- Let $E = \exists P.c(S^{i_1}x_1, \dots, S^{i_t}x_t)$ with $P = r_1 \dots r_p$. Since C and \mathcal{T} are in constraint normal form, we can assume that (eventually renaming the atomic constraints) $c = \theta_1 \wedge \dots \wedge \theta_n$ where the depths d_1, \dots, d_n satisfy that $0 =: d_0 \leq d_1 \leq \dots \leq d_n \leq d_{n+1} := p$. Since $u \in E^{\mathcal{I}}$, we know that there exists a tuple $(u_0, \dots, u_p) \in P^{\mathcal{I}}$ s.t. $u_0 = u$ and that $\mathcal{D} \models c(\gamma(u_{i_1}, x_1), \dots, \gamma(u_{i_t}, x_t))$. If we have $\theta_i = R(S^{j_1}y_1, \dots, S^{j_k}y_k)$, this means that $(\gamma(u_{j_1}, y_1), \dots, \gamma(u_{j_k}, y_k)) \in R^{\mathcal{D}}$. By definition of \mathcal{I}_a , this implies that $u_{d_i} \in B_i^{\mathcal{I}_a}$. Now, since $(a, b) \in r^{\mathcal{I}}$ implies $(a, b) \in r^{\mathcal{I}_a}$, then $(u_0, \dots, u_p) \in P^{\mathcal{I}_a}$ as well. This, together with the fact that $u_{d_i} \in B_i^{\mathcal{I}_a}$ for $i = 1, \dots, n$, implies that $u \in (\exists P_1.(B_1 \sqcap \exists P_2.(B_2 \sqcap \dots \exists P_n.(B_n \sqcap \exists P_{n+1}.\top) \dots))^{\mathcal{I}_a}$, where $\exists P_i$ is short for $\exists r_{d_{i-1}+1} \dots \exists r_{d_i}$. This shows the claim. The second claim is that $\mathcal{G}_{\mathcal{I}_a} \preceq \mathcal{D}$. Specifically, we want to prove that the valuation function

$$\gamma_{\mathcal{I}} : ([1, n]^* \times \text{Reg}_{C, \mathcal{T}}) \rightarrow \mathcal{D}$$

is a homomorphism. For this, suppose that there is a tuple $((u_1, x_1), \dots, (u_k, x_k)) \in R^{\mathcal{G}}$. By Definition 23 this means that there exist $j \in \{1, \dots, m\}$ and $wv \in (B_j)^{\mathcal{I}_a}$ s.t. θ_j has the form $R(S^{i_1}x_1, \dots, S^{i_k}x_k)$ with depth d_j and s.t. $v = v_1 \dots v_{d_j}$ and $u_t = wv_{i_t}$ for all $t = 1 \dots k$. By Definition 22, this means that $(\gamma_{\mathcal{I}}(u_1, x_1), \dots, \gamma_{\mathcal{I}}(u_t, x_t)) \in R^{\mathcal{D}}$, as wanted.

(\Leftarrow) Now we show that, given an ordinary n -tree interpretation $\mathcal{I} = ([1, n]^*, \cdot^{\mathcal{I}})$ s.t. $\mathcal{I} \models_{\mathcal{T}_a} C_a$ and a homomorphism h from $\mathcal{G}_{\mathcal{I}}$ to \mathcal{D} , we can construct a \mathcal{D} -interpretation \mathcal{J} s.t. $\mathcal{J} \models_{\mathcal{T}} C$. Let's define $\mathcal{J} = ([1, n]^*, \cdot^{\mathcal{J}}, h)$, where $\cdot^{\mathcal{J}}$ coincides with $\cdot^{\mathcal{I}}$ on all concept and role names, and is extended to all concepts using the valuation function h . We can again prove by induction that, for all concepts $E \in \text{Sub}(\mathcal{T}, C)$ and for all $u \in [1, n]^*$, $u \in (E_a)^{\mathcal{I}}$ implies $u \in E^{\mathcal{J}}$ (we show only one interesting case):

Suppose $E = \exists P.c(S^{i_1}x_1, \dots, S^{i_k}x_k)$ where $P = r_1 \dots r_p$ is a role-path of length p and c is a conjunction of atomic constraints $\theta_1 \wedge \dots \wedge \theta_n$ with depths d_1, \dots, d_n such that $0 =: d_0 \leq d_1 \leq \dots \leq d_n \leq d_{n+1} := p$. Then the abstraction of E is

$$E_a = \exists P_1.(B_1 \sqcap \exists P_2.(B_2 \sqcap \dots \exists P_n.(B_n \sqcap \exists P_{n+1}.\top) \dots)$$

with $P_i = r_{d_{i-1}+1} \dots r_{d_i}$. If $u \in (E_a)^{\mathcal{I}}$, then there exists a tuple $(u_0, \dots, u_p) \in P^{\mathcal{I}}$ with $u_0 = u$ and s.t. $u_{d_i} \in (B_i)^{\mathcal{I}}$ for $i = 1, \dots, n$. Fix $i \in \{1, \dots, n\}$, if θ_i has the form $R(S^{j_1}y_1, \dots, S^{j_t}y_t)$, according to Definition 23, this means that $((u_{j_1}, y_1), \dots, (u_{j_t}, y_t)) \in R^{\mathcal{G}}$. Now, since h is a homomorphism from $\mathcal{G}_{\mathcal{I}}$ to \mathcal{D} , we have $(h(u_{j_1}, y_1), \dots, h(u_{j_t}, y_t)) \in R^{\mathcal{D}}$, which means that $\mathcal{D} \models R(h(u_{j_1}, y_1), \dots, h(u_{j_t}, y_t))$. Since this is true for an arbitrary $i \in \{1, \dots, n\}$ this holds true for the conjunction $\theta_1 \wedge \dots \wedge \theta_n$, that is $\mathcal{D} \models c(h(u_{i_1}, x_1), \dots, h(u_{i_k}, x_k))$. Also, since $r^{\mathcal{I}} = r^{\mathcal{J}}$, we know that $(u_0, \dots, u_p) \in P^{\mathcal{J}}$. This means that $u \in E^{\mathcal{J}}$, as wanted. \square

We are now almost ready to give the proof of the main result, of this paper: Theorem 20. We only need a few additional results.

Definition 26. Given an n -tree ordinary interpretation $\mathcal{I} = ([1, n]^*, \cdot^{\mathcal{I}})$, we define an n -tree $T(\mathcal{I})$ over the signature $\{S\} \cup \text{Nc} \cup \text{Nr}$, where Nc and Nr are seen as unary predicates whose interpretation is given by: $A^{T(\mathcal{I})} = A^{\mathcal{I}}$ for each $A \in \text{Nc}$ and $r^{T(\mathcal{I})} = \{xi \in [1, n]^* \mid (x, xi) \in r^{\mathcal{I}}\}$ for all $r \in \text{Nr}$.

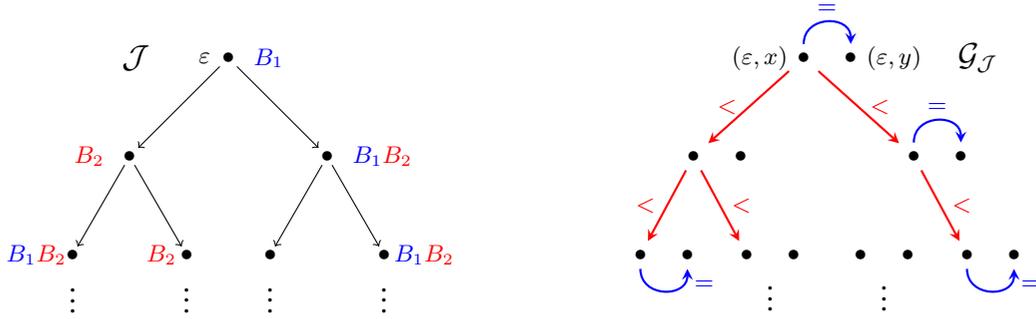


Figure 1. An ordinary 2-tree interpretation (where B_1 and B_2 have non-empty interpretations) and its associated constraint graph $\mathcal{G}_{\mathcal{I}}$ from Example 24.

Remark 27. The only difference between an n -tree interpretation \mathcal{I} and its induced n -tree $T = T(\mathcal{I})$ is that roles are turned into unary predicates s.t., if a pair $(x, y) \in r^{\mathcal{I}}$, now $y \in r^T$. In particular, if we define $\mathcal{G}_T = (([1, n]^* \times \text{Reg}_{C, \mathcal{T}}, R_1^{\mathcal{G}}, R_2^{\mathcal{G}}, \dots)$ (the *constraint graph* of T) exactly as in Definition 23, only substituting the interpretation \mathcal{I} with T , then $\mathcal{G}_{\mathcal{I}} = \mathcal{G}_T$.

Lemma 28. Given C and \mathcal{T} an \mathcal{ALC} -concept and TBox in nnf, we can write a FO-formula φ over the signature $\{S\} \cup \mathbf{N}_{\mathbb{C}} \cup \mathbf{N}_{\mathbb{R}}$, where all elements of $\mathbf{N}_{\mathbb{C}} \cup \mathbf{N}_{\mathbb{R}}$ are seen as unary symbols, s.t. for any n -tree interpretation $\mathcal{I} = ([0, 1]^*, \cdot^{\mathcal{I}})$, $\mathcal{I} \models_{\mathcal{T}} C$ if and only if $T(\mathcal{I}) \models \varphi$.

Proof. The method is similar to the one in [2, Chapter 3], with the only difference that here roles and features are seen as unary predicates added to the second node of the relation. This can be safely done due to the use of tree-shaped models. We define two translations π_x and π_y which inductively map \mathcal{ALC} -concepts to FO formulas with only one free variable, x or y respectively:

- $\pi_i(A) := A(i)$ for each $A \in \mathbf{N}_{\mathbb{C}}$ and $i = x, y$;
- $\pi_i(\neg A) := \neg A(i)$ for each $A \in \mathbf{N}_{\mathbb{C}}$ and $i = x, y$;
- $\pi_i(D \sqcap E) := \pi_i(D) \wedge \pi_i(E)$ for $i = x, y$;
- $\pi_i(D \sqcup E) := \pi_i(D) \vee \pi_i(E)$ for $i = x, y$;
- $\pi_i(\exists r.D) := \exists j. S(i, j) \wedge r(j) \wedge \pi_j(D)$ for $(i, j) = (x, y)$ or (y, x) ;
- $\pi_i(\forall r.D) := \forall j. (S(i, j) \wedge r(j)) \rightarrow \pi_j(D)$ for $(i, j) = (x, y)$ or (y, x) ;

Now let $R_{\mathcal{T}}$ be the set of role names appearing in C and \mathcal{T} , and let $F \subseteq R_{\mathcal{T}}$ be feature names. Keep in mind that the root ε of a tree is definable in FO. We define

$$\psi_{R_{\mathcal{T}}} := \forall (x \neq \varepsilon). \bigvee_{r \in R_{\mathcal{T}}} r(x) \wedge \bigwedge_{r, s \in R_{\mathcal{T}}, r \neq s} \neg(r(x) \wedge s(x))$$

$$\psi_F := \forall x. \forall (y \neq z). S(x, y) \wedge S(x, z) \rightarrow \bigwedge_{f \in F} \neg(f(y) \wedge f(z)).$$

The formula $\psi_{R_{\mathcal{T}}}$ enforces that each pair of elements (x, y) , where y is a successor of x , is assigned a unique role name. ψ_F ensures that the functionality of the features is respected. Then we can prove easily that given a tree-shaped interpretation \mathcal{I} and a TBox $\mathcal{T} = \{C_{\mathcal{T}}\}$, $\mathcal{I} \models_{\mathcal{T}} C$ if and only if $T(\mathcal{I})$ is a model for the following FO formula $\varphi = \exists x. \pi_x(C) \wedge \forall x. \pi_x(C_{\mathcal{T}}) \wedge \psi_{R_{\mathcal{T}}} \wedge \psi_F$. \square

Lemma 29. Let C, \mathcal{T} and φ be as in Lemma 28. Given an n -tree T over the relational signature $\{S\} \cup \mathbf{N}_{\mathbb{R}} \cup \mathbf{N}_{\mathbb{F}}$ that satisfies φ we can build an n -tree interpretation \mathcal{I} s.t. $T = T(\mathcal{I})$ and s.t. $\mathcal{I} \models_{\mathcal{T}} C$.

Sketch of proof. The fact that $T \models \varphi$ means in particular that $T \models \psi_{R_{\mathcal{T}}} \wedge \psi_F$, which guarantees that each node of the tree T is assigned at most one role name, and that the functionality of the features is

respected. We can therefore safely define $A^{\mathcal{I}} = A^T$ for all $A \in \mathbf{N}_{\mathbb{C}}$ and $r^{\mathcal{I}} = \{(x, y) \in ([1, n]^*)^2 \mid S(x, y) \text{ and } y \in r^T\}$ for all $r \in \mathbf{N}_{\mathbb{R}}$ and obtain a tree shaped interpretation. It is easy to see that $T(\mathcal{I}) = T$, and $\mathcal{I} \models_{\mathcal{T}} C$ can be proved by structural induction. \square

Next we show a useful property of BMWB, which is also needed to prove our main result.

Definition 30. Let $k \in \mathbb{N}$ and let $\mathcal{A} = (A, R_1^{\mathcal{A}}, R_2^{\mathcal{A}}, \dots)$ be a structure over the signature σ that does not contain relation symbols $\sim, P_1, P_2, \dots, P_k$ (\sim is binary and all P_i are unary). The k -copy of \mathcal{A} , denoted by $\mathcal{A}^{\times k}$, is the $(\sigma \cup \{\sim, P_1, P_2, \dots, P_k\})$ -structure with domain $(A \times \{1, 2, \dots, k\})$ and

- for all $R \in \sigma$ if R has arity m , $R^{\mathcal{A}^{\times k}}$ is defined as

$$\{((a_1, i), \dots, (a_m, i)) \mid (a_1, \dots, a_m) \in R^{\mathcal{A}}, 1 \leq i \leq k\},$$

- $\sim^{\mathcal{A}^{\times k}} = \{((a, i_1), (a, i_2)) \mid a \in A, 1 \leq i_1, i_2 \leq k\}$, and
- for each $1 \leq m \leq k$, $P_m^{\mathcal{A}^{\times k}} = \{(a, m) \mid a \in A\}$.

Given a structure \mathcal{A} , the k -copy operation creates a new structure, $\mathcal{A}^{\times k}$, which contains k many copies of \mathcal{A} : there are k disjoint substructures of $\mathcal{A}^{\times k}$ (identifiable through the predicates P_1, \dots, P_k) which, seen as σ -structures, are isomorphic to \mathcal{A} . The additional binary predicate \sim relates all those members of $\mathcal{A}^{\times k}$ which are a duplicate of the same element in \mathcal{A} .

The following proposition states that BMWB is *compatible* with the k -copy operation, i.e., whatever property is specified on $\mathcal{A}^{\times k}$ using BMWB can also be recognized by BMWB directly on \mathcal{A} .

Proposition 31 (Prop. 2.26 of [4]). Let $k \in \mathbb{N}$, \mathcal{A} some infinite structure over the signature σ , and $\tau = \sigma \cup \{\sim, P_1, P_2, \dots, P_k\}$ where \sim is a fresh binary relation symbol and k a fresh unary relation symbols P_1, \dots, P_k . Given a BMWB-sentence φ over τ , we can compute a BMWB-sentence φ^k over σ s.t. $\mathcal{A}^{\times k} \models \varphi$ iff $\mathcal{A} \models \varphi^k$.

Let $\tau \subseteq \sigma$, we say a τ -structure \mathcal{A} with domain A is *FO-interpretable* in a σ -structure \mathcal{B} with domain B , if there exists a FO-formula φ such that $A \cong \{b \in B \mid \mathcal{B} \models \varphi(b)\}$, and for each $R \in \tau$ of arity k , there exists an FO-formula φ_R such that $R^{\mathcal{A}} \cong \{(b_1, \dots, b_k) \in B^k \mid \mathcal{B} \models \varphi_R(b_1, \dots, b_k)\}$. Intuitively, we can use FO to *describe* in \mathcal{B} a substructure that is isomorphic to \mathcal{A} .

Lemma 32. Suppose $\text{Reg}_{C, \mathcal{T}} = \{x_1, \dots, x_k\}$, then for an n -tree T over the signature $\{S\} \cup \mathbf{N}_{\mathbb{C}} \cup \mathbf{N}_{\mathbb{R}}$, \mathcal{G}_T is FO-interpretable in $T^{\times k}$.

Proof. The domains of \mathcal{G}_T and $T^{\times k}$, $([1, n]^* \times \{x_1, \dots, x_k\})$ and $([1, n]^* \times \{1, 2, \dots, k\})$ respectively, are in a bijection via the mapping $f : (v, x_k) \mapsto (v, k)$. We extend the bijection f to tuples of elements of $([1, n]^* \times \{x_1, \dots, x_k\})$ as $f(a_1, \dots, a_k) = (f(a_1), \dots, f(a_k))$.

We claim that the relations $R_1^{\mathcal{G}}, R_2^{\mathcal{G}}, \dots$ from \mathcal{G}_T can be represented in $T^{\times k}$ using FO. We describe how: suppose the relation $R \in \sigma$ of arity t is used to form one of the atomic constraints $\theta = R(S^{i_1}y_1, \dots, S^{i_t}y_t)$, with $y_1, \dots, y_t \in \{x_1, \dots, x_k\}$ and $d = \max\{i_1, \dots, i_t\}$. Then we know that a tuple $((v_1, y_1), \dots, (v_t, y_t))$ belongs to $R^{\mathcal{G}}$ if (1) there exist elements $w_0, w_1, \dots, w_d \in [1, n]^*$ s.t. $w_{i_l} = v_l$ for $l = 1, \dots, t$ and $S(w_{j-1}, w_j)$ holds in T for $j = 1, \dots, d$, and (2) $v_d \in B_j^T$. We would like to identify the tuples in $T^{\times k}$ in bijection through f with those tuples in \mathcal{G}_T satisfying Conditions (1) and (2). These are the ones that satisfy the following FO-formula

$$\varphi_{\theta}(a_1, \dots, a_t) = \exists b_0 \dots \exists b_d \bigwedge_{j=1, \dots, d} S(b_{j-1}, b_j) \wedge \bigwedge_{l=1, \dots, t} b_{i_l} \sim a_l \wedge \bigwedge_{i=1, \dots, t} P_{z_i}(a_i)$$

where i_1, \dots, i_t are the same indices appearing in $\theta = R(S^{i_1}y_1, \dots, S^{i_t}y_t)$ and z_i is s.t. $y_i = x_{z_i}$. Once φ_{θ_j} is defined for the atomic constraints $\theta_1, \dots, \theta_n$, which appear in \mathcal{C} and \mathcal{T} , we state the following: if the relation R of arity t is used in all and only $\theta_{j_1}, \dots, \theta_{j_k}$, then $\bar{a} = (a_1, \dots, a_t) \in R^{\mathcal{G}}$ if and only if $\varphi_R(f(\bar{a})) = \varphi_{\theta_{j_1}}(f(\bar{a})) \vee \dots \vee \varphi_{\theta_{j_k}}(f(\bar{a}))$ holds. If the relation $R \in \sigma$ of arity t is not used in any of the atomic constraints $\theta_{j_1}, \dots, \theta_{j_k}$, then there will be no tuple in \mathcal{G}_T which belongs to $R^{\mathcal{G}}$. Therefore $(a_1, \dots, a_t) \in R^{\mathcal{G}}$ iff $\varphi_R(f(\bar{a})) = \perp$ holds. \square

Corollary 33 (of Lemma 32). *If α is a BMWB-formula over the signature σ , we can write a BMWB formula α' over the signature $\{S\} \cup \text{N}_{\mathcal{C}} \cup \text{N}_{\mathcal{R}} \cup \{\sim, P_1, P_n\}$ s.t. $\mathcal{G}_T \models \alpha$ if and only if $T^{\times k} \models \alpha'$.*

Sketch of proof. The formula α' is obtained from α by replacing any occurrence of a formula $R(a_1, \dots, a_t)$ by the formula $\varphi_R(a_1, \dots, a_t)$ defined in the proof of Lemma 32. \square

We are now finally ready to give the proof of our main result.

Proof of Thm. 20. Let \mathcal{C} be an $\mathcal{ALC}^P(\mathcal{D})$ -concept and \mathcal{T} a TBox respectively. Let $n = d \cdot \#_E(\mathcal{T}, \mathcal{C})$ where d is the maximum depth of all constraints that appear in $\text{Sub}(\mathcal{T}, \mathcal{C})$. Due to Lemma 15 we can assume w.l.o.g., that \mathcal{C} and \mathcal{T} are in constraint normal form. By Theorem. 25, we have to check, whether there is an ordinary n -tree interpretation \mathcal{I} s.t. $\mathcal{I} \models_{\mathcal{T}_a} C_a$ and $\mathcal{G}_{\mathcal{I}} \preceq \mathcal{D}$.

Let $\tau \subseteq \sigma$ be the finite subsignature consisting of all relation symbols that occur in \mathcal{C} and \mathcal{T} . Note that $\mathcal{G}_{\mathcal{I}}$ is actually a countable τ -structure. Since the concrete domain \mathcal{D} has the property EHD(BMWB), one can compute from τ a BMWB-sentence α s.t. for every countable τ -structure \mathcal{B} we have $\mathcal{B} \models \alpha$ iff $\mathcal{B} \preceq \mathcal{D}$. Our new goal is to decide whether there is an ordinary n -tree interpretation \mathcal{I} s.t.

$$\mathcal{I} \models_{\mathcal{T}_a} C_a \text{ and } \mathcal{G}_{\mathcal{I}} \models \alpha. \quad (3)$$

Now \mathcal{T}_a and C_a are ordinary \mathcal{ALC} -concepts. We can use Lemma 28, Lemma 29 and Remark 27, and obtain a FO formula φ s.t. if C_a is satisfied w.r.t. \mathcal{T}_a by some n -tree interpretation \mathcal{I} , then φ is satisfied by an n -tree T s.t. $\mathcal{G}_{\mathcal{I}} = \mathcal{G}_T$. Also, if φ is satisfied by some n -tree T , then there exists an n -tree interpretation \mathcal{I} s.t. $\mathcal{I} \models_{\mathcal{T}_a} C_a$ and s.t. $\mathcal{G}_T = \mathcal{G}_{\mathcal{I}}$.

Then finding \mathcal{I} s.t. (3) holds is equivalent to finding an n -tree T s.t. $T \models \varphi$ and $\mathcal{G}_T \models \alpha$. By Corollary 33, we can find a BMWB-formula β s.t. $\mathcal{G}_T \models \alpha$ iff $T^{\times k} \models \beta$. But we also know, due to Proposition 31, that we can compute a formula β^k s.t. $T^{\times k} \models \beta$ iff $T \models \beta^k$. At this point we have to check whether there exists an

n -tree T s.t. $T \models \varphi \wedge \beta^k$, where $\varphi \wedge \beta^k$ is a BMWB-sentence. By Theorem 3 this is decidable, which completes the proof. \square

5 Conclusion and Future Work

We have introduced a novel way to integrate concrete domains in \mathcal{ALC} , via *path constraints*. The resulting logic, $\mathcal{ALC}^P(\mathcal{D})$, is of incomparable expressiveness with the several variants of $\mathcal{ALC}(\mathcal{D})$ that are present in the literature. We have seen, however, how on the domains that we are interested in, our logic is strictly more expressive: $\mathcal{ALC}^P(\mathcal{D})$ allows not only feature-paths, but also full role-paths, to connect abstract individuals and their concrete attributes.

We exploit the path-structure of the constraints to show that $\mathcal{ALC}^P(\mathcal{D})$ is compatible with the EHD-method from [6] and show the very general result: satisfiability for $\mathcal{ALC}^P(\mathcal{D})$ is decidable w.r.t. general TBoxes, if the concrete domain \mathcal{D} is negation-closed and has the EHD-property. This solves the problem that has been open for more than a decade (see [13]), whether reasoning in \mathcal{ALC} with non-dense concrete domains such as the natural numbers or the integers would be decidable in the presence of general TBoxes, since these domains enjoy our required properties. Such domains did not satisfy the ω -admissibility criterion that was formulated in [16]. In this sense, we prove that this ω -admissibility is not a necessary condition to guarantee the decidability of reasoning over a concrete domain in the presence of general TBoxes.

We could have easily chosen a more expressive DL than \mathcal{ALC} as underlying logic. In principle we could add any concept constructor preserving the tree model property, and that can be then translated to MSO over trees with one successor and unary predicates only (see lemma 28). Examples of such constructors would be transitive closure, role hierarchy and qualified number restriction.

The main open question remains the complexity. The EHD method is a reduction to satisfiability of WMSO+B over infinite binary trees, which is shown to be decidable in [3]. Here the authors do not provide complexity bounds for their decision procedure. On the other hand, the WMSO+B-formulas that need to be checked for decidability are fixed and depend solely on the concrete domain. Roughly speaking, once we fix our domain \mathcal{D} , the EHD method transforms a given $\mathcal{ALC}^P(\mathcal{D})$ -TBox and -concept into a *constraint normal form* which already results in a blow-up of the size. This in turn get transformed into an MSO-formula φ (which is clearly non optimal). We then have to decide whether a conjunction of φ and a fixed WMSO+B-formula ψ (which depends on \mathcal{D}) is decidable. Analyzing this procedure would very hardy lead to tight complexity bounds. In our opinion the EHD-method is more of an admissibility criterion, which provides easy conditions on a concrete domain \mathcal{D} to establish whether reasoning with it remains decidable or not.

Also, it would be interesting to know if one can add constant predicates of the form $(=_q)_{q \in \mathbb{Q}}$ to the domain \mathbb{Q} from Prop. 19 and prove that the resulting structure still has the EHD-property. We conjecture that a method similar to the one presented in [7] for constant predicates over the integers could be applied to this case.

Another follow-up question is whether the EHD method be can adapted to show decidability for *fuzzy* concrete domains, similarly as it was shown in [17] for the criterion of ω -admissibility.

ACKNOWLEDGEMENTS

This work is supported by the German Research Foundation (DFG) within the Collaborative Research Center SFB 912 – HAEC and the Graduiertenkolleg 1763 (QuantLA).

REFERENCES

- [1] Franz Baader and Phillip Hanschke, ‘A scheme for integrating concrete domains into concept languages’, in *Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI-91*, pp. 452–457, Sydney (Australia), (1991).
- [2] Franz Baader, Ian Horrocks, and Ulrike Sattler, ‘Description logics’, in *Handbook of Knowledge Representation*, eds., F. van Harmelen, V. Lifschitz, and B. Porter, 135–179, Elsevier, (2007).
- [3] Mikołaj Bojańczyk and Szymon Toruńczyk, ‘Weak MSO+U over infinite trees’, in *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, eds., Christoph Dürr and Thomas Wilke, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 648–660, Dagstuhl, Germany, (2012). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [4] Claudia Carapelle, *On the satisfiability of temporal logics with concrete domains*, Ph.D. dissertation, Universität Leipzig, 2015. <http://nbn-resolving.de/urn:nbn:de:bsz:15-qucosa-190987>.
- [5] Claudia Carapelle, Shiguang Feng, Alexander Kartzow, and Markus Lohrey, ‘Satisfiability of ECTL* with Tree Constraints’, in *Computer Science – Theory and Applications*, eds., Lev D. Beklemishev and Daniil V. Musatov, volume 9139 of *Lecture Notes in Computer Science*, 94–108, Springer International Publishing, (2015).
- [6] Claudia Carapelle, Alexander Kartzow, and Markus Lohrey, ‘Satisfiability of CTL* with Constraints’, in *CONCUR 2013 – Concurrency Theory*, eds., Pedro R. D’Argenio and Hernán C. Melgratti, volume 8052 of *Lecture Notes in Computer Science*, pp. 455–469. Springer Berlin Heidelberg, (2013).
- [7] Claudia Carapelle, Alexander Kartzow, and Markus Lohrey, ‘Satisfiability of ECTL* with Constraints’, *Journal of Computer and System Sciences*, (2016). To appear.
- [8] Claudia Carapelle and Anni-Yasmin Turhan, ‘Decidability of $ALCP(\mathcal{D})$ for concrete domains with the EHD-property’, LTCS-Report 16-01, Chair of Automata Theory, TU Dresden, (2016). See <https://lat.inf.tu-dresden.de/research/reports/2016/CaTu-LTCS-16-01.pdf>.
- [9] Volker Haarslev, Ralf Möller, and Michael Wessel, ‘The description logic $ALCN\mathcal{H}_{R+}$ extended with concrete domains: A practically motivated approach’, in *In Proceedings of the First International Joint Conference on Automated Reasoning, IJCAR*, eds., R. Goré, A. Leitsch, and T. Nipkow, volume 2083 of *Lecture Notes in Computer Science*, pp. 29–44. Springer, (2001).
- [10] Ian Horrocks and Ulrike Sattler, ‘Ontology reasoning in the $\mathcal{SHOQ}(\mathcal{D})$ description logic’, in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI*, ed., B. Nebel, pp. 199–204, (2001).
- [11] Carsten Lutz, ‘Nexptime-complete description logics with concrete domains’, LTCS-Report LTCS-00-01, LuFG Theoretical Computer Science, RWTH Aachen, Germany, (2000). See <http://www-iti.informatik.rwth-aachen.de/Forschung/Reports.html>.
- [12] Carsten Lutz, ‘Adding numbers to the \mathcal{SHIQ} description logic—First results’, in *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR’02)*, pp. 191–202. Morgan Kaufman, (2002).
- [13] Carsten Lutz, ‘Description logics with concrete domains—a survey’, in *Advances in Modal Logic 2002 (AiML 2002)*, Toulouse, France, (2002). Final version appeared in *Advanced in Modal Logic Volume 4*, 2003.
- [14] Carsten Lutz, ‘Combining interval-based temporal reasoning with general TBoxes’, *Artificial Intelligence*, **152**(2), 235–274, (2004).
- [15] Carsten Lutz, ‘NExpTime-complete description logics with concrete domains’, *ACM Transactions on Computational Logic*, **5**(4), 669–705, (2004).
- [16] Carsten Lutz and Maja Miličić, ‘A tableau algorithm for description logics with concrete domains and general TBoxes’, *Journal of Automated Reasoning*, **38**(1-3), 227–259, (2007).
- [17] Dorian Merz, Rafael Peñaloza, and Anni-Yasmin Turhan, ‘Reasoning in ALC with fuzzy concrete domains’, in *Proceedings of 37th edition of the German Conference on Artificial Intelligence (KI’14)*, eds., Carsten Lutz and Michael Thielscher, volume 8736 of *Lecture Notes in Artificial Intelligence*, pp. 171–182. Springer Verlag, (2014).
- [18] Wolfgang Thomas, ‘Languages, automata, and logic’, in *Handbook of Formal Languages*, pp. 389–455. Springer, (1996).
- [19] David Toman and Grant E. Weddell, ‘Applications and extensions of PTIME description logics with functional constraints’, in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 948–954, (2009).
- [20] W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation, 27th October 2009. <http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>.

Realisability of Production Recipes

Lavindra de Silva¹ and Paolo Felli¹ and Jack C. Chaplin¹ and
Brian Logan² and David Sanderson¹ and Svetan Ratchev¹

Abstract. There is a rising demand for customised products with a high degree of complexity. To meet these demands, manufacturing lines are increasingly becoming autonomous, networked, and intelligent, with production lines being virtualised into a manufacturing cloud, and advertised either internally to a company, or externally in a public cloud. In this paper, we present a novel approach to two key problems in such future manufacturing systems: the *realisability problem* (whether a product can be manufactured by a set of manufacturing resources) and the *control problem* (how a particular product should be manufactured). We show how both production recipes specifying the steps necessary to manufacture a particular product, and manufacturing resources and their topology can be formalised as labelled transition systems, and define a novel simulation relation which captures what it means for a recipe to be realisable on a production topology. We show how a controller that can orchestrate the resources in order to manufacture the product on the topology can be extracted from the simulation relation, and give an algorithm to compute a simulation relation and a controller.

1 INTRODUCTION

Automating the manufacture of complex products is key to competitiveness in high-labour cost economies. However, automating the assembly of manufactured products presents significant challenges. It is widely acknowledged that responsiveness and customisability are key to the future of manufacturing [19], resulting in a drive towards “*batch-size-of-one*” production, in which each item produced differs from the items assembled immediately before and immediately after it [4]. In addition, there is drive towards “*manufacturing-as-a-service*” and the virtualisation and networking of resources to create *Cloud Manufacturing*, in which manufacturing software and resources are advertised and shared between members of a cloud [14]. In cloud environments, products may be manufactured by multiple cloud participants, representing multiple different enterprises connected via a supply chain. This trend toward flexible, adaptive, intelligent, and networked manufacturing systems has been termed the *fourth industrial revolution* or *Industrie 4.0*, in which decentralised intelligence in an *Internet of Things* connects embedded production resources to form “*smart factories*” that communicate and collaborate [12].

There is a growing body of work on automation to achieve flexibility, resilience, and monitoring in manufacturing. For example, *Flexible Manufacturing Systems* [7, 20, 10] increase the range of products that may be assembled, and *Reconfigurable Manufacturing Systems* [5, 13, 15, 21] reduce response time. However, to date, there has

been little work on the assembly of highly-customised products in a highly-networked manufacturing environment.

In such a setting, the set of products that will be manufactured is not known in advance. Rather, production recipes specifying the steps necessary to manufacture a particular product are matched against virtualised manufacturing resources (such as a plant or assembly line), and the matched resource must then self-configure to assemble the product. Manufacturing control software must therefore make decisions both about *whether* a particular product can be manufactured by a particular set of manufacturing resources, and *how* a particular customised product should be manufactured by the resources (the steps an assembly line must perform in order to assemble the product). Whether a product can be manufactured by a set of manufacturing resources is termed the *realisability problem*; how it should be manufactured by those resources is termed the *control problem*.

In this paper, we present a new approach to the realisability and control problems in manufacturing. We show how both production recipes and manufacturing resources and their topology can be formalised as labelled transition systems, and define a novel simulation relation which captures what it means for a recipe to be realisable on a production topology. We give an algorithm for computing the simulation relation, and show how a controller to manufacture a product specified by a production recipe on a topology can be extracted from the simulation relation. To the best of our knowledge, our approach is the first fully-automated solution to the realisability and control problems in manufacturing. It forms a key component of the Evolvable Assembly Systems (EAS) architecture, an agent-based architecture for manufacturing control software designed to address rapidly changing product and process requirements including batch-size-of-one customised production [8]. The EAS architecture is being evaluated on a number of real-world production system demonstrators, and we illustrate our approach with a simple example of the manufacture of products (automotive hinges) by a flexible assembly platform.

2 ASSEMBLY SYSTEMS

In this section, we motivate and informally introduce the key ideas needed for the formal development, including production recipes, production resources, and their topology.

A *production recipe* specifies the steps necessary to manufacture a particular product, including the constituent parts, the tasks and associated parameters required to process and assemble these parts into the final product, any tests that must occur to verify the product during and at the end of the manufacturing process, and how to respond to the results of tests (e.g., whether a partially completed product instance should be reworked or discarded following a test).

In industry, when recipe information is passed from the higher-

¹ Institute for Advanced Manufacturing, Faculty of Engineering, University of Nottingham, first.last@nottingham.ac.uk

² School of Computer Science, University of Nottingham, bsl@cs.nott.ac.uk

level enterprise control systems to lower level shop-floor control systems, it is usual for a “*recipe file*” to be used. Recipe files may be specified in a proprietary format, or the ANSI/ISA-95 (Enterprise-Control System Integration) family of standards [1] (also known as IEC/ISO 62264 [3]). One common approach to implementing the data models in the ISA-95 standard, is to use Business To Manufacturing Markup Language (B2MML) [2] XML schemas. Messages in B2MML contain an operation schedule, which specifies each operation to be performed. An operation consists of one or more requirements (e.g., for personnel, equipment, physical assets, or material), and precisely specifies the location and time the operation should be performed. While languages such as B2MML allow a *solution* to the control problem to be expressed (i.e., which resource should perform which operation on which part, and when), they do not provide an appropriate level of abstraction for specifying the *inputs* to the realisability and control problems. In our approach, we therefore specify recipes in a language that has some similarities to Hierarchical Task Networks (HTNs) [11] and Belief-Desire-Intention (BDI) [18] agent systems (see Section 3). For example, like these languages, we allow steps to be partially ordered (including interleaved), and do not specify which resource should perform each step.

A production recipe is enacted by a set of *production resources*. Each resource has a set of capabilities describing the basic actions it can perform, and in which order. Production resources are connected by transport links (conveyor belts, shuttles, manual item movement, etc.) to form a manufacturing line with a specific *production topology*. The production topology implicitly determines which production recipes can be manufactured by the line.

To illustrate these ideas, we briefly describe the Precision Assembly Demonstrator (PAD), a flexible assembly platform consisting of four production resources (see Figure 1a): two KUKA robot arms each with an associated workspace allowing the robots to place parts and perform operations, a testing and inspection workstation that can perform mechanical and vision-based tests, and a manual loading and unloading station where pallets of parts can be added or removed. There is additionally a shared, passive tool-changing rack between the two robot arms, and a shuttle transport system.

The shuttle transport system links workstations to give the production topology shown in Figure 1b. The shuttle transports a pallet carrier, which allows one pallet of parts at a time to be moved between resources. The robots use gripper end effectors to pick up or return pallets from the shuttle and place them on their respective workstations, where they can change end effectors from the tool rack and perform a variety of tasks.

The PAD demonstrator’s primary production recipe family is a detent hinge for the use in automotive interiors. The simplest production recipe consists of a hollow plastic hinge consisting of two leaves (an interior and an exterior), which are linked with a metal hinge pin. More complex recipes are achieved by adjusting the hinge detent force by adding up to three metal balls and springs in slots in the interior hinge leaf. Each spring-ball pair increases the force required to engage the hinge. The robots assemble the hinge using a variety of gripper end effectors. A wide range of new end effectors may be added to alter the capabilities of the robot arms, such as new grippers for alternative hinge designs, glue applicators for securing the hinge pin, or engraving tools to give hinges serial codes. The flexibility of the PAD assembly platform results in both a realisability problem (how to determine if a recipe can be produced on a given set of hardware), and a control problem (how to manage the production of recipes on the hardware).

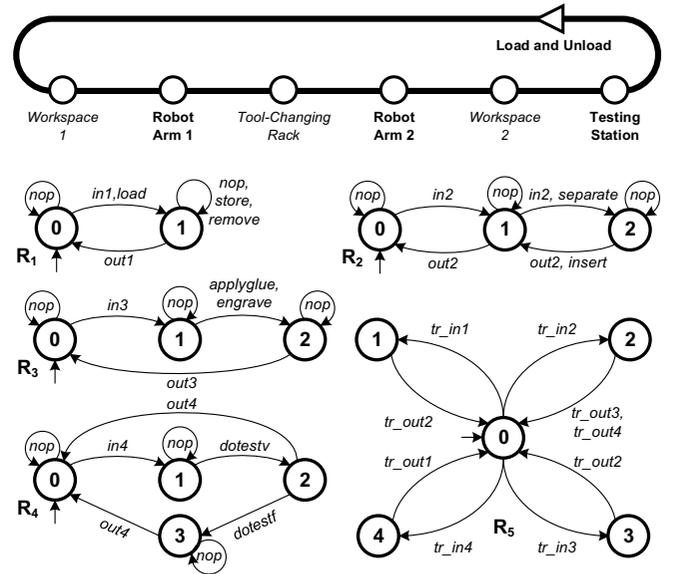


Figure 1: (a) Our Precision Assembly Demonstrator (top); (b) its diagrammatic layout (centre); and (c) its LTS model (bottom).

3 PRODUCTION RECIPE

In this section, we formalise production recipes as labelled transition systems (LTS), where labels are not just atomic steps but complex “task expressions”.

We will make use of two sets: a set C of *part constants*, which represent parts used in production, and a set of *task labels*, or simply *tasks*, which represent operations. The latter set is the union of three mutually disjoint sets: the set of *observable* tasks T_{ob} , which only occur in recipes and non-transport resources; the set of *internal* tasks T_{in} , which only occur in transport resources; and the set of *synchronisation* tasks T_{syn} , which are used to “transfer” parts between production resources. Specifically, $T_{syn} = \{t_{-}^x | x \in \mathbb{N}\} \cup \{t_{+}^x | x \in \mathbb{N}\}$, namely, the set of *in* (t_{-}^x) and *out* (t_{+}^x) synchronisation tasks; parts are moved from resources performing *out* tasks (releasing a part) to corresponding resources performing *in* tasks (accepting a part). Further, we introduce the special task *nop* to denote the special “no-op” task, which represents idling.

Production recipes specify how (though not *where*) tasks and checks ought to be carried out in order to manufacture a desired end-product. Tasks are specified by *task expressions*.

The smallest task expression is of the form $t(c, c')$, and is called a *parameterised task* (*p-task*), where $t \in T_{ob}$, and $c \in C^*$ and $c' \in C^+$ are sequences of part constants such that any constant in C occurs at most once in c and in c' . Given a p-task $t(c, c')$, sequences c and c' represent the “input” and “output” parameters of t , respec-

tively: they represent the part(s) on which t is performed, and the possibly new part(s) that results from doing t_i . We use ϵ to denote the empty sequence, and denote c by ${}^\circ t$ and c' by t° . We sometimes write t instead of $t(c, c')$ when c and c' are not relevant.

For instance, the task $cut(c, c1 \cdot c2)$ represents an operation that takes a part c and produces two parts c_1 and c_2 , whereas the task $load(\epsilon, c)$ represents an operation that introduces a new part c into the production facility.

The set of *general task expressions* is the smallest set of formulas, denoted by $Lang(\mathcal{T}_g)$, generated by the grammar $\mathcal{T}_g := ?\phi : \mathcal{T}$, where ϕ is a propositional formula from a propositional language P and \mathcal{T} is a task expression. A *task expression* is a formula in the language generated by the grammar:

$$\mathcal{T} := t(c, c') \mid \mathcal{T}; \mathcal{T}' \mid \mathcal{T} \parallel \mathcal{T}' \mid \mathcal{T} \text{ “} \mid \text{” } \mathcal{T}'$$

The operator “;” denotes a sequence; “ \parallel ” denotes parallel composition; and “ \mid ” denotes interleaved composition. We refer to a p-task or a parallel composition (of p-tasks) as an *atomic task expression* (or as *atomic*).

We impose two additional constraints on task expressions:

- any expression $\mathcal{T}_1 \parallel \dots \parallel \mathcal{T}_m$ occurring in \mathcal{T} is restricted such that each \mathcal{T}_i is a p-task, and for each pair t_i, t_j of p-tasks, it does not hold that t_i and t_j mention the same part, and ${}^\circ t_i = {}^\circ t_j = \epsilon$;
- any interleaving $\mathcal{T}_1 \mid \dots \mid \mathcal{T}_m$ occurring in \mathcal{T} is restricted such that each \mathcal{T}_i does not mention operator “ \mid ”.

The first constraint restricts parallelism to atomic tasks, and requires that parallel atomic tasks cannot share the same part constants, as a part can be a parameter of only one task at a time. It also forbids two or more parallel tasks with empty input parameters. The second constraint forbids “nested” interleaving.

The general task expression $?\phi : \mathcal{T}$ specifies that \mathcal{T} can only happen if the guard ϕ holds, based on a valuation for ϕ that is available at runtime (based on data collected in real-time). Since our analysis is instead performed offline, in the remainder of this paper, we will ignore any guard appearing in general task expressions, thus considering any valuation as equally possible. Nonetheless, guards in general task expressions stress the fact that any formula in the language of task expressions $Lang(\mathcal{T})$ may be associated with a guard.

With these definitions at hand, it is now possible to define our notion of *production recipe* (or simply recipe). A *recipe* is a labelled transition system in which labels are task expressions and nodes represent the states of parts in the assembly. This definition essentially allows for loops in task expressions, and for the specification of alternatives among task expressions.

Definition 1 (Recipe) A *recipe* is a tuple $\mathbf{R} = (s^0, S, L, \rightarrow)$, where S is a finite set of states, $s^0 \in S$ is the initial state, $L \subseteq Lang(\mathcal{T})$ is a set of task expressions, and $\rightarrow : S \times L \mapsto S$ is a non-empty transition function. We denote a transition from state s to s' , with task expression \mathcal{T} , either by $s \xrightarrow{\mathcal{T}} s'$ or $(s, \mathcal{T}, s') \in \rightarrow$.

Figure 2 shows an example of a recipe to be executed on the assembly platform in Figure 1. The first two p-tasks request a new pallet fixture (f) to be loaded and then separated into the hinge pin (p) and hollow hinge (h). These are followed by an interleaved composition, which requires glue to be applied to p , and for h to be engraved with a serial number (in any order). The two parts are then combined to form the final hinge ($h2$), by inserting p into h . The next step requests a visual test to be performed on $h2$, which simply collects test data. After this, the recipe either requests a force test,

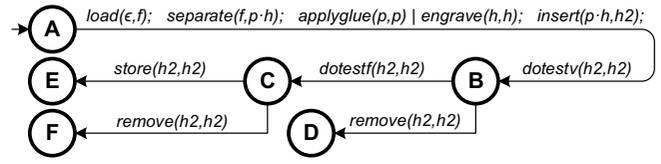


Figure 2: An example of a recipe. For readability, the sequence labelling the edge between A and B has been separated.

or for the hinge to be removed, based on the outcome of evaluating guards (not shown) against real-time visual-test data. Similarly, the outcome of evaluating guards against real-time force-test data determines whether the hinge is recycled or stored for delivery.

A *trace* of a recipe \mathbf{R} is a sequence $\pi = s_0 \xrightarrow{\mathcal{T}_1} s_1 \xrightarrow{\mathcal{T}_2} \dots \xrightarrow{\mathcal{T}_n} s_n$, namely, a sequence of states and transitions, of arbitrary length $n \geq 0$, such that $s_{i-1} \xrightarrow{\mathcal{T}_i} s_i$ for each $i \in [1, n]$. If not specified, we assume $s_0 = s^0$.

Finally, we assume that cycles in a recipe are bounded, in order to ensure that manufacturing the end-product will eventually cease. Since bounded cycles can easily be removed by unfolding the recipe graph, in what follows we deal only with acyclic recipes, i.e., recipes where the edge-labelled graph (S, \rightarrow) is acyclic.

4 PRODUCTION RESOURCE & TOPOLOGY

A *production resource* represents a production “station” (or “cell”), which performs observable operations on parts. A *transport resource* represents a transport mechanism such as a conveyer belt, which performs internal operations that route parts between stations. We model production and transport resources as labelled transition systems, where a node represents a state of the resource, and an outgoing edge from a node corresponds to a task that can be performed from that node.

Definition 2 (Resource) A *production (resp. transport) resource* is a tuple $(s^0, \underline{S}, \underline{L}, \Longrightarrow)$, where \underline{S} is a set of states, $s^0 \in \underline{S}$ is the initial state, $\underline{L} \subseteq T_{ob} \cup T_{syn} \cup \{nop\}$ (resp. $\underline{L} \subseteq T_{in} \cup T_{syn} \cup \{nop\}$) is a set of tasks,³ and $\Longrightarrow : \underline{S} \times \underline{L} \mapsto \underline{S}$ is a non-empty transition function.

Intuitively, the *nop* task allows a resource to idle for either a fixed or an unbounded number of steps, before and/or after performing other tasks. Note that \Longrightarrow is a function: we assume that when a resource performs a task from a state, there exists only one possible successor state for that task.

Figure 1c shows a simplified model of the resources in Figure 1. Multiple transitions between two states are depicted as one, with labels separated by a comma. Resource R_1 can load a new pallet (containing a hollow hinge and hinge pin), and also store a hinge for removal or delivery. Labels *in1* and *out1* are *in* and *out* synchronisation tasks, respectively. Resource R_2 is the first robotic arm in Figure 1, which can either accept (via *in2*) a pallet fixture and separate it into the hollow hinge and pin, or accept these two parts, one after the other, and then insert the pin into the hinge, thereby again producing a single part. The second robotic arm is modelled as R_3 , which can either apply glue to a given part, or engrave it with a serial number. Resource R_4 is the testing station, which can first analyse a hinge to gather visual data about its assembly, followed possibly by

³ We could also allow production resources to specify internal tasks, which would then enable operations such as data logging (e.g. from sensors).

an analysis to gather data about hinge force. The latter is not allowed before doing the former because a wrongly assembled hinge could become damaged if it is manipulated to gather force data. Moreover, due to design constraints, the resource also cannot idle between these two operations, represented by the lack of a *nop* transition.

Finally, R_5 is a transport resource, modelling the shuttle system in Figure 1. This resource represents the “legal routes” between production resources: e.g., while a newly loaded fixture can be delivered from R_1 to R_2 , delivering a glued part from R_3 to R_4 is forbidden, to prevent the part from becoming affixed during force analysis. Thus, intuitively, “out” synchronisations in production resources correspond to “in” synchronisations in transport resources, and vice versa. For example, *out1* in R_1 corresponds to *tr_in1* in R_5 , and *tr_out2* in R_5 corresponds to *in2* in R_2 .

A *production topology* (or simply *topology*) “connects” the available resources via synchronisation tasks, and represents the layout of the production system. Technically, a topology is the cross product of elements in resource tuples, excluding “invalid” transitions, i.e., those having an *out* synchronisation without the corresponding *in* synchronisation (and vice versa).

In what follows, we use two auxiliary notions. First, for any pair of states $\underline{s}, \underline{s}' \in \underline{S}$ and label $\mathbf{t} \in \underline{L}$, we use $\underline{s} \xrightarrow{\mathbf{t}} \underline{s}'$ to denote $(\underline{s}, \mathbf{t}, \underline{s}') \in \Longrightarrow$. Second, we define the *complement* of a synchronisation task $t \in T_{syn}$, denoted by $\sim t$, as t_x^- if $t = t_x^+$, and as t_x^+ otherwise, where $x \in \mathbb{N}$. For instance, referring to Figure 1c, if $t = out1$, then $\sim t = tr_in1$. While, strictly speaking, the definition requires $\sim t = in1$, in the figure we have used the prefix “tr” for the purpose of readability. Taking inspiration from [6], we define a production topology as follows.

Definition 3 (Topology) Let R_1, \dots, R_n be resources, where each $R_i = (\underline{s}_i^0, \underline{S}_i, \underline{L}_i, \Longrightarrow_i)$. A *topology* is a tuple $(\underline{s}^0, \underline{S}, \underline{L}, \Longrightarrow)$, where $\underline{S} = \underline{S}_1 \times \dots \times \underline{S}_n$ is the set of states; $\underline{s}^0 = (\underline{s}_1^0, \dots, \underline{s}_n^0)$ is the initial state; $\underline{L} = \underline{L}_1 \times \dots \times \underline{L}_n$ is the set of concurrent tasks; and the transition relation \Longrightarrow is such that for any $\underline{s}, \underline{s}' \in \underline{S}$ and $\mathbf{t} \in \underline{L}$, we have $\underline{s} \xrightarrow{\mathbf{t}} \underline{s}'$ iff for all $i \in [1, n]$:⁴

1. $t_i \notin T_{syn}$ and $\underline{s}_i \xrightarrow{t_i} \underline{s}'_i$; or
2. $t_i \in T_{syn}$ and $\underline{s}_i \xrightarrow{t_i} \underline{s}'_i$, and there exists exactly one $j \in [1, n]$ such that $\underline{s}_j \xrightarrow{t_j} \underline{s}'_j$ and $t_j = \sim t_i$.

Thus, the topology depicts the concurrent execution of tasks t_i in different resources. Without loss of generality, condition (2) checks that, within a single transition, a particular synchronisation only takes place between one resource and exactly one other resource. This is because we later use synchronisations to unambiguously transfer parts between resources.

5 REALISABILITY OF A RECIPE ON A TOPOLOGY

We shall now define what it means for a recipe to be realisable on a topology. Our definition relies on some auxiliary notions, particularly involving the movement of parts and the execution of a recipe on a topology.

The notion of a topology as defined above is “static” in that it does not account for parts which are moved between resources and manipulated by them. Thus, given a topology $(\underline{s}^0, \underline{S}, \underline{L}, \Longrightarrow)$, we keep track of the movement of parts (constants) during production via a *resource vector* $\mathbf{r} = (\mathbf{c}_1, \dots, \mathbf{c}_n)$, where each $\mathbf{c}_i \in C^*$ is a (possibly empty) sequence of parts that do not occur anywhere else in

\mathbf{r} ; we denote \mathbf{c}_i by $\mathbf{r}(i)$ for any $i \in [1, n]$, and the set of all possible resource vectors as V . Intuitively, we associate each state in the topology with a resource vector \mathbf{r} , specifying which parts are currently allocated to each resource. Note that each element in vector \mathbf{r} is a sequence and not a set: we assume a first-in-first-out approach when moving parts between resources.

Part constants get allocated to resources as p-tasks in recipes are processed. A resource R_j currently in state \underline{s}_j can execute an (atomic) p-task $t(\mathbf{c}, \mathbf{c}')$ only if (a) the task t is available from state \underline{s}_j in R_j , and (b) the parts ${}^\circ t$ appearing as the input parameters of t are currently allocated to the resource, namely, $\mathbf{r}(j) = \mathbf{c}$. After its execution, parts \mathbf{c}' appearing as its output parameters (i.e., t°) are allocated to R_j . For example, the *separate(f, p, h)* p-task in Figure 2 is executable on resource R_2 in Figure 1c only if part f is currently allocated to R_2 , and the resource is in state 1. Executing the task results in f being “removed” and R_2 being allocated parts p and h .

Formally, given a task expression $\mathcal{T} = t_1 \parallel \dots \parallel t_m$, a resource vector \mathbf{r} , a state \underline{s} , and a transition $\underline{s} \xrightarrow{\mathbf{t}} \underline{s}'$ with $\mathbf{t} = (t'_1, \dots, t'_n)$, a resource vector \mathbf{r}' is an *allocation* of \mathcal{T} to \mathbf{t} with respect to \mathbf{r} , denoted $\mathbf{r}' \in \text{AL}(\mathbf{r}, \mathcal{T}, \mathbf{t})$, if and only if

- $\forall j \in [1, n]$, either $t'_j \notin T_{ob}$ and $\mathbf{r}'(j) = \mathbf{r}(j)$, or $\exists i \in [1, m]$ s.t.

$$(a) t'_j = t_i \quad (b) \mathbf{r}(j) = {}^\circ t_i \text{ and } \mathbf{r}'(j) = t_i^\circ$$
- $\forall i \in [1, m]$, $\exists j \in [1, n]$ s.t. (a) and (b).

In other words, there must exist a one-to-one mapping from p-tasks t_i in \mathcal{T} to observable tasks t'_j in transition \mathbf{t} , and from the parts associated with t_i to the ones corresponding to t'_j (i.e., $\mathbf{r}(j)$).

Allocated parts are transferred across resources by synchronisation tasks. Formally, a resource vector \mathbf{r}' is a *transfer* or “move” of parts from \mathbf{r} relative to \mathbf{t} (\mathbf{r} and \mathbf{t} are as above), denoted $\mathbf{r}' = \text{MOV}(\mathbf{r}, \mathbf{t})$, if \mathbf{r}' is obtained from \mathbf{r} by replacing each $\mathbf{c}'_i, \mathbf{c}'_j \in \mathbf{r}$ ($\mathbf{c}'_i = c_1 \dots c_m$) with respectively $c_2 \dots c_m$ and $\mathbf{c}'_j \cdot c_1$, only if $\mathbf{t}(i) = t_x^+$, $\mathbf{t}(j) = t_x^-$ ($x \in \mathbb{N}$), and $m > 0$; if $m = 0$, then $\mathbf{r}' = \mathbf{r}$.⁵ For example, the part p in resource R_2 in Figure 1c is moved to resource R_5 via the synchronisation involving tasks *out2* and *tr_in2*.

Observe that a resource vector encodes the allocation of parts to resources (that is, which parts are currently allocated to each resource), and part allocations are not considered in the description of the resource itself. In other words, the description of a resource is purely behavioural, in the sense that it encodes the sequence of operations that it can (or is allowed to) execute, which is not constrained by the presence of a part in a given internal state as in the case of, e.g., Petri Nets [17]. The size of a resource vector is thus equal to the number of resources, and the allocation of parts is independent from the current state of the topology. This is an essential point to guarantee our complexity result.

The final auxiliary notion needed to define the execution of a recipe on a topology is the standard notion of a *linearisation*, which we have adapted for interleaved compositions. Formally, a *linearisation* \mathcal{T}' of a task expression $\mathcal{T} = \mathcal{T}_1 \mid \dots \mid \mathcal{T}_m$, denoted $\mathcal{T}' \in \text{LIN}(\mathcal{T})$, is defined inductively as follows. If $m \in \{1, 2\}$, then \mathcal{T}' is any sequence of the form $\mathcal{T}_1^1; \mathcal{T}_2^1; \mathcal{T}_1^2; \mathcal{T}_2^2; \dots; \mathcal{T}_1^m; \mathcal{T}_2^m$, where $\mathcal{T}_1 = \mathcal{T}_1^1; \dots; \mathcal{T}_1^m$, $\mathcal{T}_2 = \mathcal{T}_2^1; \dots; \mathcal{T}_2^m$, and each \mathcal{T}_i^j is a possibly empty sequence. If $m > 2$, then

$$\mathcal{T}' \in \bigcup_{\mathcal{T}'' \in \text{LIN}(\mathcal{T}_2 \mid \dots \mid \mathcal{T}_m)} \text{LIN}(\mathcal{T}_1 \mid \mathcal{T}'')$$

⁴ Recall that $\underline{s} = (\underline{s}_1, \dots, \underline{s}_n)$, $\underline{s}' = (\underline{s}'_1, \dots, \underline{s}'_n)$ and $\mathbf{t} = (t_1, \dots, t_n)$.

⁵ Given a label $\mathbf{t} = (t_1, \dots, t_k)$, we denote t_i by $\mathbf{t}(i)$, for any $i \in [1, k]$.

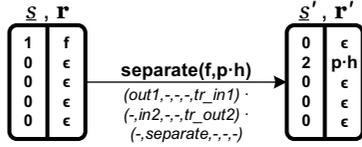


Figure 3: A graphical representation of a recursive application of the NXT operator. Given the couple $(\underline{s}, \mathbf{r})$, where $\underline{s} = (1, 0, 0, 0, 0)$ and $\mathbf{r} = (f, \epsilon, \epsilon, \epsilon, \epsilon)$, the state $(\underline{s}', \mathbf{r}')$ is in $\text{NXT}((\underline{s}, \mathbf{r}), \text{separate}(f, p \cdot h))$ due to two unobservable transitions (part f is moved to resource R_2 via two synchronisation steps), thereby allocating the p -task to R_2 .

That is, when given two interleaved task expressions \mathcal{T}_1 and \mathcal{T}_2 , we make them the same length by arbitrarily “padding” with empty elements, and then merging the two resulting expressions to form one possible \mathcal{T}' .

We can now define what it means for a recipe to be executable on a topology. We do this with the following operator, which captures what it means to “execute” an arbitrary task expression on the topology \mathbf{P} , given a state \underline{s} and resource vector \mathbf{r} . Intuitively, it returns the set of possible successor states of the topology, together with the corresponding next resource vectors.

In what follows, let $\sigma = (\underline{s}, \mathbf{r})$.

Definition 4 (NXT) We define the operator $\text{NXT}(\sigma, \mathcal{T}) =$

$$\left\{ \begin{array}{l} \bigcup_{\sigma' \in \Sigma} \text{NXT}(\sigma', \mathcal{T}_2) \quad \text{if } \mathcal{T} = \mathcal{T}_1; \mathcal{T}_2 \text{ and } \Sigma = \text{NXT}(\sigma, \mathcal{T}_1); \\ \bigcup_{\mathcal{T}' \in \text{LIN}(\mathcal{T})} \text{NXT}(\sigma, \mathcal{T}') \quad \text{if } \mathcal{T} = \mathcal{T}_1 | \dots | \mathcal{T}_n, n > 1; \\ \Sigma' \bigcup_{\sigma' \in \Sigma} \text{NXT}(\sigma', \mathcal{T}) \quad \text{if } \mathcal{T} = t_1 || \dots || t_n, n \geq 1, \\ \quad (4.1) \quad \Sigma' = \{(\underline{s}', \mathbf{r}') \mid \underline{s} \xrightarrow{t} \underline{s}', \mathbf{t} \in \underline{L}, \\ \quad \mathbf{r}'' \in \text{AL}(\mathbf{r}, \mathcal{T}, \mathbf{t}), \mathbf{r}' = \text{MOV}(\mathbf{r}'', \mathbf{t})\}, \text{ and} \\ \quad (4.2) \quad \Sigma = \{(\underline{s}', \mathbf{r}') \mid \underline{s} \xrightarrow{t} \underline{s}', \mathbf{t} \in \underline{L}, \\ \quad \mathbf{r}' = \text{MOV}(\mathbf{r}, \mathbf{t}), \forall \mathbf{t} \in \mathbf{t}, \mathbf{t} \notin T_{ob}\}. \end{array} \right.$$

In this definition, an element $(\underline{s}', \mathbf{r}') \in \text{NXT}((\underline{s}, \mathbf{r}), \mathcal{T})$ is a “successor” of $(\underline{s}, \mathbf{r})$, where \underline{s}' in the topology is a final state reachable from \underline{s} under guidance from \mathcal{T} , and \mathbf{r}' is the resource vector of \underline{s}' . More specifically,

- if $\mathcal{T} = \mathcal{T}_1; \mathcal{T}_2$, then one task at a time is considered according to the sequence;
- if $\mathcal{T} = \mathcal{T}_1 | \dots | \mathcal{T}_n$, then all linearisations of \mathcal{T} are considered;
- if $\mathcal{T} = t_1 || \dots || t_n$ (including the base case $\mathcal{T} = t_1$), then Σ' is the set of couples $(\underline{s}', \mathbf{r}')$ such that \underline{s}' is a successor of \underline{s} (in the topology) for some label \mathbf{t} , and \mathbf{r}' is the new resource vector obtained from \mathbf{r} by first allocating \mathcal{T} to \mathbf{t} , and then moving parts according to synchronisation tasks in \mathbf{t} . Similarly, Σ is the set of couples $(\underline{s}', \mathbf{r}')$ obtained from $(\underline{s}, \mathbf{r})$ when \mathbf{t} is an “unobservable” transition (one in which no observable tasks occur).

Thus, $\text{NXT}((\underline{s}, \mathbf{r}), \mathcal{T})$ is the set of couples $(\underline{s}', \mathbf{r}')$, where \underline{s}' is reachable from \underline{s} by following a possibly empty sequence of unobservable transitions, followed by an allocation of \mathcal{T} , and \mathbf{r}' is the resulting resource vector. Figure 3 depicts one application of the NXT operator in the context of Figure 1c.

We extend the above notion of executability to traces of a recipe \mathbf{R} as follows. A *trace* $\pi = s_0 \xrightarrow{\mathcal{T}_1} s_1 \xrightarrow{\mathcal{T}_2} \dots \xrightarrow{\mathcal{T}_n} s_n$ of \mathbf{R} is *realisable* in a topology \mathbf{P} iff (a) there exists a sequence of n couples $(\underline{s}_1, \mathbf{r}_1) \dots (\underline{s}_n, \mathbf{r}_n)$ such that $(\underline{s}_j, \mathbf{r}_j) \in \text{NXT}((\underline{s}_{j-1}, \mathbf{r}_{j-1}), \mathcal{T}_j)$ for each $j \in [1, n]$, and (b) $(\underline{s}_0, \mathbf{r}_0) = (\underline{s}^0, \mathbf{r}^0)$. For brevity, we shall write $(\underline{s}_n, \mathbf{r}_n) \in \text{NXT}^*((\underline{s}_0, \mathbf{r}_0), \pi)$. Then, we say that \mathbf{R} is *realisable* in \mathbf{P} iff any trace π of \mathbf{R} (recall that their number is finite) is realisable in \mathbf{P} from its initial state, i.e., $\exists (\underline{s}', \mathbf{r}') \in \text{NXT}^*((\underline{s}^0, \mathbf{r}^0), \pi)$ for any trace π of \mathbf{R} .

Finally, note that the definition of realisability highlights how our notion of a recipe is flexible. Multiple branches existing from a given state constitute a universal requirement (AND-nodes), as we need to be sure that, at run-time, *any* possible trace can be realised. At the same time, task expressions with interleaving represent existential requirements (OR-nodes), as it is sufficient to find a *possible* realisable linearisation. This makes the control problem in our setting more involved than considering any possible trace of a finite-state machine labelled with atomic tasks (as is the case in Behaviour Composition [9]), or the language expressing the set of legal traces of the system (as for Discrete Event Systems [22]).

5.1 Checking Realisability via Simulation

We can now define the notion of a *task simulation relation* for a topology and recipe, inspired by the mathematical notion of simulation [16]. Although this notion is generally applied to infinite processes, its application here to finite recipes allows us to capture the realisability property with respect to the state of the manufacturing facility. Intuitively, it relates states in the topology to states in the recipe with respect to current resource allocations, such that each transition (task expression) in the recipe can be executed by transitions (tasks) in the topology, and the same holds iteratively for the entire recipe, irrespective of the actual recipe trace that might be followed at execution-time (which depends on the outcome of evaluating guards in the recipe).

Definition 5 (Simulation) Let $\mathbf{P} = (\underline{S}^0, \underline{S}, L, \implies)$ be a topology, and $\mathbf{R} = (s^0, S, L, \rightarrow)$ a recipe. A *task simulation relation* is a relation $\text{SIM} \subseteq \underline{S} \times S \times V$,⁶ such that a tuple $(\underline{s}, s, \mathbf{r}) \in \text{SIM}$ implies that for any \mathcal{T} and s' , if $s \xrightarrow{\mathcal{T}} s'$ then there exists a state \underline{s}' and a resource vector \mathbf{r}' such that $(\underline{s}', s', \mathbf{r}') \in \text{SIM}$, with $(\underline{s}', \mathbf{r}') \in \text{NXT}((\underline{s}, \mathbf{r}), \mathcal{T})$.

We say that a state $\underline{s} \in \underline{S}$ *simulates* a state $s \in S$ with respect to a resource vector \mathbf{r} if and only if there exists a task simulation relation SIM such that $(\underline{s}, s, \mathbf{r}) \in \text{SIM}$. Indeed, many such task simulation relations may exist, each accounting for one or more ways of realising the recipe. Finally, topology \mathbf{P} *simulates* recipe \mathbf{R} if and only if there exists a task simulation relation SIM (one is sufficient) such that $(\underline{s}^0, s^0, \mathbf{r}^0) \in \text{SIM}$, where \mathbf{r}^0 denotes the resource vector (with $|\mathbf{r}^0| = |\underline{s}^0|$) composed of empty sequences, namely, the “empty” resource vector.

Theorem 1 (Realisability via simulation) Given a topology $\mathbf{P} = (\underline{S}^0, \underline{S}, L, \implies)$ and recipe $\mathbf{R} = (s^0, S, L, \rightarrow)$, the recipe is realisable in \mathbf{P} iff \mathbf{P} simulates \mathbf{R} .

The proof for this is straightforward, as a task simulation only expresses an *invariant* property with respect to task allocation (the NXT operator). Proceeding by contradiction, suppose \mathbf{R} is realisable in \mathbf{P} but that \mathbf{P} does not simulate \mathbf{R} . By the definition of a task simulation

⁶ Recall that V is the set of all resource vectors.

relation, this means that there exists a trace π in \mathbf{R} such that for any $(\underline{s}', \mathbf{r}') \in \text{NXT}^*((\underline{s}^0, \mathbf{r}^0), \pi)$ we have that $(\underline{s}', \mathbf{r}') \notin \text{SIM}$, with s' being the last state in π . Hence, it can be seen that π (and thus \mathbf{R}) is not realisable in \mathbf{P} . The proof for the opposite direction is similar.

We conclude this section by observing that our notion of task simulation has some similarity with the notion of simulation between transition systems that has been investigated to solve the problem of Behaviour Composition [9]. However there are some fundamental differences. First, the task simulation relation is defined with respect to complex task expressions that can be allocated to resources in parallel, while the notion of a simulation relation applied to behaviour composition assumes that only one behaviour module at the time is allowed to execute actions. Second, each task in the recipe may here be realised by a sequence of observable and unobservable transitions in the topology, which is a more complex setting than the other.

6 CONTROLLER SYNTHESIS

When it is possible to manufacture a product using the given set of resources, that is, when there exists a task simulation relation between the recipe and the topology, the next step is to *synthesise* a controller able to orchestrate the resources in order to execute the recipe and thus manufacture the product.

Crucially, the task simulation relation between a topology \mathbf{P} and recipe \mathbf{R} alone does not hold all the information that is necessary in order to extract solutions: the relation is sufficient to answer whether we *can* orchestrate the resources in order to realise the recipe, but not *how*. For instance, given a tuple $(\underline{s}, \mathbf{r})$ in the task simulation relation, we know, when considering a transition $s \xrightarrow{\mathcal{T}_1|\mathcal{T}_2} s'$ in the recipe, that there must exist a linearisation $\text{LIN}(\mathcal{T}_1|\mathcal{T}_2)$ that is executable on the topology, but we do not know *which*. Indeed, such information is not stored in the relation. The same applies to (atomic) p-tasks: additional, unobservable transitions may be needed before the p-task can be executed, which the relation does not keep track of.

This additional information is the set of sequences of transitions $(\mathbf{t}_1 \cdots \mathbf{t}_k)$ in the topology that realise a task expression \mathcal{T} , for any couple $\sigma = (\underline{s}, \mathbf{r})$ and $\sigma' \in \text{NXT}(\sigma, \mathcal{T})$, where $\sigma' = (\underline{s}', \mathbf{r}')$. We denote this set by $\text{TRANSOF}(\sigma, \mathcal{T}, \sigma')$. It is defined similarly to the operator $\text{NXT}(\sigma, \mathcal{T})$ in Definition 4: an element $(\mathbf{t}_1 \cdots \mathbf{t}_k)$ is in the set $\text{TRANSOF}(\sigma, \mathcal{T}, \sigma')$ if and only if it is one of the sequences of vectors \mathbf{t} corresponding to k recursive applications of steps (4.1) and (4.2) in the definition. Thus, if $(\mathbf{t}_1 \cdots \mathbf{t}_k) \in \text{TRANSOF}(\sigma, \mathcal{T}, \sigma')$, then the trace

$$\underline{s} \xrightarrow{\mathbf{t}_1} \cdots \xrightarrow{\mathbf{t}_k} \underline{s}'$$

is a trace of \mathbf{P} . For example, the following sequence of transitions $(\text{out}1, t, t, t, \text{tr_in}1) \cdot (t, \text{in}2, t, t, \text{tr_out}2) \cdot (t, \text{separate}, t, t, t)$, with $t = \text{nop}$, is in $\text{TRANSOF}(\sigma, \mathcal{T}, \sigma')$, for σ and σ' as in Figure 3. We omit the full definition for brevity, but we make its steps clearer with an algorithm.

Let \mathbf{P} be a production topology and \mathbf{R} a production recipe. Suppose that \mathbf{P} simulates \mathbf{R} , i.e., there exists a task simulation relation SIM between \mathbf{P} and \mathbf{R} . Then, we define a “controller” as a state machine in which a transition encodes a sequence of topology transitions that ought to be executed, given a transition in the recipe.

Definition 6 (Controller) Let $\mathbf{P} = (\underline{s}^0, \underline{S}, \underline{L}, \Longrightarrow)$ and $\mathbf{R} = (s^0, S, L, \rightarrow)$ be as above. A *controller* for \mathbf{R} and \mathbf{P} is a finite state machine $\mathbf{C} = (\text{SIM}, \delta)$, where

- SIM is a non empty task simulation relation, whose elements correspond to the set of states in the controller;

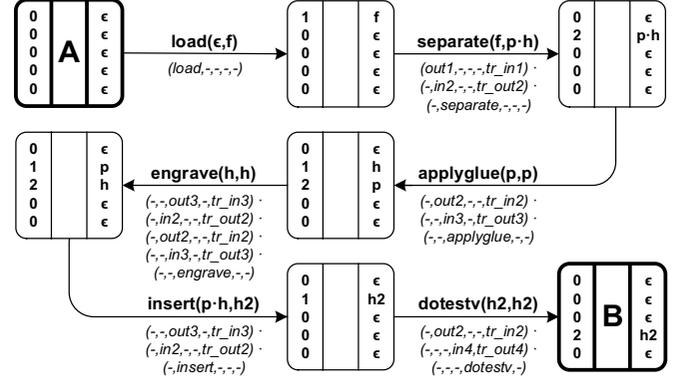


Figure 4: A graphical representation of a single transition in a controller between two tuples $(\underline{s}, s, \mathbf{r}) \in \text{SIM}$ (the first and the last), and therefore states in a possible controller. The states not in bold correspond to “intermediate” steps, namely, applications of the NXT operator to atomic tasks. For instance, the second transition corresponds to $(\underline{s}', s') \in \text{NXT}((\underline{s}, s), \text{separate}(f, p-h))$, as in Figure 3. Therefore, the transition in the controller is labelled with (a) the entire task expression \mathcal{T} between A and B as in Figure 2, and (b) the complete sequence of transitions in the topology that we need to realise \mathcal{T} , as they appear in this figure. Dashes are used in place of *nop* tasks for readability.

- $\delta : \text{SIM} \times \text{Lang}(\mathcal{T}) \times \underline{L}^+ \mapsto \text{SIM}$ is a transition function: given a state, an arbitrary task expression and a sequence of transitions in the topology, it returns the successor state. Specifically, for any state $(\underline{s}', s', \mathbf{r}')$ and task expression \mathcal{T} , if and only if $s \xrightarrow{\mathcal{T}} s'$ for some s' , then there exists at least one transition $(\underline{s}', s', \mathbf{r}') = \delta((\underline{s}, s, \mathbf{r}), \mathcal{T}, (\mathbf{t}_1 \cdots \mathbf{t}_k))$, with $(\underline{s}', \mathbf{r}') \in \text{NXT}((\underline{s}, \mathbf{r}), \mathcal{T})$, and $(\mathbf{t}_1 \cdots \mathbf{t}_k) \in \text{TRANSOF}((\underline{s}, \mathbf{r}), \mathcal{T}, (\underline{s}', \mathbf{r}'))$.

Figure 4 depicts a controller between two tuples in the task simulation relation, in relation to the recipe and resources in Figures 2 and 1c.

We use function $\omega : \text{SIM} \times \text{Lang}(\mathcal{T}) \mapsto 2^{\underline{L}^+}$ in order to “read” the information stored in the controller’s transitions. Given the recipe transition (task expression) that we want to execute, ω defines the set of possible corresponding sequences of transitions in the topology. Formally, $\omega(\theta, \mathcal{T}) = \{(\mathbf{t}_1 \cdots \mathbf{t}_k) \mid \exists \theta' : \theta' = \delta(\theta, \mathcal{T}, (\mathbf{t}_1 \cdots \mathbf{t}_k))\}$.

In Section 5, we defined what it means for a recipe to be realisable on a topology. Similarly, we define here what it means for a recipe to be *executed* on a topology by a controller. First, we say that a trace $\pi = s_0 \xrightarrow{\mathcal{T}_1} s_1 \xrightarrow{\mathcal{T}_2} \cdots \xrightarrow{\mathcal{T}_n} s_n$ of a recipe \mathbf{R} is *executed* on a topology \mathbf{P} by a controller \mathbf{C} if and only if

1. it holds by backward induction that for each $i \in [0, n-1]$, we have $\omega((\underline{s}_i, s_i, \mathbf{r}_i), \mathcal{T}_{i+1}) \neq \emptyset$ with $(\underline{s}_i, s_i, \mathbf{r}_i) = \delta((\underline{s}_{i-1}, s_{i-1}, \mathbf{r}_{i-1}), \mathcal{T}_i, (\mathbf{t}_1 \cdots \mathbf{t}_k))$ for some $(\mathbf{t}_1 \cdots \mathbf{t}_k) \in \omega((\underline{s}_{i-1}, s_{i-1}, \mathbf{r}_{i-1}), \mathcal{T}_i)$;
2. $(\underline{s}_0, s_0, \mathbf{r}_0) = (\underline{s}^0, s^0, \mathbf{r}^0)$ is the initial state.

That is, we first use ω to return the set of transitions in the topology that we may choose in order to realise the current task expression \mathcal{T}_i from the current state $(\underline{s}_i, s_i, \mathbf{r}_i)$ in the controller, and then use δ to compute the next state $(\underline{s}_{i+1}, s_{i+1}, \mathbf{r}_{i+1})$, which corresponds to the new state \underline{s}_{i+1} in the topology, the new state s_{i+1} in the recipe (as it appears in π) and the new allocation \mathbf{r}_{i+1} . This iterative process is repeated at each step of π until the trace is completed, and can

proceed (the set of choices is always non-empty) for any sequence returned by ω . In other words, a trace of the recipe is executed by a controller if, from their respective initial states, the controller can always associate the current task expression in the trace to at least one sequence of transitions in the topology realising that task, thereby orchestrating the resources such that the task expression is realised. A recipe \mathbf{R} is executed on a topology \mathbf{P} by a controller \mathbf{C} iff all its traces are executed on \mathbf{P} by \mathbf{C} .

Theorem 2 *Given \mathbf{R} and \mathbf{P} as above, any controller \mathbf{C} (for \mathbf{R} and \mathbf{P}) is such that \mathbf{R} is executed on \mathbf{P} by \mathbf{C} .*

PROOF. Let us proceed by induction on the length of traces π in \mathbf{R} . If $\pi = s_0$ then the claim holds: point 1 above does not apply because $0 = n$, and point 2 is trivially true. If $\pi = s_0 \xrightarrow{\mathcal{T}_1} s_1 \xrightarrow{\mathcal{T}_2} \dots \xrightarrow{\mathcal{T}_{n-1}} s_{n-1}$ of \mathbf{R} is executed by \mathbf{C} on \mathbf{P} , assume that $\pi \xrightarrow{\mathcal{T}_n} s_n$ is not, i.e., $\omega((s_{n-1}, s_{n-1}, \mathbf{r}_{n-1}), \mathcal{T}_n) = \emptyset$. By definition of ω , this means that there is no transition $(s_n, s_n, \mathbf{r}_n) = \delta((s_{n-1}, s_{n-1}, \mathbf{r}_{n-1}), \mathcal{T}_n, (\mathbf{t}_1 \dots \mathbf{t}_k))$ for any $(\mathbf{t}_1 \dots \mathbf{t}_k)$ and successor (s_n, s_n, \mathbf{r}_n) . By definition of δ this also implies that either there is no (s_n, \mathbf{r}_n) in $\text{NXT}((s_{n-1}, \mathbf{r}_{n-1}))$ or no $(s_n, s_n, \mathbf{r}_n) \in \text{TRANSOF}((s_{n-1}, \mathbf{r}_{n-1}), \mathcal{T}_n, (s_n, \mathbf{r}_n))$. From either case it follows that $(s_n, \mathbf{r}_n) \notin \text{NXT}^*((s_0, \mathbf{r}_0), \pi)$, and thus π is not realisable. Then, by Theorem 1, no (non-empty) task simulation relation exists, and no controller can be defined. \square

We conclude this section with a discussion about complexity.

Theorem 3 (Complexity) *Given \mathbf{R} and \mathbf{P} , checking the existence of (and computing) a controller is polynomial on the size of the topology and exponential in the size of the recipe and number of resources.*

Intuitively, for any atomic expression in the recipe we need to consider any possible combination (s, \mathbf{r}) of a new state in the topology together with any possible new resource vector (corresponding to recursive applications of the definition of NXT) to check whether there exists an allocation of the task (to a transition in the topology) such that \mathbf{r} is the resulting vector. The size of the topology is exponential in the number of resources, and polynomial in their size. The size of all possible resource vectors is exponential in the number of resources and number of parts mentioned in the recipe. As the recipe can be equivalently represented as a rooted tree, if we take the size of the recipe as the maximum length of the sequences of atomic expressions (for any linearisation) appearing along its traces, then computing a simulation relation has cost $O((|\mathbf{P}| \times |\mathbf{V}|)^{|\mathbf{R}|})$. We conclude by noting that we can compute the transitions in the controller (namely, the sequences of transitions in the topology that allow to realise each atomic task) while computing the simulation relation (see points (4.1) and (4.2) in Definition 4, and the discussion before Definition 6).

Our complexity result is consistent with that of [9], which is exponential in the number of available behaviours and polynomial in the size of the so-called enacted system behaviour [9].

7 ALGORITHM FOR REALISABILITY AND SYNTHESIS

We shall now provide an algorithm that computes a task simulation relation, as well as a controller, from a given topology and recipe. The algorithm clarifies how certain steps in our definitions could be implemented. In particular, we show how linearisations can be considered incrementally, without computing them all at the outset; how

the closure of unobservable transitions are computed; how termination is ensured; and finally, how controller transitions are gathered.

The algorithms use some additional auxiliary notions, which we define first. Given a possibly atomic task expression $\mathcal{T} = \mathcal{T}_1; \mathcal{T}_2; \dots; \mathcal{T}_n, n > 0$, we define its first element as $\text{FST}(\mathcal{T}) = \mathcal{T}_1$ (where \mathcal{T}_1 is either atomic or an interleaved composition), and the rest of its elements as $\text{RST}(\mathcal{T}) = \mathcal{T}_2; \dots; \mathcal{T}_n$ if $n > 1$, and as $\text{RST}(\mathcal{T}) = \epsilon$ if $n = 1$. We also extend these notions as follows. Given an interleaved composition $\mathcal{T} = \mathcal{T}_1 | \dots | \mathcal{T}_n, n > 1$, and an atomic task expression \mathcal{T}_a , we define the following:

$$\begin{aligned} \overline{\text{FST}}(\mathcal{T}) &= \{\mathcal{T}_1 : \mathcal{T}_i = \mathcal{T}_1; \dots; \mathcal{T}_m, \mathcal{T}_1 \text{ is atomic}\}, \\ \overline{\text{RST}}(\mathcal{T}, \mathcal{T}_a) &= \{\mathcal{T}_1 | \dots | \mathcal{T}_{i-1} | \mathcal{T}_{i+1} | \dots | \mathcal{T}_n : \mathcal{T}_i = \mathcal{T}_a\} \cup \\ &\quad \{\mathcal{T}_1 | \dots | \mathcal{T}_{i-1} | \mathcal{T}' | \mathcal{T}_{i+1} | \dots | \mathcal{T}_n : \mathcal{T}_i = \mathcal{T}_a; \mathcal{T}'\}, \\ \text{FSTRST}(\mathcal{T}) &= \{\mathcal{T}_1; \mathcal{T}_2 : \mathcal{T}_1 \in \overline{\text{FST}}(\mathcal{T}), \mathcal{T}_2 \in \overline{\text{RST}}(\mathcal{T}, \mathcal{T}_1)\}, \end{aligned}$$

where \mathcal{T}' is any non-empty sequence. These notions define the first elements of interleaved composition \mathcal{T} as the set of all first elements in each of its sequences \mathcal{T}_i , and the rest of \mathcal{T} (relative to some first element \mathcal{T}_a) as the set of all interleaved compositions obtained from \mathcal{T} by removing \mathcal{T}_a (once) if it occurs first in some \mathcal{T}_i .

Given a topology and recipe, Algorithms 1 and 2 work as follows. Algorithm 1 additionally takes a state s in the recipe and a couple $\sigma = (s, \mathbf{r})$ as input, where s is a state in the topology and \mathbf{r} is a resource vector. Then, for each outgoing transition in the recipe, from s to some s' , the algorithm checks whether the associated task expression \mathcal{T} can be simulated from the corresponding topology state s , and (recursively) whether the same holds for each outgoing transition from s' . Thus, s' is passed as a parameter to Algorithm 2 in order to “remember” to continue from s' . Whenever the pair s and s are indeed in simulation, they are added to the set SIM in line 6, together with the corresponding resource vector \mathbf{r} . Observe that this is done in a post-order (depth-first) fashion, from the “deepest” states in the recipe to the state that was passed into Algorithm 1 when it was first called. For controller synthesis, the algorithm also passes the entire recipe transition being considered and its corresponding couple σ as respectively the third and fifth parameters (line 3) to Algorithm 2.

Intuitively, Algorithm 2 implements the NXT operator in Definition 4. Basically, the algorithm performs a depth-first traversal of the topology to check whether task expression \mathcal{T}_{cur} (initially \mathcal{T}) can be simulated. We highlight the important lines below. Line 6 looks for a transition in the topology from state s_{next} that matches the first element in \mathcal{T}_{cur} , provided that the couple σ_{next} was not already explored in a previous (recursive) step. Lines 11 and 12 incrementally check each linearisation of an interleaved composition \mathcal{T}_{cur} , by recursively checking the concatenation of each first element in \mathcal{T}_{cur} , with a corresponding remainder. The loop exits as soon as any viable linearisation is found. Line 14 is only applicable if \mathcal{T}_{cur} could not be simulated so far, and the topology-transition being considered is an unobservable one. In this case, the transition is “skipped”, and a simulation of \mathcal{T}_{cur} is tried from the next state s' (with its corresponding resource allocation). Line 2 is only applicable when \mathcal{T}_{cur} has been completely “processed”; it is at this point that a transition x is added to the controller’s transition function δ (Definition 6). Finally, line 16 keeps track of the fact that σ_{next} is being explored, in order to guarantee termination. Combined with the fact that the recipe is acyclic, the following results hold trivially.

Proposition 4 *Algorithm 1 always terminates. Furthermore, it is optimal with respect to computational complexity.*

Observe that line 16 (and other lines) in Algorithm 2 adds the current transition label \mathbf{t} to the end of the sequence of topology

Algorithm 1 FindSim(σ, s)

Input: Topology ($\underline{s}^0, \underline{S}, \underline{L}, \implies$) and recipe (s^0, S, L, \rightarrow);
the couple $\sigma = (\underline{s}, \mathbf{r})$; state $s \in S$.

Output: Either (\emptyset, \emptyset) or a non-empty relation SIM and function δ .

- 1: $\delta, \text{SIM}, \text{SIM}' := \emptyset$
- 2: **for** each $(s, \mathcal{T}, s') \in \rightarrow$ **do**
- 3: $(\text{SIM}', \delta) := \text{EvalExp}(\sigma, \mathcal{T}, (s, \mathcal{T}, s'), \epsilon, \sigma, \emptyset)$
- 4: **if** $\text{SIM}' = \emptyset$ **then return** (\emptyset, \emptyset)
- 5: $\text{SIM} := \text{SIM} \cup \text{SIM}'$
- 6: **return** $(\text{SIM} \cup \{(\underline{s}, s, \mathbf{r})\}, \delta)$

Algorithm 2 EvalExp($\sigma_{next}, \mathcal{T}_{cur}, tr, (\mathbf{t}_1 \cdots \mathbf{t}_k), \sigma_0, \Sigma$)

Input: Topology ($\underline{s}^0, \underline{S}, \underline{L}, \implies$) and recipe (s^0, S, L, \rightarrow);
current successor couple $\sigma_{next} = (\underline{s}_{next}, \mathbf{r}_{next})$;
task expression being processed \mathcal{T}_{cur} ;
current recipe-transition $tr = (s, \mathcal{T}, s_{next})$;
current sequence of topology-transition labels $(\mathbf{t}_1 \cdots \mathbf{t}_k)$;
couple $\sigma_0 = (\underline{s}_0, \mathbf{r}_0)$, corresponding to s ; visited couples Σ .

Output: Either (\emptyset, \emptyset) or a non-empty relation SIM and function δ .

- 1: $\delta, \text{SIM} := \emptyset$
- 2: **if** $\mathcal{T}_{cur} = \epsilon$ **then**
- 3: $x := (\underline{s}_0, s, \mathbf{r}_0) \xrightarrow{\mathcal{T}, (\mathbf{t}_1 \cdots \mathbf{t}_k)} (\underline{s}_{next}, s_{next}, \mathbf{r}_{next})$
- 4: $(\text{SIM}, \delta) := \text{FindSim}(\sigma_{next}, s_{next})$
- 5: **return** $(\text{SIM}, \delta \cup \{x\})$
- 6: **for** each $(\underline{s}_{next}, \mathbf{t}, \underline{s}') \in \implies$ with $\sigma_{next} \notin \Sigma$ **do**
- 7: **if** $\text{FST}(\mathcal{T}_{cur}) = \mathbf{t}_1 \parallel \dots \parallel \mathbf{t}_n$ and
 $\mathbf{r}'' \in \text{AL}(\mathbf{r}_{next}, \text{FST}(\mathcal{T}_{cur}), \mathbf{t})$ **then**
- 8: $\sigma' := (\underline{s}', \mathbf{r}')$, where $\mathbf{r}' = \text{MOV}(\mathbf{r}'', \mathbf{t})$
- 9: $(\text{SIM}, \delta) :=$
 $\text{EvalExp}(\sigma', \text{RST}(\mathcal{T}_{cur}), tr, (\mathbf{t}_1 \cdots \mathbf{t}_k \cdot \mathbf{t}), \sigma_0, \emptyset)$
- 10: **else if** $\text{FST}(\mathcal{T}_{cur}) = \mathcal{T}_1 \mid \dots \mid \mathcal{T}_n$ (where $n > 1$) **then**
- 11: **for** each $\mathcal{T}' \in \text{FSTRST}(\mathcal{T}_{cur})$ **do**
- 12: $(\text{SIM}, \delta) := \text{EvalExp}(\sigma_{next}, \mathcal{T}', tr, (\mathbf{t}_1 \cdots \mathbf{t}_k), \sigma_0, \emptyset)$
- 13: **if** $\text{SIM} \neq \emptyset$ **then return** (SIM, δ)
- 14: **if** $\text{SIM} = \emptyset$ and $\forall t \in \mathbf{t}, t \notin T_{ob}$ **then**
- 15: $\sigma' := (\underline{s}', \mathbf{r}')$, where $\mathbf{r}' = \text{MOV}(\mathbf{r}_{next}, \mathbf{t})$
- 16: $(\text{SIM}, \delta) :=$
 $\text{EvalExp}(\sigma', \mathcal{T}_{cur}, tr, (\mathbf{t}_1 \cdots \mathbf{t}_k \cdot \mathbf{t}), \sigma_0, \Sigma \cup \{\sigma_{next}\})$
- 17: **if** $\text{SIM} \neq \emptyset$ **then return** (SIM, δ)
- 18: **return** (\emptyset, \emptyset)

transitions $\mathbf{t}_1 \cdots \mathbf{t}_k$ being pursued. This essentially implements the TRANSOF operator described in Section 6, and together with line 3, leads to the fact that the algorithm correctly computes a controller for the recipe and topology. Moreover, the algorithm also correctly computes a task simulation relation for them.

Theorem 5 (Correctness) Let $\mathbf{P} = (\underline{s}^0, \underline{S}, \underline{L}, \implies)$ be a topology and $\mathbf{R} = (s^0, S, L, \rightarrow)$ a recipe. Let $(\text{SIM}, \delta) = \text{FindSim}(\sigma, s^0)$ with $\sigma = (\underline{s}^0, \mathbf{r}^0)$. Then (a) $\text{SIM} = \emptyset$ iff \mathbf{R} is not realisable in \mathbf{P} ; otherwise, (b) SIM is a task simulation relation between \mathbf{P} and \mathbf{R} ; and (c) (SIM, δ) is a controller of \mathbf{P} and \mathbf{R} .

PROOF SKETCH. This is an involved proof by induction on the ‘‘height’’ h of vertices in the recipe. The main part involves showing that SIM is a task simulation relation between \mathbf{P} and \mathbf{R} . For the base case, we take $h = 0$. Then, there is no recipe transition $(s, \mathcal{T}, s') \in \rightarrow$ for any \mathcal{T} (line 2 in Algorithm 1). Thus, the algorithm returns $\text{SIM} = \{(\underline{s}, s, \mathbf{r})\}$, and $(\underline{s}, s, \mathbf{r}) \in \text{SIM}$ also holds by Definition 5. Next, we assume that the theorem holds if $h \geq x$, for some $x \in \mathbb{N}_0$ (induction hypothesis). For the inductive step, we take $h = x + 1$. Then, there must exist a recipe transition $(s, \mathcal{T}, s') \in \rightarrow$ for some \mathcal{T} . We prove that if $\text{SIM}' \neq \emptyset$ in line 3, then SIM' is ‘‘sound’’.

We do this by induction on the length of expression \mathcal{T}_{cur} in Algorithm 2. The main point is that the couples in Σ' and Σ in steps (4.1) and (4.2) in Definition 4 are the same as in lines 8 and 15 in Algorithm 2, respectively. Finally, if $\text{SIM}' \neq \emptyset$ in line 3 of Algorithm 1 for all transitions considered in line 2, it follows that $(\underline{s}, s, \mathbf{r}) \in \text{SIM}$ also holds by the induction hypothesis and Definition 5. \square

8 DISCUSSION AND FUTURE WORK

In this paper we presented an approach for modelling a flexible manufacturing system and checking whether it is possible to manufacture a given product in the system, represented as a manufacturing recipe. This check is done by computing a task simulation relation as discussed in Section 5. Further, we presented an approach for synthesising a controller that is able to orchestrate the manufacturing resources in the system so as to realise the recipe.

However, we did not address the problem of synthesising *any* controller for the given recipe. Rather, (a) we computed a *possible* task simulation relation, and (b) defined a *possible* controller. Moreover, the algorithm presented in Section 7 returns a controller which represents *exactly one* way of orchestrating the resources to realise the recipe, as this is sufficient for our application.

One direction for future research therefore involves the synthesis of *any* controller for a given recipe, and the technical content of this paper already allows for this extension. Computing any possible controller amounts to (a) computing the *largest* task simulation relation, and (b) including in the controller, between two states $(\underline{s}, s, \mathbf{r})$ and $(\underline{s}', s', \mathbf{r}')$, all possible sequences of transitions defined by $\text{TRANSOF}((\underline{s}, \mathbf{r}), \mathcal{T}, (\underline{s}', \mathbf{r}'))$ and not just at least one, as in Definition 6. An algorithm can be devised accordingly.

Another research direction involves checking realisability (and synthesising suitable controllers) *offline*, but for *classes* of recipes instead of single recipes. A class of recipes can be expressed in terms of a union of recipes, but also as unbounded (even cyclic) processes, which is already handled by our notion of task simulation, as it relies on the standard mathematical notion of simulation. To check the realisability of a new recipe, it would therefore be sufficient to check whether it can be simulated (task simulation) by one such class, which is a polynomial check in the size of the recipe.

Finally, although our current formalism allows only parallel tasks on *different* parts, we plan to extend it to capture parallel tasks operating on *the same* part (e.g. two machines performing a joint task on a single part). Allowing parallel tasks on the same part requires machines to be able to ‘‘see’’ the workspaces of each other, thus increasing the complexity of the algorithm proportionately.

ACKNOWLEDGEMENTS

This research was funded by EPSRC grants EP/K018205/1 and EP/K014161/1, the support of which is gratefully acknowledged. We would like to thank Natasha Alechina for many helpful ideas and discussions relating to the work presented here, and Nikolas Antzoulatos and Elkin Castro for useful advice and material.

REFERENCES

- [1] ANSI/ISA-95, Enterprise-Control System Integration, Parts 1-5.
- [2] Business To Manufacturing Markup Language - Operations Schedule - Version 6.0.
- [3] IEC 62264-1: Enterprise-control system integration - Parts 1-3.
- [4] ‘A Landscape for the Future of High Value Manufacturing in the UK’, Technical report, Technology Strategy Board, (2012).

- [5] Zhuming M. Bi, Sherman Y.T. Lang, Weiming Shen, and Lihui Wang, 'Reconfigurable manufacturing systems: the state of the art', *International Journal of Production Research*, **46**(4), 967–992, (2008).
- [6] Simon Bliudze and Joseph Sifakis, 'The algebra of connectors: Structuring interaction in BIP', in *Proceedings of the ACM IEEE International Conference on Embedded Software*, EMSOFT '07, pp. 11–20, (2007).
- [7] Jim Browne, Didier Dubois, Keith Rathmill, Suresh P. Sethi, and Kathryn E. Stecke, 'Classification of flexible manufacturing systems', *The FMS magazine*, **2**(2), 114–117, (1984).
- [8] J.C. Chaplin, O.J. Bakker, L. de Silva, D. Sanderson, E. Kelly, B. Logan, and S.M. Ratchev, 'Evolvable assembly systems: A distributed architecture for intelligent manufacturing', *IFAC-PapersOnLine*, **48**(3), 2065–2070, (2015).
- [9] Giuseppe De Giacomo, Fabio Patrizi, and Sebastian Sardina, 'Automatic behavior composition synthesis', *AIJ*, **196**, 106–142, (2013).
- [10] Hoda A. ElMaraghy, 'Flexible and reconfigurable manufacturing systems paradigms', *International Journal of Flexible Manufacturing Systems*, **17**(4), 261–276, (2005).
- [11] Kutluhan Erol, James Hendler, and Dana S. Nau, 'HTN planning: Complexity and expressivity', in *Proceedings of the National Conference on Artificial Intelligence (AAAI-94)*.
- [12] Henning Kagermann, Johannes Helbig, Ariane Hellinger, and Wolfgang Wahlster, *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry; Final Report of the Industrie 4.0 Working Group*, Forschungunion, 2013.
- [13] Yoram Koren, Uwe Heisel, Francesco Jovane, Toshimichi Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, 'Reconfigurable manufacturing systems', *CIRP Annals-Manufacturing Technology*, **48**(2), 527–540, (1999).
- [14] Yuqian Lu, Xun Xu, and Jenny Xu, 'Development of a hybrid manufacturing cloud', *Journal of Manufacturing Systems*, **33**(4), 551–566, (2014).
- [15] Mostafa G. Mehrabi, A. Galip Ulsoy, and Yoram Koren, 'Reconfigurable manufacturing systems: key to future manufacturing', *Journal of Intelligent Manufacturing*, **11**(4), 403–419, (2000).
- [16] Robin Milner, 'An algebraic definition of simulation between programs', Technical report, Stanford University, (1971).
- [17] Tadao Murata, 'Petri nets: Properties, analysis and applications', *Proceedings of the IEEE*, **77**(4), 541–580, (1989).
- [18] Anand S. Rao, 'AgentSpeak(L): BDI agents speak out in a logical computable language', in *Proceedings of the European workshop on Modelling Autonomous Agents in a Multi-Agent World : agents breaking away*, pp. 42–55, (1996).
- [19] Chris Rhodes, *Manufacturing: Statistics and Policy. Briefing Paper*, House of Commons Library, 2015.
- [20] Andrea Krasa Sethi and Suresh Pal Sethi, 'Flexibility in manufacturing: a survey', *International Journal of Flexible Manufacturing Systems*, **2**(4), 289–328, (1990).
- [21] Daniel Smale and Svetan Ratchev, 'A capability model and taxonomy for multiple assembly system reconfigurations', in *Proceedings of IFAC Symposium on Information Control Problems in Manufacturing (INCOM)*, volume 13, pp. 1923–1928, (2009).
- [22] W. Murray Wonham and Peter J. Ramadge, 'On the supremal controllable sub-language of a given language', *SIAMJCO*, **25**(3), 637–659, (1987).

Towards Lifelong Object Learning by Integrating Situated Robot Perception and Semantic Web Mining

Jay Young¹ and Valerio Basile² and Lars Kunze¹ and Elena Cabrio² and Nick Hawes¹

Abstract.

Autonomous robots that are to assist humans in their daily lives are required, among other things, to recognize and understand the meaning of task-related objects. However, given an open-ended set of tasks, the set of everyday objects that robots will encounter during their lifetime is not foreseeable. That is, robots have to learn and extend their knowledge about previously unknown objects on-the-job. Our approach automatically acquires parts of this knowledge (e.g., the class of an object and its typical location) in form of ranked hypotheses from the Semantic Web using contextual information extracted from observations and experiences made by robots. Thus, by integrating situated robot perception and Semantic Web mining, robots can continuously extend their object knowledge beyond perceptual models which allows them to reason about task-related objects, e.g., when searching for them, robots can infer the most likely object locations. An evaluation of the integrated system on long-term data from real office observations, demonstrates that generated hypotheses can effectively constrain the meaning of objects. Hence, we believe that the proposed system can be an essential component in a lifelong learning framework which acquires knowledge about objects from real world observations.

1 Introduction

It is crucial for autonomous robots working in human environments such as homes, offices or factories to have the ability to represent, reason about, and learn new information about the objects in their environment. Current robot perception systems must be provided with models of the objects in advance, and their extensibility is typically poor. This includes both perceptual models (used to recognise the object in the environment) and semantic models (describing what the object is, what it is used for etc.). Equipping a robot *a priori* with a (necessarily closed) database of object knowledge is problematic because the system designer must predict which subset of all the different domain objects is required, and then build all of these models (a time-consuming task). If a new object appears in the environment, or an unmodelled object becomes important to a task, the robot will be unable to perceive, or reason about, it. The solution to this problem is for the robot to *learn on-line about previously unknown objects*. This allows robots to autonomously extend their knowledge of the environment, training new models from their own experiences and observations.

The online learning of perceptual and semantic object models is a major challenge for the integration of robotics and AI. In this paper we address one problem from this larger challenge: given an observation of a scene containing an unknown object, can an autonomous system predict the semantic description of this object. This is an important problem because online-learned object models ([5]) must be integrated into the robot's existing knowledge base, and a structured, semantic description of the object is crucial to this. Our solution combines semantic descriptions of perceived scenes containing unknown objects, with a distributional semantic approach which allows us to fill gaps in the scene descriptions by mining knowledge from the Semantic Web. Our approach assumes that the knowledge onboard the robot is a subset of some larger knowledge base, i.e. that the object is not unknown beyond the robot's pre-configured knowledge. To determine which concepts from this larger knowledge base might apply to the unknown object, our approach exploits the spatio-temporal context in which objects appear, e.g. a teacup is often found next to a teapot and sugar bowl. These spatio-temporal co-occurrences provide contextual clues to the properties and identity of otherwise unknown objects.

This paper makes the following contributions:

- a novel distributional semantics-based approach for predicting both the semantic identity of an unknown, everyday object based on its spatial context and its most likely location based on semantic relatedness;
- an extension to an existing semantic perception architecture to provide this spatial context; and
- an evaluation of these techniques on real-world scenes gathered from a long-term autonomous robot deployment.

The remainder of the paper is structured as follows. In Section 2, we first state the problem of acquiring semantic descriptions for unknown objects and give an overview of our approach. We then discuss related work in Section 3. In Section 4, we describe the underlying robot perception system and explain how it is integrated with a Semantic Web mining component. Section 5 describes how the component generates answers/hypotheses to web-queries from the perception module. In Section 6, we describe the experimental setup and present the results. Before we conclude in Section 8, we provide a detailed discussion about our approach in Section 7. We also make available our data set and software source code for the benefit of the community at: <http://github.com/alooof-project/>

¹ Intelligent Robotics Lab, School of Computer Science, University of Birmingham, United Kingdom, {j.young, l.kunze, n.a.hawes}@cs.bham.ac.uk

² INRIA Sophia Antipolis Méditerranée, Sophia Antipolis, France, {valerio.basile, elena.cabrio}@inria.fr



Room	kitchen
Surface	countertop
Furniture	refrigerator, kitchen cabinet, sink
Small Objects	bowl, teabox, instant coffee, water boiler, mug

Figure 1. Perceived and interpreted kitchen scene, with various objects.

2 Problem Statement and Methodology

2.1 Problem Statement

The problem we consider in this work can be summarized as follows: *Given the context of a perceived scene and the experience from previous observations, predict the class of an as 'unknown' identified object.* The context of a scene can include information about the types and locations of recognized small objects, furniture, and the type of the room where the observation has been made.

In this paper we use the following running example (Figure 1) to illustrate the problem and our approach:

While operating 24/7 in an office environment, a robot routinely visits the kitchen and scans all surfaces for objects. On a kitchen counter it finds several household objects: a bowl, a teabox, a box of instant coffee, and a water boiler. However, one of the segmented objects, a mug, cannot be identified as one of the known object classes. The robot's task is to identify the unknown object solely based on the context of the perceived scene and scenes that have been previously perceived and in which the respective object was identified.

The problem of predicting the class of an object purely based on the context can also be seen as *top-down reasoning* or *top-down processing* of information. This stands in contrast to data-driven bottom-up processing where, for example, a robot tries to recognize an object based on its sensor data. In top-down processing, an agent, or the robot, has some expectations of what it will perceive based on commonsense knowledge and its experiences. For example, if a robot sees a fork and a knife close to each other, and a flat unknown object with a square bounding box next to them, it might deduce that

the unknown object is probably a plate. In the following, we refer to this kind of processing which combines top-down reasoning and bottom-up perception as *knowledge-enabled perception*.

Systems such as the one described in this paper are key components of integrated, situated AI systems intended for life-long learning and extensibility. We currently develop the system with two main use-cases in mind, both stemming from the system's capability to suggest information about unknown objects based on the spatial context in which they appear. The first use case is as part of a crowdsourcing platform, allowing humans that inhabit the robot's environment to help it label unknown objects. Here, the prediction system is used to narrow down the list of candidate labels and categories to be shown to users to select from alongside images of unknown objects the robot has encountered. Our second use case will be to help form more informative queries for larger machine learning systems, in our case an image classification system trained on extensive, though categorised, image data from websites like Amazon. Here, having some hints as to an object's identity, such as a distribution over a set of possible labels or categories it might belong to or be related to, could produce a significant speed boost by letting the classification system know what objects it does *not* have to test against. In this case, we aim to use the system to help a robot make smarter, more informed queries when asking external systems questions about the world.

2.2 Our Approach

In this work we address the problem of predicting information about the class of an object based on the perceived scene context by mining the Semantic Web. The extracted scene context includes a list of recognized objects and their spatial relations among each other, plus additional information from a semantic environment map. This information is then used to mine potential object classes based on the semantic relatedness of concepts in the Web. In particular, we use DBpedia as a resource for object knowledge, and will later on use WordNet to investigate object taxonomies. The result of the web mining component is a ranked list of potential objects classes, expressed as DBpedia entries, which allows us access to further information beyond just the class of an object, such as categorical knowledge. An overview of the entire developed system is given in Figure 2.

Overall, we see our context-based class prediction approach as a means to restrict the number of applicable classes for an object. The aim of our knowledge-enabled perception system is not to replace a bottom-up perception system but rather to complement it as an additional *expert*. For example, in the context of a crowdsourcing-based labeling platform our system could generate label suggestions for users. Thereby labeling tasks can be performed in less time and object labels would be more consistent across users. Hence, we believe that our system provides an essential functionality in the context of lifelong object learning.

Before we present related work in Section 3 we briefly discuss various resources of object knowledge.

Resources for object knowledge To provide a common format for object knowledge, and to access the wide variety of structured knowledge available on the Web, we link the observations made by the robot to DBpedia concepts. DBpedia [2] is a crowd-sourced community effort started by the Semantic Web community to extract structured information from Wikipedia and make this information available on the Web. DBpedia has a broad scope of entities covering different domains of human knowledge: it contains more than 4 million things classified in a consistent ontology and denoted by a URI-based

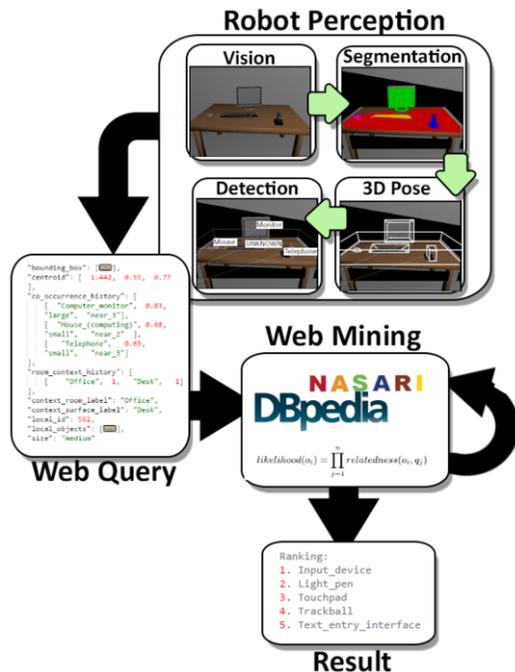


Figure 2. System overview. The robot perception component identifies all object candidates within a scene. All object candidates that can be recognized are labeled according their class, all remaining objects are labeled as 'unknown'. Furthermore, the component computes the spatial relations between all objects in the scene. Together with context information from a semantic environment map, the robot generates a query to a web service which is processed by the Semantic Web mining component. Based on the semantic relatedness of objects the component provides a ranked list of the potential classes for all unknown objects.

reference of the form <http://dbpedia.org/page/Teapot> for the Teapot concept. DBpedia supports sophisticated queries (using an SQL-like query language for RDF called SPARQL) to mine relationships and properties associated with Wikipedia resources. We link the objects that the robot can encounter in natural environments to DBpedia concepts, thus exploiting this structured, ontological knowledge.

BabelNet [16] is both a multilingual encyclopedic dictionary and a semantic network which connects concepts and named entities in a very large network of semantic relations (about 14 million entries). BabelNet covers and is obtained from the automatic integration of several resources, such as WordNet [6], Wiktionary and Wikipedia. Each concept contained in BabelNet is represented as a vector in a high-dimensional geometric space in the NASARI resource, that we use to compute the semantic relatedness among objects.

3 Related Work

To obtain information about unknown objects from the Web, a robot can use perceptual or knowledge-based queries. Future systems will inevitably need to use both. In this paper we focus on the knowledge-based approach, but this can be seen as complementary to systems which use image-based queries to search databases of labelled images for similarity, e.g. [17].

Although the online learning of new *visual* object models is currently a niche area in robotics, some approaches do exist [5, 7]. These

approaches are capable of segmenting previously unknown objects in a scene and building models to support their future re-recognition. However, this work focuses purely on visual models (what objects look like), and does not address how the learnt objects are described semantically (what objects are).

The RoboSherlock framework [1] (which we build upon, see Section 4.1) is one of the most prominent projects to add semantic descriptions to objects for everyday environments, but the framework must largely be configured *a priori* with knowledge of the objects in its environment. It does support more open ended performance, e.g. through the use of Google Goggles, but does not use spatial or semantic context for its Web queries, only vision. The same research group pioneered Web and cloud robotics, where tools such as KNOWROB [20] (also used in RoboSherlock) both formalised robot knowledge and capabilities, and used this formal structure to exploit the Web for remote data sources and knowledge sharing. In a more supervised setting, many approaches have used humans to train mobile robots about new objects in their environment [9, 19] and robots have also used Web knowledge sources to improve their performance in closed worlds, e.g. the use of object-room co-occurrence data for room categorisation in [10].

The spatial organisation of a robot's environment has also been previously exploited to improve task performance. For example, [21, 12] present a system in which the previous experience of spatial arrangements of desktop objects is used to refine the results of a noisy object categorisation system. This demonstrates the predictive power of spatial arrangements, which is something we also exploit in this paper. However this prior work matched between scenes in the same environment and input modality. In our work we connect spatial arrangements in the robot's situated experience to structured knowledge on the Web.

Our predictions for unknown objects rely on determining the semantic relatedness of terms. This is an important topic in several areas, including data mining, information retrieval and web recommendation. [18] applies ontology-based similarity measures in the robotics domain. Background knowledge about all the objects the robot could encounter, is stored in an extended version of the KNOWROB ontology. Then, WUP similarity [22] is applied to calculate relatedness of the concept types by considering the depth of the concepts and the depth of their lowest common super-concept in the ontology. [14] presents an approach for computing the semantic relatedness of terms using ontological information extracted from DBpedia for a given domain, using the results for music recommendations. Contrary to these approaches, we compute the semantic relatedness between objects by leveraging the vectorial representation of the DBpedia concepts provided by the NASARI resource [3]. This method links back to earlier distributional semantics work (e.g. Latent Semantic Analysis [13]) with the difference that here concepts are represented as vectors, rather than words.

4 Situated Robot Perception

4.1 The RoboSherlock Framework

To be able to detect both known and unknown objects in its environment a robot must have perceptual capabilities. Our perception pipeline is based on the *RoboSherlock framework* [1], an open-source framework for implementing perception systems for robots, geared towards interaction with objects in human environments. The use of RoboSherlock provides us with a suite of vision and perception algorithms. Following the paradigm of Unstructured Information Man-

agement (as used by the IBM Watson project), RoboSherlock approaches perception as a problem of content analysis, whereby sensor data is processed by a set of specialised information extraction and processing algorithms called *annotators*. The RoboSherlock perception pipeline is a sequence of annotators which include plane segmentation, RGB-D object segmentation, and object detection algorithms. The output of the pipeline includes 3D point clusters, bounding boxes of segmented objects (as seen in Figure 2), and feature vectors (colour, 3D shape and texture) describing each object. These feature vectors are important as they allow the robot to track unknown objects as it takes multiple views of the same scene. Though in this paper we work with a collected and annotated dataset, we do not require the segmentation or 3D object recognition steps RoboSherlock can provide via LINE-MOD-3D [11], though this component is used in our full Robot and Simulated system where a range of perception algorithms are connected and used instead of dataset input. We make use of all other RoboSherlock capabilities the pipeline to process the data and provide a general architecture for our representation and extraction of historical spatial context, web query generation and the application of Qualitative Spatial Relations, which we will discuss in a following section.

4.2 Scene Perception

In this paper we assume the robot is tasked with observing objects in natural environments. Whilst this is not a service robot task in itself, it is a precursor to many other task-driven capabilities such as object search, manipulation, human-robot interaction etc. Similar to prior work (e.g. [18]) we assume that the robot already has a semantic map of its environment which provides it with at least annotations of supporting surfaces (desks, worktops, shelves etc.), plus the semantic category of the area in which the surface is located (office, kitchen, meeting room etc.). Surfaces and locations are linked to DBpedia entries just as object labels are, typically as entities under the categories Furniture and Room respectively.

From here, we have access to object, surface and furniture labels described by the data, along with 3D bounding boxes via 3D point data. In the kitchen scene the robot may observe various objects typical of the room, such as a refrigerator, a cabinet, mugs, sugar bowls or coffee tins. Their positions in space relative to a global map frame are recorded and we can then record the distance between objects, estimate their size (volume) and record information about their co-occurrences, and the surfaces upon which they were observed, by updating histograms attached to each object.

In the following we assume that each scene only contains a single unknown object, but the approach generalises to multiple unknown objects treated independently. Joint inference over multiple unknown objects is future work.

4.3 Spatial and Semantic Context Extraction

In order to provide additional information to help subsequent components predict the unknown object, we augment the scene description with additional spatial and semantic *context* information, describing the relationships between the unknown object and the surrounding known objects and furniture. This context starts from the knowledge we already have in the semantic map: labels for the room and surface the object is supported by.

We make use of *Qualitative Spatial Relations* (QSRs) to represent information about objects [8]. QSRs discretise continuous spatial measurements, particularly relational information such as the dis-

tance and orientation between points, yielding symbolic representations of ranges of possible continuous values. In this work, we make use of a qualitative distance measure, often called a Ring calculus. When observing an object, we categorise its distance relationship with any other objects in a scene with the following set of symbols: $near_0, near_1, near_2$, where $near_0$ is the closest. This is accomplished by placing sets of thresholds on the distance function between objects, taken from the centroid of the 3D cluster. For example, this allows us to represent that the mug is closer to the spoon than the kettle ($near_0(mug, spoon) near_2(mug, kettle)$) without using floating-point distance values based on noisy and unreliable readings from the robot's sensors. The RoboSherlock framework provides a measure of the qualitative size of objects by thresholding the values associated with the volume of 3D bounding-boxes around objects as they are observed. We categorise objects as *small, medium, large* in this way, allowing the robot to represent and compare object sizes. Whilst our symbolic abstractions are currently based on manual thresholds, approaches exist for learning parametrisations of QSRs through experience (e.g. [23]) and we will try this in the future. For now, we choose parameters for our qualitative calculi tuned by our own knowledge of objects in the world, and how they might relate. We use $near_0$ for distances in cluster space lower than 0.5, $near_1$ for distances between 0.5 and 1.0, $near_2$ for distances between 1.0 and 3.5 and $near_3$ for distances greater than 3.5.

As the robot makes subsequent observations, it may re-identify the same unknown object in additional scenes. When this happens we store all the scene descriptions together, providing additional context descriptions for the same object. In Figure 3 we show part of the data structure describing the objects that co-occured with a plate in a kitchen, and their *most common* qualitative spatial relations.

```

1 "co_occurrences": [
2   ["Coffee", 0.5, "near_0" ],
3   ["Kitchen_side", 1.0, "near_0" ],
4   ["Kitchen_cabinet", 1.0, "near_1" ],
5   ["Fridge", 0.625, "near_1" ],
6   ["Teabox", 0.625, "near_0" ],
7   ["Waste_container", 0.375, "near_2" ],
8   ["Utensil_rack", 0.625, "near_1" ],
9   ["Sugar_bowl_(dishware)", 0.625, "near_0" ]
10  ],
11  "context_history": [
12    ["Kitchen", 1.0, "Kitchen_counter", 1 ],
13    [ "Office", 0.0, "Desk", 0 ]],
14  "context_room_label": "Kitchen",
15  "context_surface_label": "Kitchen_counter",
16

```

Figure 3. An example data fragment taken from a series of observations of a Plate in a series of kitchen scenes, showing object, furniture, room and surface co-occurrence

5 Semantic Web Mining

For an unknown object, our aim is to be able to provide a list of likely DBpedia concepts to describe it, and we will later consider and compare the merits and difficulties associated with providing object *labels* and object *categories*. As this knowledge is not available on the robot (the object is *locally* unknown), it must query an external data

source to fill this knowledge gap. We therefore use the scene descriptions and spatial contexts for an unknown object to generate a query to a Web service. In return this service provides a list of the possible DBpedia concepts which may describe the unknown object. We expect the robot to use this list in the future to either automatically label a new object model, or to use the list of possible concepts to guide a human through a restricted (rather than open-ended) learning interaction.

The Web service provides access to object- and scene-relevant knowledge extracted from Web sources. It is queried using a JSON structure sent via an HTTP request (shown in Figure 2). This structure aggregates the spatial contexts collected over multiple observations of the unknown object. In our current work we focus on the co-occurrence structure. Each entry in this structure describes an object that was observed with the unknown object, the ratio of observations this object was in, and the spatial relation that most frequently held between the two. The room and surface fields describe where the observations were made.

Upon receiving a query, the service computes the *semantic relatedness* between each object included in the co-occurrence structure and every object in a large set of candidate objects from which possible concepts are drawn from (we discuss the nature of this set in Section 6).

This semantic relatedness is computed by leveraging the vectorial representation of the DBpedia concepts provided by the NASARI resource [3]. In NASARI each concept contained in the multilingual resource BabelNet [16] is represented as a vector in a high-dimensional geometric space. The vector components are computed with the *word2vec* [15] tool, based on the cooccurrence of the mentions of each concept, in this case using Wikipedia as source corpus.

Since the vectors are based on distributional semantic knowledge (based on the *distributional hypothesis*: words that occur together often are likely semantically related.), vectors that represent related entities end up close in the vector space. We are able to measure such relatedness by computing the inverse of the cosine distance between two vectors. For instance, the NASARI vectors for *Pointing_device* and *Mouse_(computing)* have relatedness 0.98 (on a continuous scale from 0 to 1), while *Mousepad* and *Teabox* are 0.26 related.

The system computes the aggregate of the relatedness of a candidate object to each of the scene objects contained in the query. Using relatedness to score the likely descriptions of an unknown object follows from the intuition that related objects are more likely than unrelated objects to appear in a scene, e.g., to identify a *Teapot* is more useful to know that there is a *Teacup* at the scene rather than a *Desk*.

Formally, given n observed objects in the query q_1, \dots, q_n , and m candidate objects in the universe under consideration $o_1, \dots, o_m \in O$, each o_i is given a score that indicates its likelihood of being the unknown object by aggregating its relatedness across all observed objects. The aggregation function can be as simple as the arithmetic mean of the relatedness scores, or a more complex function. For instance, if the aggregation function is the product, the likelihood of an object o_i is given by:

$$\text{likelihood}(o_i) = \prod_{j=1}^n \text{relatedness}(o_i, q_j)$$

For the sake of this work, we experimented with the product as aggregating function. This way of aggregating similarity scores gives higher weight to highly related pairs, as opposed to the arithmetic

mean, where each query object contributes equally to the final score. The idea behind this choice is that if an object is highly related to the target it should be regarded as more informative.

The information carried by each query is richer than just a bare set of object labels. One piece of knowledge that can be exploited to obtain a more accurate prediction is the relative position of the observed objects with respect to the target unknown object. Since this information is represented as a discrete level or proximity (from *near_0* to *near_3*), we can use this as a threshold to determine whether or not an object should be included in relatedness calculation. In this work we discard any object related by *near_3*, based on the intuition that the further away an object is spatially, the less related it is. Section 6.2 includes an empirical investigation into approach.

For clarity, here we present an example of execution of the algorithm described above on the query corresponding to the kitchen example seen throughout the paper. The input to the Web module is a query containing a list of pairs (object, distance): (*Refrigerator*, 3), (*Kitchen_cabinet*, 3), (*Sink*, 3), (*Kitchen_cabinet*, 3), (*Sugar_bowl_(dishware)*, 1), (*Teabox*, 1), (*Instant_coffee*, 2), (*Electric_water_boiler*, 3). For the sake of readability, let us assume a set of candidate objects made only of three elements: *Tea_cosy*, *Pitcher_(container)* and *Mug*. Table 1 show the full matrix of pairwise similarities.

	Tea_cosy	Pitcher_(container)	Mug
Refrigerator	0.473	0.544	0.522
Sink	0.565	0.693	0.621
Sugar_bowl_(dishware)	0.555	0.600	0.627
Teabox	0.781	0.466	0.602
Instant_coffee	0.821	0.575	0.796
Electric_water_boiler	0.503	0.559	0.488
product	0.048	0.034	0.047

Table 1. Object similarity of the three candidates *Tea_cosy*, *Pitcher_(container)* and *Mug* to the objects observed at the example kitchen scene. The last line shows the similarity scores aggregated by product.

Among the three candidates, the one with highest aggregated score is *Tea_cosy*, followed by *Mug* and *Pitcher_(container)*. For reference, the ground truth in the example query is *Mug*, that ended up second in the final ranking returned by the algorithm.

We can also alter the performance of the system using the *frequency* of the objects returned by the query. The notion of frequency, taken from [4], is a measure based on the number of incoming links in the Wikipedia page of an entity. Using this measure we can choose to filter uncommon objects from the results of the query, by thresholding with a given frequency value. In the example above, the frequency counts of *Tea_cosy*, *Pitcher_(container)* and *Mug* are respectively 25, 161 and 108. Setting a threshold anywhere between 25 and 100 would filter *Tea_cosy* out of the result, moving up the ground truth to rank 1. Similarly, we can filter out objects that are too far from the target by imposing a limit on their observed distance. A threshold of 2 (inclusive) for the distance of the objects in the example would exclude *Refrigerator*, *Kitchen_cabinet*, *Sink* and *Electric_water_boiler* from the computation.

Other useful information available from the spatial context includes the label of the room, surface or furniture where the unknown was observed. Unfortunately, in order to leverage such information, one needs a complete knowledge base containing these kind of relations, and such a collection is unavailable at the moment. However,

the room and the surface labels are included in the relatedness calculations along with the observed objects.

6 Experiments

In order to evaluate the effectiveness of the method we propose in predicting unknown objects' labels, we perform some experimental tests. In this section we report on the experimental setup and the results we obtained, before discussing them in further detail.

6.1 Experimental Set-up

Our experimental evaluation is an experiment based on a collection of panoramic RGB-D scans taken from an autonomous mobile service robot deployed in a working office for a month. It took these scans at fixed locations according to a flexible schedule. After the deployment we annotated the objects and furniture items in these sweeps, providing each one with a DBpedia concept. This gives us 1329 real world scenes (384 kitchen, 945 office) on which we can test our approach. From this data, our evaluation treats each labeled object in turn as an unknown object in a leave-one-out experiment, querying the Web service with the historical spatial context data for the unknown object similar to that shown in Figure 3.



Figure 4. An example office scene as an RGB image from our real-world deployment. Our data contains 945 office scenes, and 384 kitchen scenes.

In all of the experiments we compare the ground truth (known label in the data) to the DBpedia concepts predicted by our system. We measure performance based on two metrics. The first *WUP similarity* measures the semantic similarity between the ground truth and the concept predicted as most likely for the unknown object. The second measure is the *ranking* of the ground truth in the list of suggested concepts.

For the experiments, the set of candidate objects (O in Section 5) was created by adding all concepts from the DBpedia ontology connected to the room types in our data by up to a depth of 3. For example, starting from office leads us to office equipment, computers, stationary etc. This resulted in a set of 1248 possible concepts. We set the frequency threshold to 20, meaning we ignored any suggest concept which had a frequency value lower than this. This means uncommon concepts such as *Chafing_dish* (frequency=13) would

always be ignored if suggested, but more common ones such as *Mouse_(computing)* (frequency=1106) would be kept.

6.2 Results

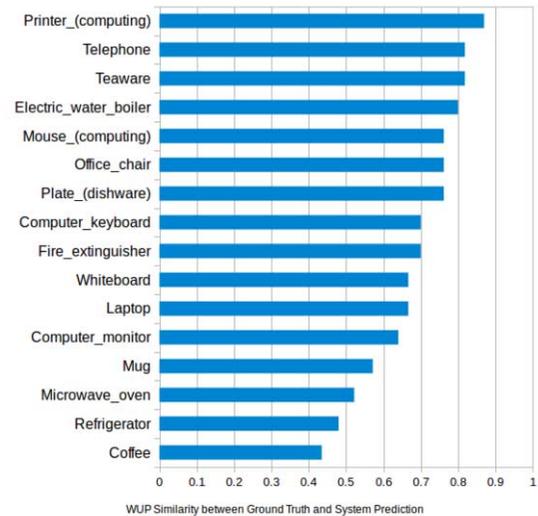


Figure 5. WUP similarity measure between WordNet synsets of ground truth and top-ranked result, with $t = 50$, $p = 2$ using the *prod* method. Ranks closer to 1 are better. Values closer to 1 indicate similarity, and values closer to 0 indicate dissimilarity.

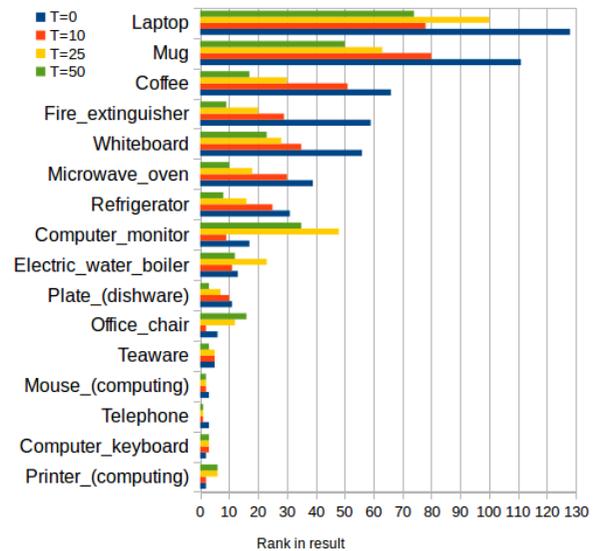


Figure 6. Rank in result by object category, matching the highest ranked object with a category shared with the ground truth in the result set, with varying values of the parameter t , with $p = 2$ and the *prod* method. Ranks closer to 1 are better. Ranking is determined by the position in the result of the first object with an immediate category in common with the ground truth. 56% (9/16) achieve ≤ 10 .

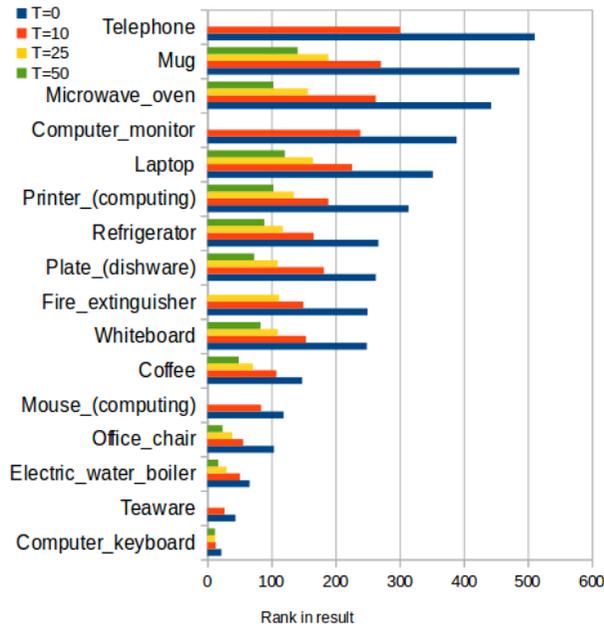


Figure 7. Rank in result by object label, matching the label of the ground truth in the result set, with varying values of the parameter t , with $p = 2$ and the *prod* method. Increasing values of T can cause some objects to be excluded from the result set entirely, such as the Teaware or Monitor at $T=50$

Figure 5 shows the result of calculating the WUP similarity [22] between the WordNet synsets of the ground truth and the top-ranked result from our semantic web-mining system. WUP measures semantic relatedness by considering the depth of two synsets in addition to the depth of their Lowest Common Subsumer (LCS). This means that large leaps between concepts will reduce the eventual similarity score more than small hops might. To do this we used ready available mappings to link DBpedia concepts in our system to WordNet synsets, which are themselves organised as a hierarchy of *is-a* relations. This is in contrast to DBpedia, which is organised as a directed acyclic graph, and while that still means that we could apply the WUP measure to DBpedia nodes directly, WordNet offers a more structured taxonomy of concepts that is more well-suited to this kind of work. This serves to highlight the importance of a multi-modal approach to the use of such ontologies. In the results, the system predicted *Lightpen* when the ground truth was *Mouse* producing a WUP score of 0.73, with the LCS being the *Device* concept, with *Mouse* and *Lightpen* having depth 10 and 11 respectively, and *Device* having depth 8 measured from the root node of *Entity*. In this case, the system suggested an object that fell within 3 concepts of the ground truth, and this is true for the majority of the results in 5. However, in the case of *refrigerator* as the ground truth, the system suggests *keypad* as the highest ranked result, producing a WUP score of 0.52. Here, the LCS is at depth 6 with the concept *Artifact*, the ground truth *refrigerator* is at depth 13 and the prediction *keypad* is at depth 10. While in this case the node distance between the LCS and the prediction is 4, where in the previous example it was 3, the WUP score is much worse here (0.73 vs 0.52) as there are more large leaps across conceptual space. Our best result in this experiment is for *Printer* as the ground truth, for which the system suggests *keypad* again, however the LCS here is the *peripheral* node at depth 10, where *printer* is at depth 11 and *keypad* is at depth 12.

	Mean	Median	Std. Dev	Variance	Range
WUP	0.69	0.70	0.12	0.01	0.43
Category Rank	17.00	9.50	20.17	407.20	73.00
Object Rank	50.93	36.5	50.18	2518.32	141

Figure 8. Statistics on WUP and Rank-in-result data, both for $t = 50$, $p = 2$ using *prod*

The system suggests a range of objects that are closely related to the unknown object, inferred only from its spatial context and knowledge of the objects and environment around it. From here this allows us to generate a list of candidate concepts which we can use in a second stage of refinement, such as by presentation to a human-in-loop.

Figure 6 shows how frequency thresholding effects the performance of the system. In this experiment we consider the position in the ranked result of the first object with an immediate parent DBpedia category in common with the ground truth. Doing so essentially maps the larger set of object labels to a smaller set of object categories. This is in contrast to considering the position in the result of the specific ground truth label, as shown in Figure 7, and allows us to generate a ranking over *categories of objects*. To ensure categories remain relevant to the situated objects we are interested in, we prune a number of DBpedia categories such as those listing objects invented in certain years, or in certain countries. We regard these as being overly broad, and provide a more abstract degree of semantic knowledge about objects than we are interested in. As such, we retrieve the rank-in-result of the first object that shares an immediate DBpedia category with the ground truth, which in the case of *Electric water boiler* turns out to be *Samovar*, a kind of Russian water boiler, as both share the immediate ancestor category *Boilers_(cookware)*. The *Samovar*, and thus the boiler category, appears at rank 12, whereas the specific label *Electric water boiler* appears near the end of the result set of 1248 objects, which covers 641 unique DBpedia categories. In our results, categories associated with 9 of the 16 objects (56%) appear within the result's top 10 entries. Here as we filter out uncommon words by increasing the filter threshold T we improve the position of the concept in the list. Whilst this allows us to definitely remove very unlikely answers that appear related due to some quirk of the data, the more we also start to reduce the ability of the robot to learn about certain objects. This is discussed further in Section 7.

Unlike WordNet synsets and concepts, DBpedia categories are more loosely defined and structured, being generated from Wikipedia, but this means they are typically richer in the kind of semantic detail and broad knowledge representation that may be more suitable for presentation to humans, or more easily mapped to human-authored domains. While WordNet affords us access to a well-defined hierarchy of concepts, categories like *device* and *container* are fairly broad, whereas DBpedia categories such as *Video_game_control_methods* or *Kitchenware* describe a smaller set of potential objects, but may be more semantically meaningful when presented to humans.

7 Discussion

Overall, whilst the results of our object category prediction system show that it is possible for this novel system to generate some good predictions, the performance is variable across objects. There are a number of factors that influence performance, and lead to this variability. The first issue is that the current system does not rule out

suggestions of things it already knows. For example if the unknown object is a keyboard, the spatial context and relatedness may result in a top suggestion of a mouse, but as the system already knows about that, it is probably a less useful suggestion. However, it is possible that the unknown object could be a mouse, but has not been recognised correctly. Perhaps the most fundamental issue in the challenge of predicting objects concepts from limited information is how the limit the scope of suggestions. In our system we restricted ourselves to 1248 possible concepts, automatically selected from DBpedia by ontological connectivity. This is clearly a tiny fraction of all the possible objects in existence. On one hand this means that our autonomous robot will potentially be quite limited in what it can learn about. On the other hand, a large number of this restricted set of objects still make for highly unlikely suggestions. One reason for this is the corpus-based automatically-extracted nature of DBpedia, which means that it includes interesting objects which may never be observed by a robot (e.g. *Mangle_(machine)*). More interestingly though is the effect that the structure of the ontology has on the nature of suggestions. In this work we have been using hierarchical knowledge to unpin our space of hypotheses (i.e. the wider world our robot is placed within), but have not addressed this within our system. This leads to a mismatch between our expectations and the performance of the system with respect to arbitrary precision. For example, if the robot sees a joystick as an unknown object, an appropriate DBpedia concept would seem (to us) to be *Controller_(computing)* or *Joystick*. However, much more specific concepts such as *Thrustmaster* and *LogitechThunderpadDigital* are also available to the system in its current form. When learning about an object for the first time, it seems much more useful for the robot to receive a suggestion of the former kind (allowing it to later refine its knowledge to locally observable instances) than the latter (which unlikely to match the environment of the robot). Instead, returning the *category* of the ranked objects our system suggests allows us to go some way towards this as shown in Figure 6, but still provides us a range of possible candidate categories – though narrowed down from 641 possible categories, to in some cases less than 5. As such, from here we can switch to a secondary level of labelling: that of a human-in-loop. We will next integrate the suggestion system with a crowd-sourcing platform, allowing humans that inhabit the robot’s environment to help it label unknown objects. The suggestion system will be used to narrow down the list of candidate categories that will be shown to users as they provide labels for images of objects the robot has seen and learned, but has not yet labelled. While further work is necessary to refine the current 56% of objects that have a category in the top-10 ranked result, we expect that the current results will be sufficient enough to allow a human to pick a good label when provided a brief list of candidates and shown images of the unknown objects. Such systems are crucial for life-long situated learning for mobile robot platforms, and will allow robot systems to extend their world models over time, and learn new objects and patterns.

The issue of how to select which set of possible objects to draw suggestions from is at the heart of the challenge of this work: make the set too large and it is hard to get good, accurate suggestions, but make it too small and you risk ruling out objects that your robot may need to know about. Whilst the use of frequency-based filtering improved our results by removing low-frequency outliers, more semantically-aware approaches may be necessary to improve things further. Further improvements can be made, for instance we largely do not use current instance observations about the object, but prefer its historical context when forming queries. This may be the wrong

thing to do in some cases, in fact it may be preferable to weight observations of object context based on their recency. The difference between historical context and the context of an object in a particular instance may provide important contextual clues, and allow us to perform other tasks such as anomaly detection or boost the speed of object search tasks.

One issue we believe our work highlights is the need to integrate a multi-modal approach to the use of differing corpora and ontologies. For instance, the more formal WordNet hierarchy was used to calculate the semantic relatedness of our experiment results, rather than the less formal DBpedia ontology. However we hold that the DBpedia category relationships are more useful in the human-facing component of our system. There exist other ontologies such as YAGO which integrates both WordNet and DBpedia, along with its own category system, that will certainly be of interest to us in the future as we seek to improve the performance of our system. One of our primary goals is to better exploit the hierarchical nature of these ontologies to provide a way of retrieving richer categorical information about objects. While reliably predicting the specific object label from spatial context alone is difficult, we *can* provide higher-level ancestor categories that could be used to spur further learning or improve previous results. As such, we view the prediction process as one of matching the characteristics of a series of increasingly more specific categories to the characteristics of an unknown object, rather than immediately attempting to match the specific lowest-level characteristics and produce the direct object label. This requires an ontology both formally-defined enough to express a meaningful hierarchy of categories for each item, *and* broad enough to provide us mapping to a large set of common-sense categories and objects. It is not clear yet which combination of existing tools will provide the best route to accomplishing this.

8 Conclusions

In this paper we presented an integrated system for solving a novel problem: the suggestion of concept labels for unknown objects observed by a mobile robot. Our system stores the spatial contexts in which objects are observed and uses these to query a Web-based suggestion system to receive a list of possible concepts that could apply to the unknown object. These suggestions are based on the relatedness of the objects observed with the unknown object, and can be improved by filtering the results based on both frequency and spatial proximity. We evaluated our system data from real office observations and demonstrated how various filter parameters changed the match of the results to ground truth data.

We showed that the suggestion systems provides object label suggestions with a reasonably high degree of semantic similarity, as measured by WUP similarity on WordNet synsets. We also achieved success in retrieving the *categories* of objects, rather than their direct labels. In the future we will explore the hierarchical nature of the knowledge used for object concept suggestions, explore different corpora and ontologies to base the suggesting system on, and perform a situated evaluation of our system on a mobile robot with additional perceptual learning capabilities, and crowd-sourcing functionality to label objects on-line with the help of humans using the suggestion system.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under ALOOF project (CHIST-ERA program)

REFERENCES

- [1] Michael Beetz, Ferenc Bálint-Benczédi, Nico Blodow, Daniel Nyga, Thiemo Wiedemeyer, and Zoltán-Csaba Marton, 'Robosherlock: Unstructured information processing for robot perception', in *ICRA*, (2015).
- [2] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann, 'DBpedia - a crystallization point for the web of data', *Web Semant.*, **7**(3), 154–165, (September 2009).
- [3] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli, 'Nasari: a novel approach to a semantically-aware representation of items.', in *HLT-NAACL*, eds., Rada Mihalcea, Joyce Yue Chai, and Anoop Sarkar, pp. 567–577. The Association for Computational Linguistics, (2015).
- [4] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes, 'Improving efficiency and accuracy in multilingual entity extraction', in *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, (2013).
- [5] T. Faeulhammer, R. Ambrus, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze, 'Autonomous learning of object models on a mobile robot', *IEEE RAL*, **PP**(99), 1–1, (2016).
- [6] Christiane Fellbaum, *WordNet: An Electronic Lexical Database*, Bradford Books, 1998.
- [7] Ross Finman, Thomas Whelan, Michael Kaess, and John J Leonard, 'Toward lifelong object segmentation from change detection in dense rgb-d maps', in *ECMR*. IEEE, (2013).
- [8] L. Frommberger and D. Wolter, 'Structural knowledge transfer by spatial abstraction for reinforcement learning agents', *Adaptive Behavior*, **18**(6), 507–525, (December 2010).
- [9] Guglielmo Gemignani, Roberto Capobianco, Emanuele Bastianelli, Domenico Bloisi, Luca Iocchi, and Daniele Nardi, 'Living with robots: Interactive environmental knowledge acquisition', *Robotics and Autonomous Systems*, (2016).
- [10] Marc Hanheide, Charles Gretton, Richard Dearden, Nick Hawes, Jeremy L. Wyatt, Andrzej Pronobis, Alper Aydemir, Moritz Göbelbecker, and Hendrik Zender, 'Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour', in *IJCAI'11*, Barcelona, Spain, (July 2011).
- [11] S. Hinterstoisser, S. Holzer, C. Cagniard, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, 'Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes', in *IEEE ICCV*, (2011).
- [12] Lars Kunze, Chris Burbridge, Marina Alberti, Akshaya Tippur, John Folkesson, Patric Jensfelt, and Nick Hawes, 'Combining top-down spatial reasoning and bottom-up object class recognition for scene understanding', in *IEEE IROS*, Chicago, Illinois, US, (September, 14–18 2014).
- [13] Thomas K Landauer and Susan T. Dumais, 'A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge', *PSYCHOLOGICAL REVIEW*, **104**(2), 211–240, (1997).
- [14] José Paulo Leal, Vânia Rodrigues, and Ricardo Queirós, 'Computing Semantic Relatedness using DBpedia', in *1st Symposium on Languages, Applications and Technologies*, eds., Alberto Simões, Ricardo Queirós, and Daniela da Cruz, volume 21 of *OpenAccess Series in Informatics (OASISs)*, pp. 133–147, Dagstuhl, Germany, (2012). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781*, (2013).
- [16] Roberto Navigli and Simone Paolo Ponzetto, 'Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network', *Artificial Intelligence*, **193**(0), 217 – 250, (2012).
- [17] J Philbin, 'Lost in quantization: Improving particular object retrieval in large scale image databases', in *CVPR 2008*, pp. 1–8, (June 2008).
- [18] M.J. Schuster, D. Jain, M. Tenorth, and M. Beetz, 'Learning organizational principles in human environments', in *ICRA*, pp. 3867–3874, (May 2012).
- [19] Shuran Song, Linguang Zhang, and Jianxiong Xiao, 'Robot in a room: Toward perfect object recognition in closed environments', *CoRR*, **abs/1507.02703**, (2015).
- [20] Moritz Tenorth, Lars Kunze, Dominik Jain, and Michael Beetz, 'KNOWROB-MAP – knowledge-linked semantic object maps', in *IEEE-RAS ICHR*, pp. 430–435, Nashville, TN, USA, (December 6-8 2010).
- [21] Akshaya Thippur, Chris Burbridge, Lars Kunze, Marina Alberti, John Folkesson, Patric Jensfelt, and Nick Hawes, 'A comparison of qualitative and metric spatial relation models for scene understanding', in *AAAI'15*, (January 2015).
- [22] Zhibiao Wu and Martha Palmer, 'Verbs semantics and lexical selection', in *ACL*, ACL '94, pp. 133–138, Stroudsburg, PA, USA, (1994). Association for Computational Linguistics.
- [23] Jay Young and Nick Hawes, 'Learning by observation using qualitative spatial relations', in *AAMAS 2015*, (May 2015).

Declaratively Capturing Local Label Correlations with Multi-Label Trees

Reem Al-Otaibi^{1,2} and Meelis Kull^{1,3} and Peter Flach¹

Abstract. The goal of multi-label classification is to predict multiple labels per data point simultaneously. Real-world applications tend to have high-dimensional label spaces, employing hundreds or even thousands of labels. While these labels could be predicted separately, by capturing label correlation we might achieve better predictive performance. In contrast with previous attempts in the literature that have modelled label correlations globally, this paper proposes a novel algorithm to model correlations and cluster labels locally. **LaCovaC** is a multi-label decision tree classifier that clusters labels into several dependent subsets at various points during training. The clusters are obtained locally by identifying the conditionally-dependent labels in localised regions of the feature space using the label correlation matrix. **LaCovaC** interleaves between two main decisions on the label matrix with training instances in rows and labels in columns: splitting this matrix vertically by partitioning the labels into subsets, or splitting it horizontally using features in the conventional way. Experiments on 13 benchmark datasets demonstrate that our proposal achieves competitive performance over a wide range of evaluation metrics when compared with the state-of-the-art multi-label classifiers.

1 Introduction and Motivation

In traditional classification each data point is assigned to a single label. In binary classification each point can belong to one of two classes, whereas in multi-class classification the setting is more general, allowing each training point to belong to one of more than two classes. Multi-label classification generalises both by allowing more than one label for each data point [20, 28]. Thus, it allows for a wide range of applications, such as text categorisation, image and movie tagging, and gene function prediction. For example, a medical diagnosis might find a patient has multiple diseases at one time; an article that gives statistics about the number of students who have applied to Medical schools in a country could be categorised as both educational and medical; and an image that captures a beach at sunset could belong to both beach and sunset groups. Thus, all these examples naturally yield multiple labels.

Existing approaches in multi-label learning can be categorised into two main types. Problem transformation approaches decompose multi-label data into several binary problems, in order to use a binary classifier. For example, a multi-label problem with $|L|$ labels can be solved with $|L|$ binary classifiers, in which all predictions are then merged to produce final predictions. In the second type of approaches the algorithms handle multi-label data directly.

The main challenge to note is that labels in real-world applications often have a relationship or connection, whereby the presence of one label affects or depends on another. Several studies argue that exploiting label correlations is important in the area of multi-label classification [7, 27, 28]. Although a considerable amount of work has been done in this area, these have chiefly focused on a global approach, in which label correlations are identified as a pre-processing step prior to training the model.

Decision tree algorithms are among the most widely used algorithms for classification [13, 17]. Considering the advantages of decision tree models, this paper proposes **LaCovaC**, which is a multi-label decision tree classifier. **LaCovaC** utilises the label correlation matrix at every node of the tree to find possible clusters among the labels. In addition to internal nodes that split the dataset horizontally based on selected features, **LaCovaC** introduces a second kind of node for splitting the label space vertically. At deployment, a horizontal split routes to exactly one child node according to a feature value as per normal, while a vertical split tests all outgoing edges to collect predictions about the entire labelset.

The remainder of this paper is organised as follows. Section 2 provides an overview of existing approaches to exploit correlations in multi-label classification. **LaCovaC** is presented in Section 3, and an experimental evaluation is presented and discussed in Section 4. Section 5 concludes the paper by stating the main findings and possible avenues for further work.

2 Related Work

Two well-known baseline algorithms have been considered for use as problem transformation methods: Binary Relevance (BR) [19] and Label Powerset (LP) [23]. BR applies one binary classifier to each individual label. It transforms the original dataset D into $|L|$ datasets, each of which comprises all examples of the original dataset. The examples are labelled positively if the labelset for the original example contains this label, and negatively if not. To classify a new instance, BR outputs the union of labels that have been positively predicted by the $|L|$ classifiers. Label Powerset (LP), also known as Label Combination, considers each unique set of labels that exists in a multi-label training set as a new class in a multi-class classification task. It is apparent that BR does not model label dependency, whereas LP does. However, overfitting and the exponential number of label combinations are potential difficulties affecting LP. Therefore, many different directions have been taken in the literature to address label correlations. We summarise these into several distinct approaches below.

The first approach is to transform a multi-label problem into several binary classification tasks by considering label correlations. A well-known algorithm called the Classifier Chain (CC) involves $|L|$ binary classifiers as in BR, but orders them along a chain, wherein

¹ Intelligent System Laboratory, University of Bristol, United Kingdom, emails: ra12404, meelis.kull, peter.flach@bristol.ac.uk

² King Abdulaziz University, Saudi Arabia

³ University of Tartu, Estonia

each classifier deals with the binary relevance problem associated with a label. Importantly, the feature space of each link in the chain is extended with the 0/1 label predictions for all previous links [16].

The same authors of CC have proposed the Ensembles of Classifier Chains (ECC), an ensemble method where the individual classifiers are trained with different orders of labels in the chain and a random subset of the training data, encouraging variability among the classifiers. These predictions are summed per label so that each label receives a number of votes. A threshold is used to select the most popular labels, which form the final predicted label set [16].

Clearly, CC and ECC are sensitive to the label order in the training process. A number of extensions to the original CC method have been proposed in [6, 7, 11, 15, 26], aiming at eliminating the key drawback in the CC, which is the lack of a principled way to decide on the label ordering.

Although CC considers label correlations by inserting the labels as new features, there is an important point to consider. It is the fact that all labels are inserted as additional features along the chain until the last label in the chain contains all previous labels as extra input features. This can be a limiting factor in high-dimensional label spaces. Furthermore, these methods force all preceding labels to be additional features for the examined label which might not be relevant or useful.

The second approach is to exploit correlations between labels by clustering them. The hierarchy of multi-label classifier (Homer) organises all labels into a tree-shaped hierarchy, with leaf nodes containing a single label [21]. Each node has training instances that are annotated with at least one of its labels. In the training phase, a multi-label classifier is trained for each node to predict a subset of labels in that node. In particular, a leaf node constructs a binary classifier to predict its single label. Given an unseen instance, Homer starts from the root node and proceeds to any successor node only if at least one of its labels was predicted by its parent node. In the end, this process reaches a subset of leaves and the final prediction is combined from predictions of these leaves. A recent work proposed in [4] combines the LP and BR methods, and is called LPBR. Its first step is to explore the dependencies between labels and then to cluster these labels into several independent subsets, according to the chi-squared statistic. Subsequently, a multi-label classifier is learnt: if the set contains only one label, BR is applied; for a group of dependent labels, LP is used. LPBR implements a greedy clustering algorithm that continues clustering as long as the loss function improves. While LPBR showed an improvement in terms of classification accuracy, it is computationally expensive.

In [12] the authors propose a method ML-LOC which first clusters the instances with respect to similarity of features and labels jointly. Subsequently, the feature space is extended with an additional feature encoding the cluster membership. Given a test instance, this additional feature is predicted using a regression model. The final predictions are then obtained from any standard multi-label classifier, trained on the extended feature space.

The final approach is to adopt decision tree algorithms in a multi-label setting. ML-C4.5 was proposed by [5] to deal with multi-label data, while the basic strategy was to define multi-label entropy separately over a set of multi-label examples. The modified entropy sums the entropies for each individual label. Another recent work was proposed in [10], and also builds a single tree for a multi-label dataset. They proposed a hybrid decision tree architecture to utilise support vector machines (SVMs) at its leaves. This approach, known as ML-SVMMDT, combines two models: ML-C4.5 and BR. It builds a single decision tree, similar to ML-C4.5, whose leaves contain BR clas-

sifiers giving multi-label predictions using SVM. LaCova was proposed in [2] and is a tree based multi-label classifier that uses label covariance as a splitting criterion. The principle of LaCova is to use the label covariance matrix at each node of the tree to treat labels independently (i.e., learn a BR model from then on) or keep them together (LP) for now.

In this work we explore the value of mediating between these extreme decisions at each node in the tree. We propose **LaCovaC** which – different from previous methods – clusters labels *dynamically* during the construction of the decision tree, and hence models conditional label correlations at every node of the tree. Although other models attempt to cluster labels, they do so over the entire dataset, e.g. Homer and LPBR. In practice, conditional correlations that are used for clustering may be local, and depend on specific feature values. Hence **LaCovaC** will not separate labels automatically as happens in BR, but only in cases when labels are uncorrelated. Additionally, **LaCovaC** would not model the joint distribution at all times, as this can cause overfitting, as in LP. These decisions are taken locally at every node in the tree.

3 The Proposed Model

The key theoretical underpinning of decision trees is that they identify regions in instance space with low label variance, which is built into the splitting criterion. Naive extensions to the multi-label setting would either lead to a separate tree for each label (as in BR) or keeping all labels together (as in LP). Learning a separate tree for each label would result in as many trees as there are labels, which can be hundreds or thousands in some domains (in our experiments it can be up to 374). Furthermore, the explanatory power of the decision tree models would be reduced, which is an important factor in medical domains, among many others.

3.1 An Illustrative Example

Before providing a formal definition, a simple example is introduced here to illustrate the proposed algorithm. IMDB⁴ is a multi-label dataset that contains keywords describing movies, and the classification task is to predict the movie’s genre. Each movie can be assigned multiple genres from among 27 labels. For simplicity, we have selected two input features: dark and love; and three movie genres: crime, horror and drama.

Table 1: A small multi-label dataset with 12 instances as might be used in movie genre classification.

	labels			features	
	crime	horror	drama	love	dark
1	1	1	1	1	1
2	1	0	0	0	1
3	1	0	1	1	1
4	1	0	0	0	1
5	1	0	1	1	1
6	0	1	0	0	1
7	0	1	1	1	1
8	0	1	0	0	1
9	0	1	1	1	1
10	1	1	1	1	0
11	1	0	0	0	0
12	0	1	1	1	0

Inspired by the IMDB dataset we have created a toy dataset, shown in Table 1, which we use to demonstrate the advantages of **LaCovaC**

⁴ <http://meka.sourceforge.net/>

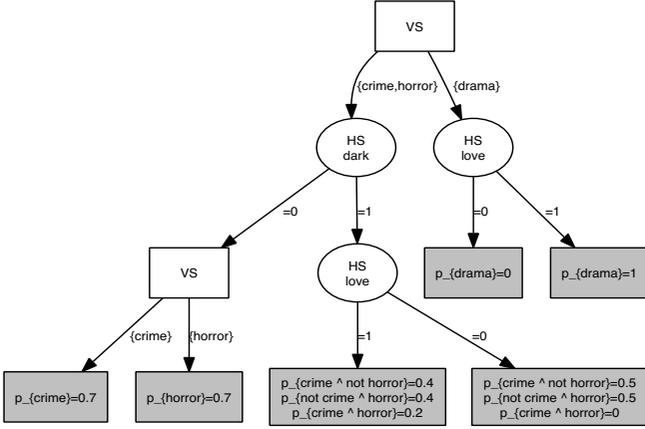


Figure 1: An example of a tree learned by **LaCovaC**. VS and HS stand for vertical and horizontal splits, respectively. The leaves show the probabilities of each label subset, except for the empty subset which can be calculated as 1 minus the sum of the given probabilities.

over binary relevance. Figure 1 shows the tree generated using **LaCovaC**. In this example, **LaCovaC** initially finds two label clusters: $\{\text{crime}, \text{horror}\}$ and $\{\text{drama}\}$. Therefore, it splits the dataset vertically, and recursively creates a tree for each cluster of labels. On the left-hand side of the tree a horizontal split can be observed at the next level using the feature `dark`, leading to another vertical split on the left and a node that requires further horizontal splitting. Note that the second vertical split did not split further due to a minimum cardinality constraint (2 instances in this toy example). As can be seen on the left-hand side of the tree, $\{\text{crime}, \text{horror}\}$ are independent when `dark` is false, and dependent otherwise.

When the feature `love` is true both labels share the same estimated marginal probability of 0.6, which can lead to predicting both of them as relevant labels, assuming 0.5 as a threshold. However, the proposed model suggests that the two subsets $\{\text{crime}=0, \text{horror}=1\}$ or $\{\text{crime}=1, \text{horror}=0\}$ with the highest probability among other subsets can lead to a better prediction. In other words, leaves predict the label subset that has the highest probability among others, regardless of their marginal probabilities. With regards to the third label $\{\text{drama}\}$, the proposed algorithm learns it separately as can be seen on the right-hand side of the tree.

This example demonstrates the power of modelling label correlations locally in the tree instead of globally in pre-processing, and also that the tree model represents such local correlations in a declarative way.

When applying the learned model on a test instance, one outgoing branch of a horizontal split should be followed based on the feature value as in a standard decision tree. In contrast, in a vertical split, all outgoing branches should be followed to collect predictions for all labels reaching the node. For example, in order to label a movie which has features $\{\text{dark}=1, \text{love}=1\}$, the two branches of the first vertical split should be visited. On the right-hand side of the tree, we find that the `drama` label does apply. On the left-hand side of the tree, we will have a horizontal split based on the `dark` feature leading to the second horizontal node based on the `love` feature. The right leaf of this second horizontal split suggests two label subsets: $\{\text{crime}=1, \text{horror}=0\}$ or $\{\text{crime}=0, \text{horror}=1\}$. Suppose that the first set is chosen, then the overall predictions for this movie will be the fol-

lowing labels $\{\text{crime}=1, \text{horror}=0, \text{drama}=1\}$.

Figure 2 shows the result of binary relevance on this example with single-label decision trees as base model. The leftmost tree is for the `crime` label; the root node splits on the feature `dark`, when `dark=0` it leads to a leaf due to a minimum cardinality constraint (2 instances). On the right-hand side of the tree, it splits further based on the feature `love`. Leaves show the estimated probabilities and the predicted label. The middle and right-most tree were constructed using the same method to learn labels `horror` and `drama`, respectively.

In order to predict genres for a new movie with $\{\text{dark}=1$ and $\text{love}=1\}$ all three genres should be tested. The prediction for this movie will be $\{\text{crime}, \text{horror}, \text{drama}\}$, using 0.5 as a threshold. This is different from what the proposed model **LaCovaC** predicts.

3.2 The Main Algorithm

LaCovaC implements three key decisions at every node of the decision tree, and the process can be summarised as follows. Firstly, if labels are pure or the set of instances reaches the minimum number of data points, the algorithm stops and returns a leaf. Secondly, if the label's correlation matrix suggests the presence of cluster structure, the algorithm splits the dataset reaching that node according to the label clusters (vertical split). Finally, a set of labels located together at a node are learned, and the best features to split are determined (horizontal split).

The first decision requires a threshold on the label variance. In our experiments we found that, in combination with a minimum number of instances at a leaf, a variance threshold of 0 (i.e. all labels are pure) works well. The detection of cluster structure is presented in the next section. The third decision that splits instance space horizontally also demands a splitting criterion. The most popular splitting criteria in standard decision trees learning are Gini-split (which uses the Gini index to measure impurity) and information gain (which uses Shannon entropy). However, **LaCovaC** implements a splitting criterion designed specifically for multi-label data, following a previous work [2], which can be summarised as follows. It evaluates for each child both its sum of label variances and its sum of absolute label covariances, and assigns as a splitting measure the minimum of these two (low is better). Then, it selects the feature that has the lowest splitting measure. This criterion can identify regions with either low multi-label variance or low label covariance. We improve on this by clustering the labels in a principled way into independent groups of correlated labels.

Algorithm 1 details the main algorithm implementing the approach described here.

3.3 Label Clustering

The labels are clustered based on the correlation matrix Cor , which is a square $|L|$ by $|L|$ matrix that contains the Pearson correlation coefficients cor_{jk} between pair of labels l_j and l_k . As labels are binary in the multi-label setting, the Pearson coefficient is reduced to the Phi coefficient, a well-known measure of association between two binary (dichotomous) variables [3, 24]. Whenever the absolute value of correlation between two labels is greater than a threshold $\lambda_{|D|}$ calculated from the number of instances (derivation given below), we decide that these labels are correlated and should be in the same cluster.

Algorithmically, this can be achieved by single linkage agglomerative hierarchical clustering as shown in Algorithm 2. The algorithm starts by creating a separate cluster for each label. It then proceeds

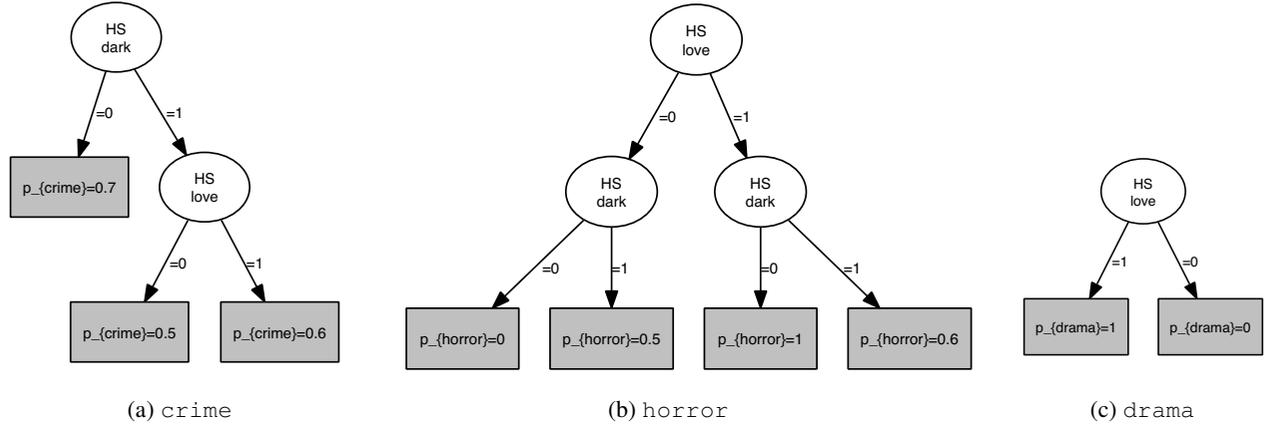


Figure 2: A separate decision tree for each label as learned by the binary relevance baseline method.

Algorithm 1 LaCovaC (D,L): Learn a tree-based multi-label classifier from training data.

Input: Dataset D ; Labelset L ; Minimum number m of instances to split
Output: Tree-based multi-label classifier
if Labels are pure or $|D| < m$ **then**
 Return Leaf with relative frequencies of labelsets, predicted labelset that has the highest probability
else
 $clust = \text{CLUST}(D, L)$
 if $|clust| > 1$ **then**
 for each set s in $clust$ **do**
 $D_s = \text{Split } D \text{ vertically based on the label cluster } s$
 /* Learn a decision tree for the label cluster s */
 $T_s = \text{LaCovaC}(D_s, s)$.
 end for
 Return Node splitting on $clust$ with subtrees T_s
 else
 $f, \{D_i\} = \text{SelectBestFeature}(D)$
 for each child node D_i **do**
 $T_i = \text{LaCovaC}(D_i, L)$
 end for
 Return Node splitting on f with subtrees T_i
 end if
end if

iteratively by taking pairs of labels in the order of increasing distance between them, where distance is defined as one minus absolute correlation, $dist_{jk} = 1 - |cor_{jk}|$. For each pair of labels, the clusters containing these labels are merged (unless they belong to the same cluster already). Such merges are performed until the distance between labels becomes greater than $1 - \lambda_{|D|}$ (that is, absolute correlation less than $\lambda_{|D|}$).

We now derive the threshold λ_n to decide whether a pair of labels are correlated or not, before merging them into one cluster. The idea is to set this threshold equal to two standard deviations in the distribution of correlation under the assumption of independent labels, hence enclosing a 95% confidence interval. The threshold depends on n , which is the number of instances reaching the current node in the tree. To derive the threshold we consider two labels with empirical frequencies p_j and p_k , respectively. The empirical Pearson correlation between these labels as Bernoulli variables can be calculated as

Algorithm 2 CLUST(D,L): Cluster labels based on the correlation matrix

Input: Dataset D ; a set of labels $L = l_1, \dots, l_{|L|}$;
Output: $currClust$ - the clusters of labels
 /* Compute the Pearson correlation coefficients between each pair of labels.*/
 $Cor = \text{Correlation Matrix}$
 /* Compute the distance between each pair of labels using the absolute correlations.*/
 $Dist = 1 - |Cor|$
 /* Build initial clusters with each label in a separate cluster.*/
 $currClust = \{l_1\}, \{l_2\}, \dots, \{l_{|L|}\}$
 /* Create a list of label pairs sorted in ascending order of distance value.*/
 $pairList = \text{all pairs of labels } (l_j, l_k) \text{ with } dist_{jk} < 1 - \lambda_{|D|}, \text{ sorted by } dist_{jk} \text{ ascendingly}$
for each label pair (l_j, l_k) in $pairList$ **do**
 if labels l_j and l_k are in different clusters **then**
 merge clusters containing l_j and l_k within $currClust$
 end if
end for
Return $currClust$

follows:

$$\widehat{cor}_{jk} \approx \frac{C_{jk}/n - p_j p_k}{\sqrt{p_j p_k (1 - p_j)(1 - p_k)}}, \quad (1)$$

where C_{jk} is the the number of instances with both of those labels, and hence C_{jk}/n is the proportion of such instances. The terms $p_j(1 - p_j)$ and $p_k(1 - p_k)$ in the denominator are the variances of these Bernoulli variables.

Next, we study the distribution of \widehat{cor}_{jk} under the assumption that labels l_j and l_k are independent. This distribution can be generated by randomly reassigning one of those labels between the instances, keeping the total frequency constant. In such case C_{jk} has a hypergeometric distribution, and we can then directly calculate its mean μ and variance σ^2 :

$$\begin{aligned} C_{jk} &\sim \text{Hypergeometric}(np_j, np_k, n) \\ \mu &= np_j p_k \\ \sigma^2 &= \frac{n^2}{n-1} p_j p_k (1 - p_j)(1 - p_k) \end{aligned}$$

From this we can calculate the mean and variance of the correlation \widehat{cor}_{jk} between labels l_j and l_k , as follows:

$$\begin{aligned} \text{mean}(\widehat{cor}_{jk}) &= \frac{np_j p_k / n - p_j p_k}{\sqrt{p_j p_k (1-p_j)(1-p_k)}} = 0 \\ \text{var}(\widehat{cor}_{jk}) &= \frac{\frac{n^2}{n-1} p_j p_k (1-p_j)(1-p_k) / n^2}{p_j p_k (1-p_j)(1-p_k)} = \frac{1}{n-1} \end{aligned}$$

We define λ_n as the value enclosing a 95% confidence interval assuming a normal distribution for \widehat{cor}_{jk} which can be calculated in the usual way:

$$\lambda_n = 1.96 \cdot \sqrt{\text{var}(\widehat{cor}_{jk})} = \frac{1.96}{\sqrt{n-1}} \quad (2)$$

where n is the number of instances reaching a particular decision node in the tree. Note that this threshold is always less than 1 as $n > 5$ in our experiments.

4 Experimental Evaluation

In total, 13 common benchmarks were retrieved for use from the Meka⁵ and Mulan [22] repositories. The key statistics of these datasets are shown in Table 2.

Table 2: The statistics for the datasets used in the experiments. $|L|$ is the number of labels, $|D|$ is the number of instances, att is the number of attributes (features), $card$ is the average number of labels per instance, and $dens$ is the label density.

	$ L $	$ D $	att	$card$	$dens$
Corel5k	374	5000	499	3.522	0.94%
Cal500	174	502	68	26.044	14.96%
Bibtex	159	7395	1836	2.402	1.51%
Language log	75	1460	1004	1.179	1.57%
Enron	53	1702	1001	3.378	6.37%
Medical	45	978	1449	1.245	2.76%
Genebase	27	662	1186	1.252	4.63%
Slashdot	22	3782	1079	1.180	5.36%
Birds	19	645	260	1.014	5.30%
Yeast	14	2417	103	4.237	30.26%
Flags	7	194	19	3.392	48.45%
Emotions	6	593	66	1.869	31.14%
Scene	6	2407	294	1.074	17.89%

4.1 Experimental Setup

BR, LP, and CC algorithms were run in Meka, whereas Mulan was used for the Homer and LPBR algorithms. When employing all these methods, the trees were produced with Weka’s J48 algorithm⁶. The Homer algorithm was run using the best setting, as reported by [21]. LPBR requires parameter configurations, such as non-improving counter to prevent clustering. The default parameters settings in Mulan were 5 for the non-improving counter and a 5-fold cross validation for testing the clustering performance. The target loss function to evaluate the clustering was set to exact-match according to Mulan. Exact-match is a strict measure that examines whether the predicted and actual label sets are equal or not.

⁵ <http://meka.sourceforge.net/>

⁶ <http://sourceforge.net/projects/weka>

LaCovaC was implemented in Java on the Meka platform. The LaCova and ML-C4.5 implementations are provided in [2]. The minimal number of instances in the leaves in ML-C4.5, LaCova, and **LaCovaC** was set to 5.

10-fold cross-validation was performed throughout except the two largest datasets in terms of both number of labels and number of instances: Corel5k and Bibtex. The exception is because cross validation was intensive for LPBR as we were not able to run this algorithm using the available resources⁷. Therefore, a train/test split was used instead, which was provided by Mulan.

For all methods we used the Pcut method to convert the label probabilities into relevant labels [1, 25]. This method uses a single threshold for all labels, chosen such that the average label density in test data is the same as in the training data.

4.2 Evaluation Metrics

All algorithms were evaluated under the Meka framework. We used the most common multi-label metrics, namely multi-label accuracy (Jaccard index), exact-match (subset accuracy), Hamming loss and three versions of the F_1 score: micro-averaged over all instance-label combinations, macro-averaged per label, and macro-averaged per instance. We also used a multi-label version of log-loss [16] to evaluate the predicted probabilities. Log-loss evaluates a classifier’s confidence by punishing errors with higher probability more severely. We used a dataset-dependent maximum of $\log(\frac{1}{|D|})$ to limit the magnitudes of penalty as suggested in [16]. The description and formal definition of these evaluation measures for multi-label data are given below.

Multi-label accuracy calculates the ratio of the union and intersection of the predicted and actual labelsets, averaged over all examples [28]:

$$mla = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap \hat{Y}_i|}{|Y_i \cup \hat{Y}_i|} \quad (3)$$

where Y_i and \hat{Y}_i are the set of actual labels and predicted labels for an instance, respectively.

Exact-match examines if the predicted and actual label subsets are equal or not [9]:

$$em = \frac{1}{|D|} \sum_{i=1}^{|D|} I(Y_i = \hat{Y}_i) \quad (4)$$

where $I(true) = 1$ and $I(false) = 0$.

Hamming loss takes the proportion of misclassified labels (labels predicted incorrectly and correct labels that are not predicted) averaged over all examples [18]:

$$hl = \frac{1}{|L| * |D|} \sum_{i=1}^{|D|} |Y_i \Delta \hat{Y}_i| \quad (5)$$

where Δ is the symmetric difference.

⁷ 2.7 GHz Intel Core i5 processor and 8GB of memory

Log-loss evaluates a classifier’s confidence by punishing errors with higher probability more severely [16]:

$$ls = \frac{1}{|D|} \sum_{i=1}^{|D|} \sum_{j=1}^{|L|} \log(pr_{ij}) \cdot y_{ij} + \log(1 - pr_{ij}) \cdot (1 - y_{ij}) \quad (6)$$

where y_{ij} indicates whether the j^{th} label is relevant for the i^{th} instance (value 1) or irrelevant (value 0). pr_{ij} is the probability estimate for the i^{th} instance and the j^{th} label.

Micro F_1 corresponds to the harmonic mean of precision and recall [14]. It put all predictions on all labels in one vector as in binary classification and then calculates the F_1 :

$$\text{micro } F_1 = \frac{\sum_{i=1}^{|D|} \sum_{j=1}^{|L|} 2 \cdot y_{ij} \cdot \hat{y}_{ij}}{\sum_{i=1}^{|D|} \sum_{j=1}^{|L|} (y_{ij} + \hat{y}_{ij})} \quad (7)$$

Macro f_l is the averaged F_1 score over labels [14]:

$$\text{macro } f_l = \frac{1}{|L|} \sum_{j=1}^{|L|} F_1^{(j)} \quad (8)$$

where $F_1^{(j)}$ is the F_1 score for the j^{th} label vector.

Macro f_e is the averaged F_1 score over examples [14]:

$$\text{macro } f_e = \frac{1}{|D|} \sum_{i=1}^{|D|} F_1^i \quad (9)$$

where F_1^i is the F_1 score for the i^{th} instance row vector.

4.3 Results and Discussion

We conducted the Friedman test based on the average ranks for all datasets [8]. This test ranks the algorithms for each dataset separately, thus the best algorithm gets the rank of 1, the second best the rank of 2, etc. Then, it calculates the test statistic on the ranks averaged over all datasets in order to verify whether the differences between algorithms are statistically significant.

Table 3 shows the average ranks for each algorithm over 13 datasets and seven evaluation measures. We also show the mean of these average ranks aggregated over all evaluation measures, which shows that overall **LaCovaC** scores highest, followed by CC and BR. The Friedman test gave a significant difference at 5% confidence for all metrics except multi-label accuracy; therefore, we carried out post-hoc Nemenyi tests as shown in Figure 3.

It can be seen that **LaCovaC** outperforms Homer, ML-C4.5, and LPBR in the average ranks with respect to all the evaluation measures used in this paper, and in several cases statistically significantly. **LaCovaC** is not significantly worse than the best performing algorithm (LP) in terms of exact-match. In fact, according to Table 4 **LaCovaC** loses to LP in the 4 datasets with the smallest numbers of labels (up to 14), but wins in 6 out of the 9 datasets with more labels. This fact confirms the assumption that LP can overfit the training data in case of large number of labels and label combinations because it can only model labelsets observed in the training. Notably, for exact-match the proposed algorithm **LaCovaC** is the overall best method or tied with the best in the three datasets with the biggest numbers of labels

(Corel5k, Cal500 and Bibtex with 374, 174 and 159 labels, respectively). The proposed algorithm has better Hamming loss than LP in 9 out of 13 datasets.

Additionally, **LaCovaC** outperforms BR in terms of exact-match, log-loss and Hamming loss. BR has the best average rank considering F_1 score per instance average f_e , F_1 per label average f_l and micro F_1 as it decomposes the multi-label problem into several binary classifiers, which can be better for F_1 score.

We can further observe that **LaCovaC** has the best average rank in terms of log-loss that evaluates the classifiers’ scores regardless of the thresholding technique. It is significantly better than CC, ML-C4.5, Homer and LP.

Finally, CC gets top average rank for Hamming loss and **LaCovaC** is the second best without significant loss against CC. Detailed results of the experiments are given in Tables 4, 5 and 6.

Table 3: Average ranks obtained by the Friedman test over 13 datasets. The last column shows the mean of the average ranks obtained by each algorithm over all the evaluation measures, and the algorithms are sorted on decreasing mean. *mLa*, *em*, *hl* and *ls* denote multi-label accuracy, exact-match, Hamming loss and log-loss, respectively. micro F_1 is the micro averaged F_1 . macro f_l and macro f_e are F_1 scores averaged in terms of labels and examples, respectively.

	<i>mLa</i>	<i>em</i>	<i>hl</i>	<i>ls</i>	micro F_1	macro f_l	macro f_e	mean
LaCovaC	3.57	2.69	2.73	2.23	3.88	4.34	3.96	3.34
CC	3.46	2.8	1.69	5.19	3.38	4.19	3.69	3.49
BR	4.57	6.8	4.8	2.3	2.69	1.92	3.26	3.76
LaCova	3.65	5.46	5.03	3.38	3.69	4.15	3.46	4.12
LP	4.92	2.42	4.34	6.46	6	5.57	5.61	5.05
LPBR	4.84	4.61	6.57	4.88	5.38	4.53	4.8	5.09
Homer	5.3	6.03	4.88	6.07	4.61	4.61	5.15	5.24
ML-C4.5	5.65	5.15	5.92	5.46	6.34	6.65	5.76	5.85

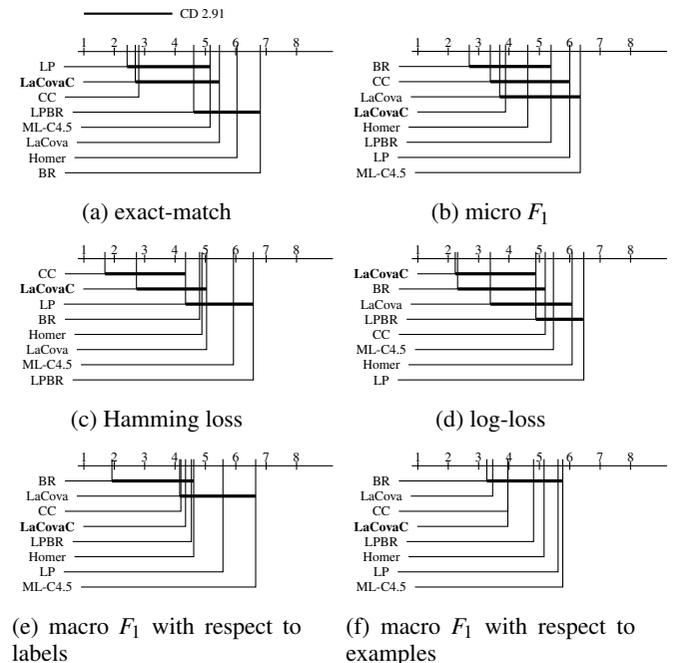


Figure 3: Critical Difference diagrams using pairwise comparisons for experiments where the Friedman test yields significance at 0.05.

Table 4: Results on 13 datasets with regards to multi-label accuracy and exact-match, the higher the value the better.

	BR	LP	CC	Homer	LPBR	ML-C4.5	LaCova	LaCovaC
multi-label accuracy								
Corel5k	0.089(2)	0.080(7)	0.083(6)	0.092(1)	0.023(8)	0.087(4)	0.084(5)	0.088(3)
Cal500	0.201(6)	0.204(5)	0.208(4)	0.197(7)	0.067(8)	0.278(1)	0.254(2)	0.209(3)
Bibtex	0.259(5)	0.244(6)	0.276(3)	0.208(7)	0.296(1)	0.176(8)	0.267(4)	0.280(2)
Language log	0.245(2)	0.234(4)	0.247(1)	0.233(5)	0.184(8)	0.224(7)	0.235(3)	0.232(6)
Enron	0.296(3)	0.277(5)	0.315(2)	0.281(4)	0.320(1)	0.207(8)	0.256(6)	0.246(7)
Medical	0.729(4)	0.724(6)	0.728(5)	0.700(7)	0.745(2)	0.363(8)	0.732(3)	0.762(1)
Genebase	0.901(8)	0.980(3)	0.986(1)	0.952(5)	0.955(4)	0.946(7)	0.950(6)	0.981(2)
Slashdot	0.425(4)	0.420(5)	0.429(3)	0.379(6)	0.376(7)	0.372(8)	0.439(2)	0.449(1)
Birds	0.491(3)	0.453(7)	0.486(4)	0.493(2)	0.411(8)	0.466(6)	0.481(5)	0.499(1)
Yeast	0.391(6.5)	0.396(4.5)	0.397(3)	0.381(8)	0.411(2)	0.396(4.5)	0.416(1)	0.391(6.5)
Flags	0.534(6)	0.531(7)	0.540(5)	0.544(4)	0.509(8)	0.572(1)	0.556(3)	0.561(2)
Emotions	0.402(4)	0.416(2)	0.388(7)	0.390(6)	0.417(1)	0.406(3)	0.400(5)	0.384(8)
Scene	0.403(6)	0.432(2.5)	0.464(1)	0.380(7)	0.421(5)	0.366(8)	0.432(2.5)	0.429(4)
Average ranks	4.57	4.92	3.46	5.30	4.84	5.65	3.65	3.57
exact-match								
Corel5k	0.000(8)	0.008(2)	0.006(3)	0.004(4.5)	0.002(6.5)	0.002(6.5)	0.004(4.5)	0.018(1)
Cal500	0.000(4.5)	0.000(4.5)	0.000(4.5)	0.000(4.5)	0.000(4.5)	0.000(4.5)	0.000(4.5)	0.000(4.5)
Bibtex	0.079(8)	0.137(2)	0.122(4)	0.084(7)	0.123(3)	0.092(6)	0.108(5)	0.149(1)
Language log	0.184(5)	0.205(1)	0.203(2)	0.183(6)	0.153(8)	0.195(4)	0.177(7)	0.197(3)
Enron	0.022(7)	0.079(1)	0.076(2)	0.031(5.5)	0.045(3)	0.020(8)	0.031(5.5)	0.032(4)
Medical	0.607(6)	0.642(3)	0.641(4)	0.598(7)	0.655(2)	0.293(8)	0.625(5)	0.673(1)
Genebase	0.809(8)	0.962(3)	0.971(1)	0.921(6)	0.938(4)	0.923(5)	0.914(7)	0.967(2)
Slashdot	0.296(7)	0.368(2)	0.356(3)	0.262(8)	0.306(6)	0.308(5)	0.353(4)	0.392(1)
Birds	0.398(4)	0.376(7)	0.410(1)	0.404(3)	0.339(8)	0.379(6)	0.395(5)	0.407(2)
Yeast	0.035(8)	0.134(1)	0.118(4)	0.050(7)	0.122(3)	0.109(5)	0.107(6)	0.124(2)
Flags	0.119(8)	0.202(2)	0.182(3)	0.155(5)	0.145(6)	0.207(1)	0.135(7)	0.171(4)
Emotions	0.125(8)	0.202(1)	0.157(4)	0.142(7)	0.165(3)	0.179(2)	0.143(5.5)	0.143(5.5)
Scene	0.284(7)	0.393(2)	0.416(1)	0.277(8)	0.375(3)	0.307(6)	0.354(5)	0.366(4)
Average ranks	6.80	2.42	2.80	6.03	4.61	5.15	5.46	2.69

Table 5: Results on 13 datasets with regards to Hamming loss and log-loss, the lower the value the better.

	BR	LP	CC	Homer	LPBR	ML-C4.5	LaCova	LaCovaC
Hamming loss								
Corel5k	0.017(5.5)	0.016(3)	0.011(1)	0.016(3)	0.023(7)	0.203(8)	0.016(3)	0.017(5.5)
Cal500	0.223(7)	0.201(5)	0.189(2.5)	0.204(6)	0.540(8)	0.188(1)	0.189(2.5)	0.198(4)
Bibtex	0.022(5.5)	0.020(4)	0.017(1)	0.022(5.5)	0.018(2.5)	0.026(8)	0.024(7)	0.018(2.5)
Language log	0.031(6)	0.026(3)	0.023(1)	0.027(4)	0.069(8)	0.029(5)	0.033(7)	0.025(2)
Enron	0.082(6)	0.078(4.5)	0.062(1)	0.078(4.5)	0.077(2.5)	0.202(8)	0.085(7)	0.077(2.5)
Medical	0.013(3.5)	0.014(5.5)	0.011(2)	0.016(7)	0.013(3.5)	0.163(8)	0.014(5.5)	0.010(1)
Genebase	0.008(4.5)	0.002(2.5)	0.001(1)	0.008(4.5)	0.019(7.5)	0.019(7.5)	0.012(6)	0.002(2.5)
Slashdot	0.055(3)	0.058(4.5)	0.045(2)	0.075(7)	0.086(8)	0.066(6)	0.058(4.5)	0.028(1)
Birds	0.072(3)	0.076(5.5)	0.063(1.5)	0.075(4)	0.113(8)	0.077(7)	0.076(5.5)	0.063(1.5)
Yeast	0.296(7)	0.288(4)	0.281(2)	0.290(6)	0.363(8)	0.289(5)	0.276(1)	0.285(3)
Flags	0.307(6)	0.309(7)	0.302(5)	0.301(4)	0.332(8)	0.273(1)	0.293(3)	0.280(2)
Emotions	0.296(2)	0.304(3)	0.286(1)	0.305(4.5)	0.309(7.5)	0.305(4.5)	0.309(7.5)	0.306(6)
Scene	0.193(3.5)	0.200(5)	0.185(1)	0.193(3.5)	0.238(7)	0.294(8)	0.201(6)	0.188(2)
Average ranks	4.80	4.34	1.69	4.88	6.57	5.92	5.03	2.73
log-loss								
Corel5k	0.048(1)	0.101(5)	0.070(4)	0.206(6)	0.275(8)	0.216(7)	0.064(3)	0.061(2)
Cal500	0.481(4)	0.786(7)	0.741(6)	0.586(5)	0.853(8)	0.381(1)	0.407(3)	0.394(2)
Bibtex	0.068(1)	0.158(6)	0.132(5)	0.194(7)	0.075(3)	0.220(8)	0.078(4)	0.070(2)
Language log	0.080(1)	0.130(6)	0.113(4)	0.165(7)	0.122(5)	0.349(8)	0.085(2)	0.093(3)
Enron	0.197(1)	0.402(7)	0.353(5)	0.367(6)	0.230(3)	0.437(8)	0.261(4)	0.218(2)
Medical	0.035(1)	0.064(5)	0.052(3.5)	0.084(7)	0.052(3.5)	0.364(8)	0.066(6)	0.037(2)
Genebase	0.008(3)	0.007(2)	0.005(1)	0.022(6)	0.012(4)	0.064(8)	0.032(7)	0.017(5)
Slashdot	0.163(1)	0.346(7)	0.269(5)	0.333(6)	0.241(4)	0.368(8)	0.204(3)	0.171(2)
Birds	0.181(2)	0.262(8)	0.217(5)	0.238(6)	0.215(4)	0.239(7)	0.190(3)	0.180(1)
Yeast	0.671(4)	1.583(8)	1.540(7)	0.900(6)	0.799(5)	0.585(2)	0.566(1)	0.603(3)
Flags	0.619(4)	0.918(8)	0.897(7)	0.716(6)	0.713(5)	0.546(1)	0.616(3)	0.559(2)
Emotions	0.640(4)	1.243(7)	1.245(8)	0.832(6)	0.815(5)	0.603(1)	0.630(3)	0.619(2)
Scene	0.492(3)	1.095(8)	1.014(7)	0.726(5)	0.733(6)	0.530(4)	0.482(2)	0.475(1)
Average ranks	2.30	6.46	5.19	6.07	4.88	5.46	3.38	2.23

Table 6: Results on 13 datasets with regards to micro F_1 , macro f_1 and macro f_e , the higher the value the better.

	BR	LP	CC	Homer	LPBR	ML-C4.5	LaCova	LaCovaC
micro F_1								
Corel5k	0.149(2)	0.114(6)	0.148(3)	0.154(1)	0.038(7)	0.031(8)	0.141(4)	0.126(5)
Cal500	0.334(5)	0.332(6)	0.338(4)	0.328(7)	0.182(8)	0.420(1)	0.397(2)	0.341(3)
Bibtex	0.345(4)	0.285(6)	0.361(2)	0.272(7)	0.391(1)	0.195(8)	0.311(5)	0.351(3)
Language log	0.175(2)	0.122(7)	0.177(1)	0.158(3)	0.123(6)	0.106(8)	0.152(4)	0.142(5)
Enron	0.428(2)	0.364(7)	0.426(3)	0.399(4)	0.442(1)	0.313(8)	0.373(5)	0.366(6)
Medical	0.778(3)	0.744(6)	0.786(2)	0.726(7)	0.774(4)	0.355(8)	0.761(5)	0.806(1)
Genebase	0.921(4)	0.982(2)	0.988(1)	0.920(5)	0.853(7)	0.841(8)	0.893(6)	0.981(3)
Slashdot	0.506(2)	0.429(6)	0.512(1)	0.456(5)	0.381(8)	0.383(7)	0.468(4)	0.496(3)
Birds	0.366(1)	0.295(7)	0.312(6)	0.334(4)	0.281(8)	0.330(5)	0.336(3)	0.346(2)
Yeast	0.541(2)	0.522(8)	0.528(5)	0.531(3)	0.530(4)	0.525(6)	0.549(1)	0.523(7)
Flags	0.690(5)	0.677(7)	0.680(6)	0.696(4)	0.665(8)	0.718(1)	0.704(3)	0.706(2)
Emotions	0.539(2)	0.521(4)	0.499(8)	0.520(5)	0.540(1)	0.511(6.5)	0.526(3)	0.511(6.5)
Scene	0.493(1)	0.445(6)	0.487(2)	0.466(5)	0.429(7)	0.380(8)	0.473(3)	0.472(4)
Average ranks	2.69	6.00	3.38	4.61	5.38	6.34	3.69	3.88
macro f_1								
Corel5k	0.040(1)	0.030(2)	0.027(4)	0.024(6)	0.012(8)	0.021(7)	0.029(3)	0.026(5)
Cal500	0.161(2)	0.147(6.5)	0.146(8)	0.156(3.5)	0.168(1)	0.147(6.5)	0.155(5)	0.156(3.5)
Bibtex	0.275(2)	0.193(6)	0.258(3)	0.147(7)	0.280(1)	0.118(8)	0.240(5)	0.248(4)
Language log	0.068(1)	0.039(6)	0.057(2)	0.050(4.5)	0.050(4.5)	0.038(7.5)	0.054(3)	0.038(7.5)
Enron	0.132(1)	0.090(7.5)	0.122(3)	0.123(2)	0.111(4)	0.098(5)	0.097(6)	0.090(7.5)
Medical	0.342(1)	0.311(6)	0.329(3)	0.304(7)	0.323(4)	0.172(8)	0.314(5)	0.334(2)
Genebase	0.518(4)	0.556(1.5)	0.556(1.5)	0.510(5)	0.504(6)	0.474(8)	0.492(7)	0.544(3)
Slashdot	0.332(1)	0.267(6)	0.315(2)	0.280(5)	0.235(8)	0.239(7)	0.294(4)	0.310(3)
Birds	0.208(1)	0.140(8)	0.143(7)	0.182(4)	0.164(5.5)	0.164(5.5)	0.190(2)	0.185(3)
Yeast	0.394(2)	0.372(7)	0.381(5)	0.389(3)	0.406(1)	0.359(8)	0.385(4)	0.379(6)
Flags	0.600(6)	0.601(5)	0.581(7)	0.616(3)	0.567(8)	0.622(1)	0.610(4)	0.620(2)
Emotions	0.530(1)	0.503(4)	0.488(8)	0.502(5)	0.527(2)	0.494(7)	0.511(3)	0.495(6)
Scene	0.244(2)	0.221(7)	0.247(1)	0.233(5)	0.223(6)	0.213(8)	0.236(3)	0.235(4)
Average ranks	1.92	5.57	4.19	4.61	4.53	6.65	4.15	4.34
macro f_e								
Corel5k	0.140(2)	0.115(7)	0.121(6)	0.143(1)	0.032(8)	0.131(3)	0.127(4)	0.125(5)
Cal500	0.330(5)	0.327(6)	0.334(4)	0.323(7)	0.119(8)	0.424(1)	0.393(2)	0.335(3)
Bibtex	0.339(4)	0.292(6)	0.341(2)	0.266(7)	0.369(1)	0.216(8)	0.338(5)	0.340(3)
Language log	0.133(1)	0.104(6)	0.124(2)	0.115(4)	0.096(7)	0.094(8)	0.121(3)	0.106(5)
Enron	0.416(2.5)	0.368(5)	0.416(2.5)	0.392(4)	0.434(1)	0.297(8)	0.360(6)	0.342(7)
Medical	0.770(3)	0.752(6)	0.757(5)	0.735(7)	0.776(2)	0.389(8)	0.769(4)	0.793(1)
Genebase	0.931(8)	0.985(3)	0.990(1)	0.962(4)	0.961(5)	0.952(7)	0.960(6)	0.986(2)
Slashdot	0.473(1)	0.438(5)	0.454(4)	0.422(6)	0.408(7)	0.394(8)	0.471(2)	0.469(3)
Birds	0.182(1)	0.142(6)	0.138(7.5)	0.150(4)	0.138(7.5)	0.146(5)	0.172(2)	0.160(3)
Yeast	0.520(3)	0.495(8)	0.502(4)	0.500(6)	0.526(1)	0.501(5)	0.524(2)	0.498(7)
Flags	0.647(6)	0.636(7)	0.650(5)	0.655(4)	0.618(8)	0.678(1)	0.673(3)	0.674(2)
Emotions	0.500(2)	0.494(3)	0.469(7.5)	0.483(6)	0.505(1)	0.487(5)	0.488(4)	0.469(7.5)
Scene	0.446(4)	0.445(5)	0.481(1)	0.416(7)	0.441(6)	0.394(8)	0.460(2)	0.451(3)
Average ranks	3.26	5.61	3.96	5.15	4.80	5.76	3.46	3.96

5 Concluding Remarks

We presented a novel decision tree algorithm for multi-label classification called **LaCovaC**. The key idea of this algorithm is to compute the label correlation matrix at each node of the tree in order to identify label correlations and then cluster them locally. Its main innovation is to introduce vertical splits that separate locally independent labels, in addition to the traditional feature-based horizontal splits that divide the instance space as in the standard decision tree algorithms.

To evaluate **LaCovaC** we compared it to state-of-the-art approaches. We used seven common evaluation metrics and 13 datasets. **LaCovaC** has the best average rank for log-loss and the second best for multi-label accuracy, exact-match and Hamming loss (without significant loss). For exact-match, **LaCovaC** shows a comparable performance to the best algorithm LP, which is a strong baseline for exact-match and hard to beat. In addition, it outperforms LP on 6 of the 9 largest datasets in terms of number of labels. This suggests that combining LP and BR as in **LaCovaC** leads to improvement over those metrics and reduces the overfitting risk.

Overall, the main strength of **LaCovaC** is its strong performance across all these metrics, as can be seen in the right-most column of Table 3. Furthermore, on each of the metrics individually **LaCovaC** is not significantly worse than the top contender. It is widely recognised in the literature that different multi-label evaluation metrics measure different things; hence there is value in demonstrating that an algorithm performs well across the board.

Several directions can be taken for further work. We plan to investigate the parameter configuration for **LaCovaC**, for example, a stopping criterion for the clustering algorithm. Moreover, it would

be interesting to investigate alternative clustering approaches, such as spectral clustering. Furthermore, to counteract the large variance associated with decision tree learning – which carries over to the proposed model – we could use bootstrap aggregates as in random forests. Finally, using the significance threshold on correlation balances the sample size and the strength of the correlation, such that both low correlation on a large sample size and high correlation on a small sample are detected as significant. However, finding an even better balance between effect size and sample size is an interesting future research direction.

Acknowledgments

Reem Al-Otaibi's PhD study is sponsored by King Abdulaziz University, Saudi Arabia. This work was partly supported by the REFRAME project granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences and Technologies ERA-Net (CHISTERA), and funded by the Engineering and Physical Sciences Research Council in the UK under grant EP/K018728/1. We acknowledge the comments of the anonymous reviewers which helped us improve the paper.

REFERENCES

- [1] Reem Al-Otaibi, Peter Flach, and Meelis Kull, 'Multi-label classification: A comparative study on threshold selection methods', in *First International Workshop on Learning over Multiple Contexts (LMCE) at ECML-PKDD 2014*, Nancy, France, (September 2014).
- [2] Reem Al-Otaibi, Meelis Kull, and Peter Flach, 'LaCova: A tree-based multi-label classifier using label covariance as splitting criterion', in *Proceedings of the IEEE 13th International Conference on Machine*

- Learning and Application (ICMLA-2014)*, pp. 74–79, Detroit, USA, (December 2014). IEEE.
- [3] Alan H. Cheetham and Joseph E. Hazel, ‘Binary (Presence-Absence) Similarity Coefficients’, *Journal of Paleontology*, **43**(5), 1130–1136, (1969).
- [4] Lena Chekina, Dan Gutfreund, Aryeh Kontorovich, Lior Rokach, and Bracha Shapira, ‘Exploiting label dependencies for improved sample complexity’, *Machine Learning*, **91**(1), 1–42, (April 2013).
- [5] Amanda Clare, A. Clare, and Ross D. King, ‘Knowledge discovery in multi-label phenotype data’, in *Lecture Notes in Computer Science*, pp. 42–53. Springer, (2001).
- [6] Pablo Nascimento da Silva, Eduardo Corrêa Gonçalves, Alexandre Plastino, and Alex Freitas, ‘Distinct chains for different instances: An effective strategy for multi-label classifier chains’, in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, volume 8725, 453–468, Springer Berlin Heidelberg, (2014).
- [7] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier, ‘Bayes optimal multilabel classification via probabilistic classifier chains.’, in *Proceedings of the International Conference on Machine Learning (ICML10)*, eds., Johannes Fürnkranz and Thorsten Joachims, pp. 279–286. Omnipress, (2010).
- [8] Janez Demšar, ‘Statistical comparisons of classifiers over multiple data sets’, *Journal of Machine Learning Research*, **7**, 1–30, (December 2006).
- [9] Nadia Ghamrawi and Andrew McCallum, ‘Collective multi-label classification’, in *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM ’05, pp. 195–200, New York, NY, USA, (2005). ACM.
- [10] Dejan Gjorgjevič, Gjorgji Madjarov, and Sašo Džeroski, ‘Hybrid decision tree architecture utilizing local svms for efficient multi-label learning’, *IJPRAI*, **27**(7), (2013).
- [11] Eduardo Corrêa Gonçalves, Alexandre Plastino, and Alex Freitas, ‘Simpler is better: A novel genetic algorithm to induce compact multi-label chain classifiers’, in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO ’15, pp. 559–566, New York, NY, USA, (2015). ACM.
- [12] Sheng-Jun Huang and Zhi-Hua Zhou, ‘Multi-label learning by exploiting label correlations locally’, in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, July 22-26, 2012, Toronto, Ontario, Canada., (2012).
- [13] S.B. Kotsiantis, ‘Decision trees: a recent overview’, *Artificial Intelligence Review*, **39**(4), 261–283, (2013).
- [14] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevič, and Sašo Džeroski, ‘An extensive experimental comparison of methods for multi-label learning’, *Pattern Recognition*, **45**(9), 3084–3104, (September 2012).
- [15] Jesse Read, Luca Martino, and David Luengo, ‘Efficient monte carlo methods for multi-dimensional learning with classifier chains’, *Pattern Recognition*, **47**(3), 1535–1546, (2014).
- [16] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank, ‘Classifier chains for multi-label classification’, in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD ’09, pp. 254–269, Berlin, Heidelberg, (2009). Springer-Verlag.
- [17] Lior Rokach, ‘Decision forest: Twenty years of research’, *Information Fusion*, **27**, 111–125, (2016).
- [18] Robert E. Schapire and Yoram Singer, ‘Improved boosting algorithms using confidence-rated predictions’, in *Machine Learning*, pp. 297–336, (1999).
- [19] Grigorios Tsoumakas and Ioannis Katakis, ‘Multi-label classification: An overview’, *International Journal of Data Warehousing and Mining*, **2007**, 1–13, (2007).
- [20] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas, ‘Mining multi-label data’, in *Data Mining and Knowledge Discovery Handbook*, pp. 667–685, (2010).
- [21] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas, ‘Effective and Efficient Multilabel Classification in Domains with Large Number of Labels’, in *ECML/PKDD 2008 Workshop on Mining Multi-dimensional Data*, (2008).
- [22] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas, ‘Mulan: A java library for multi-label learning’, *Journal of Machine Learning Research*, **12**, 2411–2414, (2011).
- [23] Grigorios Tsoumakas and Ioannis Vlahavas, ‘Random k-labelsets: An ensemble method for multilabel classification’, in *Proceedings of the 18th European Conference on Machine Learning*, ECML07, pp. 406–417, Berlin, Heidelberg, (2007). Springer-Verlag.
- [24] Matthijs J. Warrens, ‘On association coefficients for 2x2 tables and properties that do not depend on the marginal distributions’, *Psychometrika*, **73**(4), 777–789, (2008).
- [25] Yiming Yang, ‘A study of thresholding strategies for text categorization’, in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’01, pp. 137–145, New York, NY, USA, (2001). ACM.
- [26] Julio H. Zaragoza, L. Enrique Sucar, Eduardo F. Morales, Concha Bielza, and Pedro Larrañaga, ‘Bayesian chain classifiers for multidimensional classification’, in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI’11, pp. 2192–2197. AAAI Press, (2011).
- [27] Min-Ling Zhang and Kun Zhang, ‘Multi-label learning by exploiting label dependency’, in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’10, pp. 999–1008, New York, NY, USA, (2010). ACM.
- [28] Min-Ling Zhang and Zhi-Hua Zhou, ‘A review on multi-label learning algorithms’, *Knowledge and Data Engineering, IEEE Transactions on*, **26**(8), 1819–1837, (2014).

Get Me to My GATE on Time: Efficiently Solving General-Sum Bayesian Threat Screening Games

Aaron Schlenker and Matthew Brown and Arunesh Sinha and Milind Tambe¹ and Ruta Mehta²

Abstract. Threat Screening Games (TSGs) are used in domains where there is a set of individuals or objects to screen with a limited amount of screening resources available to screen them. TSGs are broadly applicable to domains like airport passenger screening, stadium screening, cargo container screening, etc. Previous work on TSGs focused only on the Bayesian zero-sum case and provided the MGA algorithm to solve these games. In this paper, we solve Bayesian general-sum TSGs which we prove are NP-hard even when exploiting a compact marginal representation. We also present an algorithm based upon an adversary type hierarchical tree decomposition and an efficient branch-and-bound search to solve Bayesian general-sum TSGs. With this we provide four contributions: (1) GATE, the first algorithm for solving Bayesian general-sum TSGs, which uses hierarchical type trees and a novel branch-and-bound search, (2) the Branch-and-Guide approach which combines branch-and-bound search with the MGA algorithm for the first time, (3) heuristics based on properties of TSGs for accelerated computation of GATE, and (4) experimental results showing the scalability of GATE needed for real-world domains.

1 Introduction

Screening for threats represents a significant security challenge, whether it is preventing an attack at a sports complex, interdicting illicit cargo shipping, or enforcing border protection. Such domains require finding the best way to allocate limited screening resources so as to reduce the expected consequences of a possible attack. This challenge is especially pronounced in settings where both screening efficiency and screening effectiveness are central objectives. As an example, the Transportation Security Administration (TSA) in the United States has launched the Dynamic Aviation Risk Management Solution (DARMS) initiative where risk-based approaches are being proposed to improve vital aspects of aviation security such as passenger screening [1].

Threat Screening Games (TSGs) [4] have been introduced to model screening domains where the screener must process a set of people or objects coming into an secure area, while an adversary tries to sneak in an attack through screening. An example is a terrorist trying to pass through airport screening with plastic explosives in their carry-on luggage to attack a flight from New York to Los Angeles. Indeed, airport threat screening by itself is a vast worldwide problem, emphasizing the tremendous importance of research on the TSG model; particularly given that the TSA in the United States appears

to be moving forward with this model [4]. However, even beyond airport screening the TSG model is applicable for screening of cargo in ports (again an important worldwide challenge), the screening of stadium patrons or protecting against illegal trafficking across borders. In TSGs, the screener uses different types of screening resources where each resource: (i) detects a possible attack method with different levels of effectiveness; and (ii) has a capacity limit in terms of the number of screenings that can be processed by that resource in a certain time period. The goal of the screener is to find the most effective way to allocate screenings to screening resources while being constrained by the capacity limits on the use of those resources. Further complicating the optimization is that screening resources can be combined into screening teams. The presence of screening teams introduce complex capacity constraints into the optimization problem as all teams containing the same resource must satisfy the original resource capacity constraints on that resource. This creates a difficult challenge as the most effective teams can only be used a limited number of times and selecting the passengers to apply the highest scrutiny to becomes an important problem.

The focus of this paper is on solving Bayesian general-sum TSGs. Previous research in TSGs [4] solves the game with the zero-sum assumption - a restrictive case that does not hold in many real world domains. By using the zero-sum assumption and a compact *marginal representation* the previous work[4] solves the game in polynomial time using an LP. Unfortunately, Bayesian general-sum games are NP-hard to solve even when exploiting a compact marginal representation for the screener's strategy. Hence, the problem we are solving is fundamentally more difficult than what is considered in previous research and in order for the TSG model to be extended to real world domains an approach to solve the general-sum case is necessary. Beyond [4], there has been work done on Bayesian general-sum security games [6, 8, 13] but as discussed in Section 2, TSGs differ from security games in crucial aspects. Thus, for large scale TSGs novel techniques and efficient solution algorithms need to be developed to apply the Bayesian general-sum TSG model to real world domains.

To solve Bayesian general-sum TSGs we present an algorithm that uses hierarchical adversary type trees to break a TSG down into smaller, restricted games containing a subset of the adversary types. These restricted games are solved using an efficient branch-and-bound search tree combined with MGA [4]; the solution information from these 'child' nodes are then passed up to the 'parent' nodes in the hierarchical tree where the information provides (i) infeasible strategy information for pruning, (ii) tighter bounds, and (iii) branching heuristics which provide faster computation at the parent nodes. We also provide heuristics based upon the properties of TSGs that increase the computational speed of the algorithm even further.

Thus, by improving upon techniques for solving Bayesian general-

¹ University of Southern California, United States, email: {aschlenk,mattheab,aruneshs,tambe}@usc.edu

² University of Illinois at Urbana-Champaign, United States, email: rutamehta@cs.illinois.edu

sum Stackelberg games and exploiting a compact TSG representation, we provide the following contributions: (1) GATE, the first algorithm for solving Bayesian general-sum TSGs, which uses hierarchical type trees and a novel branch-and-bound search, (2) the Branch-and-Guide approach which combines branch-and-bound search with MGA for the first time, (3) heuristics based on the properties of TSGs for accelerated computation of GATE by reducing the adversary strategy space that needs to be explored in the hierarchical type tree, and (4) solution quality and experimental results showing the scalability of GATE for large scale TSG instances.

2 Related Work

The problem of threat screening has been explored extensively in literature. However, it has been pointed out [4] that the TSG approach (which is inspired by security games) is much better than prior non-game-theoretic models in domains such as, screening for shipping containers [2], stadium patrons [15], and airport passengers [11, 12] or simple game-based models such as [19]. Specifically, previous non-game-theoretic approaches fails to model a rational adversary aiming to take advantage of vulnerabilities in the screening strategies whereas TSGs take this into account when devising screening strategies.

Furthermore, previous solution methods for solving security games fail to apply directly to our domain. This happens because TSGs (i) include a group of non-player screenees that all must be screened while a single adversary tries to pass through screening undetected, (ii) model screening resources with different efficacies and capacities that can be combined to work in teams and (iii) do not have an explicitly modeled set of targets. These are fundamental differences from traditional security games [9, 14, 16, 18] where the defender protects a set of targets against a single adversary. Previous research in TSGs provides an efficient solution method for the zero-sum case where our techniques are not helpful, but as mentioned in the introduction these techniques are not applicable to Bayesian general-sum TSGs.

In terms of solving Bayesian Stackelberg games there are three main approaches in previous literature: the Multiple LPs approach [5], the DOBSS algorithm [13], and the HSBA algorithm [8]. Multiple LPs fails to scale up for Bayesian Stackelberg games as it requires solving an exponential number of LPs [5]. DOBSS is an algorithm that exactly solves Bayesian Stackelberg games. However, it uses the normal form representation of the game which in our case is infeasible as our pure strategy space is exponentially larger than the space they were considering. HSBA represents the closest alternative solution method for our problem. HSBA breaks down the Bayesian game into smaller restricted games and solves the restricted games using an efficient branch and bound search with column generation. The solution information from these restricted games is then used to solve the original game more effectively. Unfortunately, [4] shows that column generation is not a scalable approach even for solving large scale Bayesian zero-sum TSGs. Still, in this paper we make use of some techniques in HSBA, namely breaking the game down into smaller restricted games, which we describe in more depth in Section 5 along with significant innovations for solving general-sum TSGs in Sections 5 and 6.

3 Threat Screening Game Model

A *threat screening game* (TSG) is a Stackelberg game played between the screener (leader) and an adversary (follower) in the pres-

ence of a set of non-player screenees that pass through a screening checkpoint operated by the screener. While TSGs are applicable to many domains, given that readers may be familiar with passenger screening at airports, we use examples from that domain. A TSG is composed of the following:

- *Time windows*: These represent the temporal dimension of screening as screenees arrive over time. This is modeled by slicing the game into a set of time windows W . The superscript w is used to indicate a specific time window.
- *Screenee categories*: Screenees have implicit characteristics, e.g., risk level and flight, which can be used to aggregate them into screenee categories $c \in C$ as they are functionally indistinguishable within the context of the game. The total number of screenees in each category c is N_c and the number of screenees in c that arrive at the screening checkpoint during time window w is N_c^w . A constant arrival rate is assumed for screenees within each screenee category and time window. All screenees in a category arriving in the same time window are screened equally in expectation according to the randomized screening strategy.
- *Adversary actions*: The adversary chooses a time window $w \in W$ to go through screening, a screenee category $c \in C$ to pose as during screening, and an attack method $m \in M$. An example adversary action is $w = 8:00AM-9:00AM$, $c = \{HighRisk, Flight1\}$, and $m = on-body\ explosives$.
- *Adversary types*: The adversary has implicit characteristics that cannot be chosen, e.g., TSA-assigned risk level. We consider *adversary types* $\theta \in \Theta$, which restrict the adversary to select from screenee categories $C_\theta \subset C$. The adversary knows their own type, but the screener only knows a prior distribution \mathbf{z} over the adversary types.
- *Resource types*: The set of screening resource types is R and all resources of type $r \in R$, e.g., all walk-through metal detectors, can be used to screen a combined total of at most L_r^w screenees during time window w .
- *Team types*: Screenees are screened by one or more resources types, e.g., walk through metal detector *and* x-ray machine. Each unique combination of resources types constitutes a screening team type t . The set of all valid team types, denoted by T , is given a priori.
- *Team type effectiveness*: Team types vary in their ability to detect different attack methods. For team type t , E_m^t is the probability of detection against attack method m .

Pure strategy A pure strategy P for the screener can be represented by $|W| \times |C| \times |T|$ non-negative *integer-valued* numbers $P_{c,t}^w$, where each $P_{c,t}^w$ is the number of screenees in c assigned to be screened by team type t during time window w . Pure strategy P must assign every screenee to a team type while satisfying the resource type capacity constraints for each time window, via the following constraints:

$$\sum_{t \in T} I_r^t \sum_{c \in C} P_{c,t}^w \leq L_r^w \quad \forall w, \forall r \quad (1)$$

$$\sum_{t \in T} P_{c,t}^w = N_c^w \quad \forall w, \forall c \quad (2)$$

where I_r^t is an indicator function returning 1 if team type t contains resource type r and 0 otherwise. We denote the set of all valid pure strategies as \hat{P} and we assume $\hat{P} \neq \emptyset$, i.e., it is possible to assign every screenee to a team type.

For example, consider a game with one time window w_1 and two screening resources r_1, r_2 with capacity constraints $L_{r_1, r_2} = 20$, respectively. The resources are combined into three screening teams $t_1 = \{r_1\}$, $t_2 = \{r_1, r_2\}$ and $t_3 = \{r_3\}$. There are three categories

c_1 , c_2 and c_3 and each has $N_c = 9$ passengers. Figure 1(a) shows an example of a pure strategy allocation in this game.

The pure strategies for the adversary types are denoted as $a_{c,m}^{\theta,w}$ which specifies that adversary type θ selects time window w , screening category c , and attack method m .

	t_1	t_2	t_3		t_1	t_2	t_3
c_1	4	2	3	c_1	4.3	1.5	3.2
c_2	1	4	4	c_2	0.7	4.1	4.2
c_3	3	5	1	c_3	3	5.4	0.6
(a) Pure Strategy				(b) Marginal Strategy			

Figure 1. TSG Strategies

In the security games literature, two approaches are commonly used to handle scale-up: marginal strategies [9, 10] and column generation [6, 20]. To solve large-scale zero-sum TSGs [4] employed a marginal-based approach, showing that such an approach significantly outperformed the use of column generation. Hence we continue with the use of marginal strategies.

Marginal strategy A marginal strategy \mathbf{n} for the screener can be represented by $|W| \times |C| \times |T|$ non-negative *real-valued* numbers $n_{c,t}^w$, where $n_{c,t}^w$ is the number of screenees in c assigned to be screened by team type t during time window w . We would want this marginal strategy to be a valid mixed strategy (*implementable*), i.e., there should exist a probability distribution over \hat{P} given by q_P (i.e., $\sum_{P \in \hat{P}} q_P = 1$, $0 \leq q_P \leq 1$) such that $n_{c,t}^w = \sum_P q_P P_{c,t}^w$. An example marginal screener strategy is shown in Figure 1(b) with the same game parameters described previously.

Utilities Since all screenees in category c are screened equally in expectation, we can interpret $n_{c,t}^w/N_c^w$ as the probability that a screenee in category c arriving during time window w will be screened by team type t . Then, the probability of detecting an adversary type in category c during time window w using attack method m is given by $x_{c,m}^w = \sum_t E_m^t n_{c,t}^w/N_c^w$. The payoffs for the screener are given in terms of whether adversary type θ chooses screening category c and is either detected during screening, denoted as $U_{s,c}^d$, or is undetected during screening, denoted as $U_{s,c}^u$. Similarly, the payoffs for adversary type θ are given in terms of whether θ chooses screening category c and is either detected during screening, denoted as $U_{\theta,c}^d$, or is undetected during screening, denoted as $U_{\theta,c}^u$. Given adversary type θ pure strategy $a_{c,m}^{\theta,w}$, the screener's expected utility is given by $U_s(a_{c,m}^{\theta,w}) = x_{c,m}^w U_{s,c}^d + (1 - x_{c,m}^w) U_{s,c}^u$ and the expected utility for adversary type θ is given by $U_{\theta}(a_{c,m}^{\theta,w}) = x_{c,m}^w U_{\theta,c}^d + (1 - x_{c,m}^w) U_{\theta,c}^u$.

4 Approach

While it has been shown that Bayesian Stackelberg games are hard to solve [5], it is interesting to observe that Bayesian general-sum TSGs are hard to solve even in the marginal strategy space as we prove next. This result shows that finding the optimal marginal strategy \mathbf{n} for Bayesian general-sum TSGs is fundamentally more complex than the zero-sum case which can be solved in polynomial time as an LP. Thus, it is not surprising that the solution approaches used in [4] are not directly applicable to the general-sum case.

Theorem 1. *Finding the optimal solution in Bayesian general-sum TSGs is NP-hard even in the relaxed marginal strategy space.*

Proof. We reduce from the knapsack problem to our problem. Assume n items with weights w_i and value v_i with a sack of capacity K . Wlog, assume w_i and K are integers. Construct a game with n adversary types $|\Theta| = n$. Each type of adversary has two flights to board: f_0, f_1 . Thus, $C = \{(\theta_i, f_j) \mid 0 \leq i \leq n, j \in \{0, 1\}\}$. Choose the other parameters of the game as follows: two resources r_1, r_2 with capacities $L_{r_1} = K$ and $L_{r_2} = \infty$. We have two teams $t_1 = \{r_1\}$ and $t_2 = \{r_2\}$. There is only one attack method m_1 . For this game $E_{m_1}^{t_1} = 1$ and $E_{m_1}^{t_2} = 0$, i.e., t_2 is not effective at all at detecting m_1 . The number of screenees for each screening category $N_{(\theta_i, f_0)} = w_i$ and $N_{(\theta_i, f_1)} = 1$ for all $\theta_i \in \Theta$. Each type θ_i occurs with probability $\frac{v_i}{\sum_{\theta_i} v_i}$.

The utilities for the screener are $U_{s,(\theta_i, f_0)}^d = 0, U_{s,(\theta_i, f_0)}^u = 0, \forall \theta_i$ and $U_{s,(\theta_i, f_1)}^d = 2, U_{s,(\theta_i, f_1)}^u = 1, \forall \theta_i$. Thus, the screener strictly prefers the adversary of every type to choose f_1 . The utilities for the adversary are set as follows: $U_{\Theta,(\theta_i, f_0)}^d = 1, U_{\Theta,(\theta_i, f_0)}^u = 2, \forall \theta_i$ and $U_{\Theta,(\theta_i, f_1)}^d = 0, U_{\Theta,(\theta_i, f_1)}^u = 1, \forall \theta_i$. Thus, the adversary will choose (θ_i, f_1) only when the probability of detection $x_{(\theta_i, f_0)} = 1, x_{(\theta_i, f_1)} = 0$ (breaking ties in favor of the screener). This happens only when all of the screenees $N_{(\theta_i, f_0)} = w_i$ are screened by t_1 . Thus, we have from the capacity constraints that $\sum_{\theta_i} w_i \leq K$. Therefore, for the choice of f_1 by adversary of type θ_i , the screener earns $\frac{v_i}{\sum_{\theta_i} v_i}$ and otherwise the screener earns 0. Given, the optimization problem maximizes this utility: $\sum_{\theta_i} w_i \frac{v_i}{\sum_{\theta_i} v_i}$, the optimization provides a solution for the knapsack problem.

The resulting TSG instance from the knapsack problem has two flights, two resources, two teams, and as many adversary types as the number of items n in the knapsack problem. The screenee types assigned to t_1 gives the knapsack solution. Therefore, the reduction from the knapsack problem is overall polynomial in the number of items n . \square

The optimal marginal strategy for a Bayesian general-sum TSG can be obtained by solving the mixed integer linear program *MarginalStrategyMILP*, provided below.

$$\max_{\mathbf{n}, \mathbf{s}, \mathbf{x}, \mathbf{a}} \sum_{\theta \in \Theta} z_{\theta} s_{\theta} \quad (3)$$

$$s_{\theta} - U_s(a_{c,m}^{\theta,w}) \leq (1 - a_{c,m}^{\theta,w}) \cdot Z \quad \forall \theta, w, c, m \quad (4)$$

$$0 \leq k_{\theta} - U_{\theta}(a_{c,m}^{\theta,w}) \leq (1 - a_{c,m}^{\theta,w}) \cdot Z \quad \forall \theta, w, c, m \quad (5)$$

$$x_{c,m}^w = \sum_{t \in T} E_m^t \frac{n_{c,t}^w}{N_c^w} \quad \forall w, c, m \quad (6)$$

$$\sum_{t \in T} I_r^t \sum_{c \in C} n_{c,t}^w \leq L_r^w \quad \forall w, r \quad (7)$$

$$\sum_{t \in T} n_{c,t}^w = N_c^w \quad \forall w, c \quad (8)$$

$$n_{c,t}^w \geq 0 \quad \forall w, c, t \quad (9)$$

$$a_{c,m}^{\theta,w} \in \{0, 1\} \quad \forall \theta, w, c, m \quad (10)$$

$$\sum_{w,c,m} a_{c,m}^{\theta,w} = 1 \quad \forall \theta \quad (11)$$

Equations 10 and 11 force each adversary type θ to choose a pure strategy. Equation 3 is the objective function which maximizes the screener expected utility as a weighted summation of screener expected utility against adversary type θ , s_{θ} , multiplied by the probability of encountering adversary type θ , z_{θ} . Equation 4 defines the

screeener's expected payoff against each adversary type, contingent on the choice of pure strategies by the adversary types. The constraint places an upper bound of $U_s(a_{c,m}^{\theta,w})$ on s_θ , but only if $a_{c,m}^{\theta,w} = 1$, as Z denotes an arbitrarily large constant. For all other pure strategies of adversary type θ , the RHS is arbitrarily large. Similarly, Equation 5 places an upper bound of $U_\theta(a_{c,m}^{\theta,w})$ on the utility of adversary type θ , k_θ , for $a_{c,m}^{\theta,w} = 1$. Additionally, Equation 5 also lower bounds k_θ by the largest $U_\theta(a_{c,m}^{\theta,w})$ over all $a_{c,m}^{\theta,w}$. Taken together these upper and lower bounds ensure that the pure strategy selected by adversary type θ is a best response to the screener marginal strategy \mathbf{n} . Equations 7-9 requires that \mathbf{n} satisfies the resource type capacity constraints (i.e., Equation 1) and screeenee category assignment constraints (i.e., Equation 2).

Even though *MarginalStrategyMILP* represents a NP-hard problem, solving it does not necessarily produce an implementable marginal screener strategy, i.e., the marginal strategy may not map to a probability distribution over pure strategies. The issue of implementability was addressed in [4] for TSGs by introducing the Marginal Guided Algorithm (MGA), which uses the (potentially non-implementable) marginal strategy to additionally restrict the constraints of a TSG to obtain a provably implementable marginal strategy though it can possibly lose solution quality in the process.

5 GATE: Solving Bayesian General-Sum TSG

Despite operating in the marginal screener strategy space, solving the *MarginalStrategyMILP* is computationally expensive due to the presence of the integer variables that encode the adversary strategy space. Therefore, we instead seek to solve Bayesian general-sum TSGs using an algorithmic approach for exploring the adversary strategy space. An example of the adversary strategy space for two adversary types and two actions per type is shown in Figure 2(a). The leaf nodes in this tree represent all possible joint pure strategy combinations for the two adversary types.

As mentioned previously a standard algorithmic approach that could be used to solve Bayesian general-sum TSGs is Multiple LPs [5]. This technique exploits the fact that, by fixing the joint adversary pure strategy, the underlying optimization problem is converted from a MILP to an LP. Thus, an LP can be solved for each leaf node in the adversary strategy tree to obtain the best marginal screener strategy which induces that joint adversary pure strategy as a best response. The marginal strategy with the highest screener utility is returned as the solution for the TSG. However, the number of joint adversary pure strategies is exponential in the number of adversary types Θ and thus Multiple LPs is not scalable for large problem instances. Therefore, we propose the General-sum Algorithm for Threat screening game Equilibria (GATE) and in this section we provide an intuitive, high level description. In Section 6 we provide a detailed algorithm as our experimental results show GATE does not scale leading to the need for heuristics to further speed up computation.

5.1 Hierarchical Type Trees

At a high level GATE seeks to exploit the structure of TSGs and reduce the number of joint adversary pure strategies that need to be evaluated. GATE achieves this reduction by building off intuition from the HBSA algorithm [8], which involves constructing a hierarchical type tree. Such a tree decomposes the game with each node in the tree corresponding to a restricted game over a subset of adversary types. The idea is to solve these smaller, restricted games to efficiently obtain (1) infeasibility information to eliminate large sets

of joint adversary pure strategies, and (2) utility upper bound information that can be used to terminate the evaluation of joint adversary pure strategies.

GATE operates on restricted TSGs, where we define $TSG(\Theta')$ to be a TSG with a subset of adversary types $\Theta' \subset \Theta$. It is important to note that, despite not including all adversary types, $TSG(\Theta')$ does not ignore the screeenee categories associated with adversary types $\Theta \setminus \Theta'$. Indeed, $TSG(\Theta')$ must still satisfy the constraint that all screeenees in each category must be assigned to a screening team type, i.e., Equation 8. By continuing to enforce these constraints, the upper bounds generated will be tighter as the screener cannot focus all the screening resources on just a subset of screeenee categories, helping to improve the ability of GATE to prune out joint adversary pure strategies.

The subsets of adversary types are decomposed such that each level in the hierarchical type tree forms a partition over Θ satisfying $\Theta'_i \cap \Theta'_j = \emptyset, \forall i, \forall j, i \neq j$ as well as $\cup_i \Theta'_i = \Theta$. Additionally, the set of adversary types in each parent node is the union of the sets of adversary types of all of its children, with the root node of the hierarchical type tree corresponding to the full problem. Figure 2(b) shows an example of a hierarchical tree structure with full binary partitioning for a game with four adversary types. The root node is the parent to two restricted games with two adversary types, each of which is a parent to two restricted games for individual adversary types.

The evaluation of the hierarchical tree starts at the leaf nodes and works up the tree such that all child nodes are evaluated before the parent nodes are evaluated. Every node is processed by evaluating the pure strategies of the restricted game and propagating up only the feasible pure strategies (i.e., pure strategies inducible as an adversary best response). [8] proved that if a pure strategy $a_{\theta'}$ can never be a best response for adversary type θ' in a restricted game $TSG(\Theta')$ with $\Theta' = \{\theta'\}$ then any joint pure strategy containing $a_{\theta'}$ can never be a best response in any $TSG(\Theta'')$ with $\Theta' \subset \Theta''$. Thus, at a given node, it is only necessary to consider joint pure strategies in the cross product of the sets of feasible pure strategies passed up from the child nodes.

For each pure strategy to be propagated, the corresponding utility with respect to the restricted game is also passed up. [8] also proved that it is possible to upper bound the screener utility for joint adversary pure strategy a_Θ in $TSG(\Theta)$ by $\sum_{\theta \in \Theta} z_\theta \beta(a_\theta)$ where z_θ is the normalized probability of adversary type θ in $TSG(\Theta')$ and $\beta(a_\theta)$ is the upper bound on the screener utility for adversary pure strategy a_θ in the restricted game $TSG(\{\theta\})$. These upper bounds can be used to determine the order to evaluate joint pure strategies as well as when it no longer necessary to evaluate joint pure strategies. This propagation of pure strategies and upper bounds continues until the root node is solved to obtain the best solution for the game.

Example Consider a game with four adversary types $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4\}$, like in Figure 2(b), where each adversary type has two actions available, $a_{\theta_i}^1, a_{\theta_i}^2$. The full game is broken down into the restricted games and we solve the leaf nodes in the hierarchical tree first. Suppose that after we solve all of the leaf nodes we get the following action sets for the adversaries: $\theta_1 = \{a_{\theta_1}^1\}$, $\theta_2 = \{a_{\theta_2}^1, a_{\theta_2}^2\}$, $\theta_3 = \{a_{\theta_3}^1\}$, $\theta_4 = \{a_{\theta_4}^1\}$ (as the other actions are found to be infeasible). Then, in the node $\{\theta_1, \theta_2\}$ we evaluate $[a_{\theta_1}^1, a_{\theta_2}^2]$ and $[a_{\theta_1}^1, a_{\theta_2}^1]$ while for node $\{\theta_3, \theta_4\}$ we evaluate $[a_{\theta_3}^2, a_{\theta_4}^1]$. Now, at the root node (Θ) we only have to evaluate two joint adversary pure strategies (i.e., $[a_{\theta_1}^1, a_{\theta_2}^1, a_{\theta_3}^2, a_{\theta_4}^1]$ and $[a_{\theta_1}^1, a_{\theta_2}^2, a_{\theta_3}^2, a_{\theta_4}^1]$) instead of 16 (cross product of all adversary type strategy sets) in order to find the optimal strategy for the game.

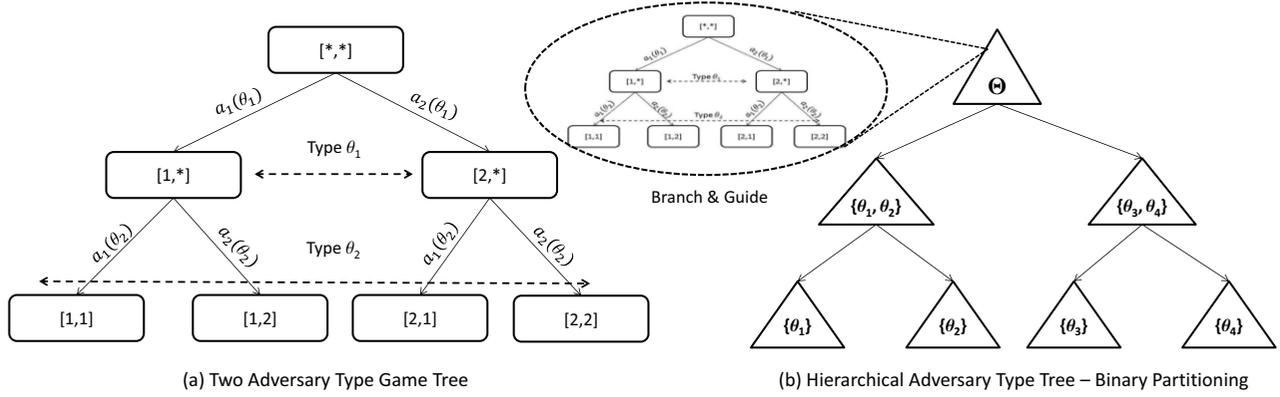


Figure 2. Bayesian Adversary Strategy Space

5.2 Advantages of GATE

As mentioned earlier, security games techniques do not apply to TSGs directly, and thus, HBSA is not well suited for our problem. In particular, HBSA utilizes a Branch-and-Price framework [3] which requires running column generation (given the large number of defender strategies in complex domains) to evaluate every single adversary joint pure strategy in the hierarchical type tree. While Branch-and-Price is a general approach frequently used for Bayesian Stackelberg games, column generation has been shown to be incapable of scaling for large-scale TSGs even in the zero-sum case due to the massive number of screener pure strategies [4]. Thus, having to repeatedly run column generation for the Bayesian general-sum TSGs is a non-starter.

To efficiently evaluate the joint adversary pure strategies in the nodes of the hierarchical tree, we introduce Branch-and-Guide, which combines branch-and-bound search with MGA to simultaneously mitigate the challenges of both scalability and implementability when solving Bayesian general-sum TSGs. Branch-and-Guide may be run at the root node of the hierarchical type tree, so that given a large of joint adversary pure strategies, we may be able to prune out a large number of them using upper-bounds, speeding up our computation. Furthermore, Branch-and-Guide exploits the fact that for a fixed joint adversary pure strategy, an implementable marginal screener strategy can be obtained quickly (even if it not necessarily optimal) and thus can avoid having to rely on column generation.

An example of the adversary strategy tree explored in Branch-and-Guide is shown in Figure 3, with the size and ordering of the tree based on the feasible joint adversary pure strategies and corresponding upper bounds propagated up by the child nodes. Branches to the left fix the joint adversary pure strategy, converting *MarginalStrategyMILP* into a linear program which can be solved efficiently. However, the resulting marginal strategy may not be implementable and thus we run MGA on the marginal while ensuring that the selected joint adversary pure strategy is still a best response. This two-step process produces an implementable marginal strategy that gives a lower bound on the overall solution quality. Branches to the right represent the upper bound on the screener utility for the next best joint adversary pure strategy which is calculated using the solution quality information passed up from the child nodes. If the screener utility for the best solution found thus far is higher than the upper bound than the next best joint adversary pure strategy, then the execution of Branch-and-Guide can be terminated without exploring the remaining joint adversary pure strategies.

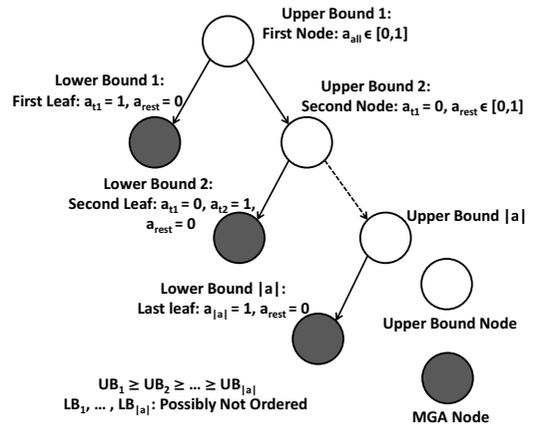


Figure 3. Branch-and-Guide Tree

6 Scaling Up GATE

While GATE incorporates state-of-the-art techniques to solve *MarginalStrategyMILP*, it fails to scale up to real world problem sizes for TSGs (see comparison in Evaluation). Thus, we employ intuitive heuristics that further narrows down the search space for GATE, thereby enabling up to 10X run time improvement with only 5-10% solution quality loss. There are two distinct steps in GATE where additional heuristics can help: the processing step at the leaves of the hierarchical type tree and the processing step at the intermediate and root nodes. In this section we first present GATE-H (GATE with heuristics) formally and then describe the heuristics used to speed up the computation.

6.1 GATE-H: GATE with Heuristics

GATE-H solves TSGs efficiently by limiting both the number of adversary pure strategies passed up the hierarchical adversary type tree from restricted games and by limiting the number of adversary strategies evaluated in the individual nodes. Each node in the hierarchical tree is solved using Algorithm 1, beginning at the leaf nodes. The feasible adversary pure strategy set, denoted as A' , is passed up to the parent nodes as each child is solved. Notice that not all strategies need be evaluated at a given node for the computation to terminate

Algorithm 1: GATE – H – NODE($\Theta, A_\Theta^i, B^i, U_s, U_\Theta, K$)

```

// $A_\Theta^i$ : Pruned feasible pure strategy set for all adversary types
1.  $A'' := \text{all-Joint-Adversary-Pure-Strategies}()$ 
2.  $B'(a_\Theta) := \text{getBound}(a_\Theta, B^i) \forall a_\Theta \in \prod_\Theta A_\Theta^i$ 
3.  $\text{sort}(A'', B'(a_\Theta)) // \text{sort } a_\Theta \text{ in descending order of } B'(a_\Theta)$ 
4.  $a_\Theta := [A_\Theta^1(1), A_\Theta^2(1), \dots, A_\Theta^{|\Theta|}(1)]$ 
5.  $r^*, r' = -\infty // \text{Save best and iterative solutions}$ 
repeat
  6.  $(\text{feasible}, n, r) := \text{MGA}(a_\Theta)$ 
    if feasible then
      7.  $A' := A' \cup a_\Theta$ 
      if  $r > r^*$  then
        8a.  $r^* := r$ 
        8b.  $n^* := n$ 
      9.  $B'(a_\Theta) := r$ 
    else
      10.  $A'' := A'' \setminus a_\Theta$ 
  11. Every  $K$  iterations
    if  $r' = r^* \neq -\infty$  then
      12. break
    else
      13.  $r' := r^*$ 
  14.  $a_\Theta := \text{getNextStrategy}(a_\Theta, r^*, A_\Theta^i, B^i)$ 
until  $a_\Theta = \text{NULL}$ 
return  $(n^*, r^*, A', B')$ 

```

as either the Branch-and-Guide heuristic or K cutoff heuristic, both introduced later in this section, can end the computation early.

When solving a given node in GATE-H we take in the adversary set (Θ) and the screener's and adversary's utilities (U_s and U_Θ , respectively), with the feasible strategies (A_Θ^i) and bound information (B^i) is acquired from the children of that node as shown in Algorithm 1. In the case of solving a leaf node in the binary tree we enumerate all of that adversary type's strategies and get bound information by solving an upper bound LP, described in Section 6.2. After constructing the joint adversary pure strategy set (Line 1), we order the set by their upper bound values (Line 3) and evaluate each strategy one by one (main loop starts after Line 5). Heuristics are used inside of this loop, leading to GATE-H.

Algorithm 2: getNextStrategy($a_\Theta, r^*, A_\Theta^i, B^i$)

```

for  $i = |\Theta|$  to 1 Step-1 do
   $j := \text{index-of}(a_\Theta, A_\Theta^i)$ 
  //Set each adversary type strategy equal to left most leaf
   $a_\Theta^i := [A_\Theta^1(1), \dots, A_\Theta^{|\Theta|-1}(1), A_\Theta^{|\Theta|}(j+1)]$ 
  if  $r^* < \text{getBound}(a_\Theta, B)$  then
    return  $a_\Theta$ 
return NULL

```

The first of the two heuristics used is the Branch-and-Guide approach (Line 14 - getNextStrategy()) which ends the computation early if the value of the current best strategy is greater than the next highest upper bound. The second heuristic, discussed in more detail in Section 6.3, is the K cutoff (Line 11) which ends the computation early if the current best solution is not improving after K iterations.

Algorithm 2 describes the getNextStrategy function in detail. Es-

entially the function builds the next strategy to be evaluated by iterating through all of the adversary types and grabbing the highest valued strategy in their respective pure strategy lists.

6.2 Tuning Leaf Node Computation

At the leaf nodes in our hierarchical type tree we solve the restricted game for each adversary type. For GATE to be exact, we must return all feasible adversary strategies from each leaf node. If we do not return all feasible pure strategies, we have a heuristic approach, which may run well in practice but is not guaranteed to be optimal. Nonetheless, in some cases even for optimality it might suffice for us it might suffice for us to return only some promising strategies. Below we note a special condition under which it is optimal to just consider one adversary strategy at each leaf in the hierarchical tree.

Lemma 1. *Let \mathbf{n}_θ represent the optimal allocation against type θ at the leaf. Let $\mathbf{n}_\theta[C_\theta]$ be the part of the allocation that is assigned to screenees in screenee category C_θ . If the strategy \mathbf{n} formed by putting together all $\mathbf{n}_\theta[C_\theta]$: $\mathbf{n} = \sum_\theta \mathbf{n}_\theta[C_\theta]$ is feasible then \mathbf{n} is the optimal defender strategy and the single adversary best response for each single type is the adversary best response in the overall game.*

Proof Sketch. First, note that the strategy \mathbf{n} achieves the payoff $\sum_\theta z_\theta s_\theta^*$, where s_θ^* is the defender utility in the restricted game with just the type θ . Also, clearly s_θ^* is an upper bound on the defender utility for the restricted game. By the result from [8], the upper bound on the defender utility is $\sum_\theta z_\theta s_\theta^*$ which is achieved by \mathbf{n} . \square

We can use Branch-and-Guide as described earlier to return fewer promising adversary strategies, but the question that arises when using Branch-and-Guide at the leaf nodes is how to compute the upper bounds for the nodes on the right of the tree (Recall for non-leaf nodes this upper bound is computed from the upper bounds that are propagated up from each child). One approach to compute this upper bound is to adapt the ORIGAMI [9] approach; ORIGAMI is the fastest technique for solving non-Bayesian general-sum security games without resource scheduling constraints. The underlying idea is to solve an LP to minimize the utility of the adversary by inducing the largest possible attack set (set of targets that are equally and most attractive for the attacker) and [9] shows this provides the optimal defender utility in games without scheduling constraints. However, even for a TSG with a single time window and a single adversary type, ORIGAMI may not provide the optimal solution.

Therefore, ORIGAMI cannot be applied directly to find upper bounds on the adversary pure strategies at the leaf nodes. Thus, we provide *UpperBoundLP*, shown below, to calculate the upper bound at the leaf nodes in the hierarchical tree.

$$\min_{\mathbf{n}, \mathbf{q}, \mathbf{s}, \mathbf{x}} k_{\theta'} \quad (12)$$

$$s.t. \quad k_{\theta'} \geq x_{c,m}^w U_{\theta',c}^d + (1 - x_{c,m}^w) U_{\theta',c}^u \quad \forall w, \forall c, \forall m \quad (13)$$

$$x_{c,m}^w = \sum_{t \in T} E_m^t \frac{n_{c,t}^w}{N_c^w} \quad \forall w, \forall c, \forall m \quad (14)$$

$$\sum_{t \in T} I_r^t \sum_{c \in C} n_{c,t}^w \leq L_r^w \quad \forall r, \forall w \quad (15)$$

$$\sum_{t \in T} n_{c,t}^w \leq N_c^w, \quad n_{c,t}^w \geq 0 \quad \forall c, \forall w \quad (16)$$

The objective function 12 minimizes the attacker's utility for the adversary type θ' . Equation 13 enforces that the adversary payoff be the maximal payoff for the adversary given a marginal strategy \mathbf{n} .

Equations 14-16 enforce the resource constraints from the original game. This LP uses a slightly modified formulation when enforcing the capacity constraints. Namely, replace Equation 7 with 16, i.e., we relax the constraint that every passenger in a given category c for a time window w must be screened.

Theorem 2. *UpperBoundLP provides an upper bound on the screener utility $d_{\theta'}$ for a non-Bayesian TSG.*

Proof. By relaxing constraint 7 to constraint 16, we can only expand the attack set of the adversary from the original formulation. This happens as Equation 16 allows for an adversary to not be screened which can only increase their utility for targets, thus possibly increasing their attack set. Since the adversary breaks ties in the screener's favor this can only increase the screener's possible utility. \square

The above approach serves two purposes: we enormously reduce the set of strategies that are sent up to the parent even if all the adversary strategies are evaluated in the Branch-and-Guide tree, as the attack set is much smaller than the set of all feasible strategies. The running time is also reduced, given the small attack set and the efficient LP to obtain this attack set and upper bounds.

6.3 Tuning Non-leaf Node Computation

Section 6.2 focused on reducing the number of adversary pure strategies returned from the leaf nodes. However, another very important area to prune the search space is at the interior nodes in the binary tree. One way to approach this problem is using Branch-and-Guide in these nodes and stopping the search once the current best solution is better than the next highest upper bound. This can possibly provide significant speed-ups in terms of the computation speed of GATE but it is not quite enough. Unfortunately, in our experiments it turned out that most of the joint adversary pure strategies were evaluated in the interior nodes by MGA as the stopping condition for Branch-and-Guide was almost never met.

Thus, we take inspiration from column generation and security games literature [17] where column generation is stopped if the solution quality does not change much with new columns being added. This heuristic almost always provides a very good approximation. We adopt this same principle so that when evaluating adversary strategies in the Branch-and-Guide approach if the current solution quality does not change over the next K strategies then we can stop the computation and declare the current solution as the final solution with the adversary strategies evaluated so far propagated to the parent. This approach can also be adopted at the root. Here K is a parameter that we can vary, but we use $K = 30$ for the experiments as it seems to work the best for our problem. This approach serves two purposes: (1) it reduces the run time at each intermediate node and the root node and (2) it reduces the number of adversary pure strategies propagated up the tree.

GATE-H then allows us to solve large-scale Bayesian general-sum TSGs. Further, empirically these heuristics maintain high solution quality while decreasing the runtime by an order of magnitude.

7 Evaluation

We evaluate GATE-H using synthetic examples from the passenger screening domain as real world data is not available. We solved the LPs and MILPs using CPLEX 12.5 with the barrier method, as this was found to work the best, on USC's HPC Linux cluster limited

to one Hewlett-Packard SL230 node with 2 processors. The adversary and screener payoffs are generated uniformly at random with $U_a^u \in [2, 11]$ and $U_s^u \in [-1, -10]$. For both the adversary and the screener we set $U^d = 0$. The default values for the experiments

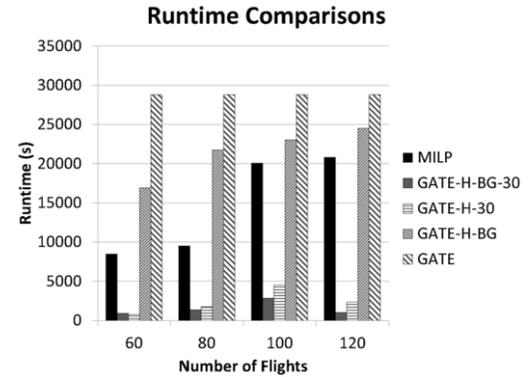


Figure 4. Runtime Comparison - MILP and GATE

are 6 adversary risk levels, 5 screening resource types, 10 screening teams, 2 attack methods and 3 time windows unless otherwise specified. For all experiments the capacity resource constraints also remain constant. All results are averaged over 20 randomly generated game instances.

Scaling Up and Solution Quality The first experiment tests the scalability of each approach and provides solution quality information for GATE-H relative to the *MarginalStrategyMILP*. This experiment provides information about the trade-off between runtime and solution quality for our heuristic algorithm. The four different variations of GATE that were tested are: (1) GATE which evaluates all adversary pure strategies in each of the restricted games and only uses Branch-and-Guide at the root, (2) GATE-H-BG which uses the Branch-and-Guide heuristic in all nodes, (3) GATE-H- K which uses the $K = 30$ cutoff in all nodes and (4) GATE-H-BG- K which uses both Branch-and-Guide and the K cutoff in all nodes. In Figure 4 we show the runtime results for all of the algorithms. On the x-axis we vary the number of flights from 60 to 120 in increments of 20. On the y-axis is the runtime in seconds. For example, for 80 flights *MarginalStrategyMILP* takes almost 10,000 seconds to finish. GATE did not finish in any of the instances showing it cannot scale to large TSG instances. GATE-H-BG also does not perform well as it fails to beat the average runtime of the MILP over all of the instances. As can be seen GATE-H-BG-30 and GATE-H-30 significantly reduce the runtime with an average of a 10 fold improvement over all cases and GATE-H-BG-30 providing a 20 fold speed up at 120 flights.

In Figure 5 we compare the solution quality of the MILP with GATE-H-BG-30 and GATE-H-30. We do not include GATE and GATE-H-BG as they do not finish in a majority of instances making it difficult to compare the solution quality. On the x-axis we again vary the number of flights from 60 to 120 in increments of 20. On the y-axis is the screener's utility. For instance, for 60 flights GATE-H-BG-30 returns an average screener utility of -0.5974. As the graph shows the average solution quality loss over these game instances is always less than .0411 for both GATE-H-BG-30 and GATE-H-30 compared to the MILP. These results show that both GATE-H-BG-30 and GATE-H-30 provide good approximations for large scale TSGs.

Our next experiment aimed to test the ability of GATE-H-BG-30 and GATE-H-30 to scale up to much larger TSG instances. The re-

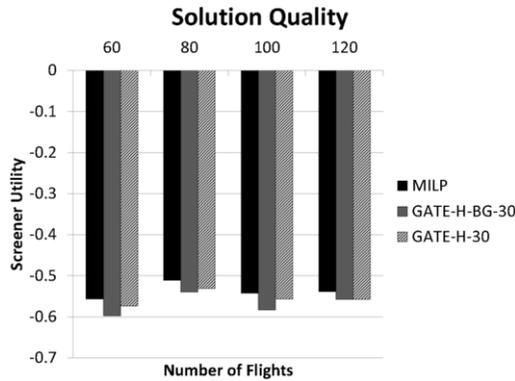


Figure 5. Solution Quality - MILP and GATE

sults are shown in Figure 6. On the x-axis we increase the number of flights from 160 to 220 in increments of 20. On the y-axis we show the average runtime in seconds to solve each of the TSG instances. For example, GATE-H-BG-30 took on average 2,396 seconds to solve a game with 200 flights. An interesting trend here is that the runtime peaks at 180 flights and starts to decrease afterward. This trend could be related to the resource saturation problem as seen in other security games [7], where the observation is that resource optimization is easiest when the resources available are comparatively small or equal to the number of targets and becomes difficult when this is not the case.

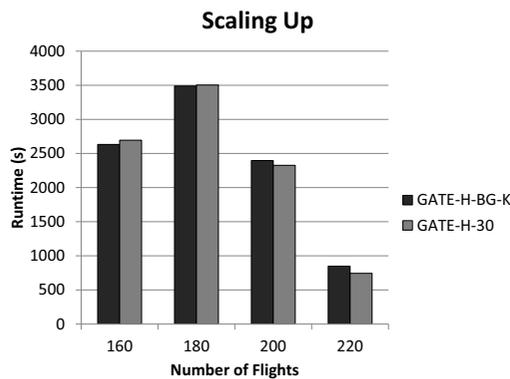


Figure 6. Scaling Up to Larger TSG Instances

Moving Towards Zero Sum The last experiment aimed to test what happens to the solution quality of GATE-H-BG-30 and GATE-H-30 as the game payoffs move toward zero-sum. For this experiment, we use 40 flights and have only one time window as the MILP does not scale in these instances. We vary an r -coefficient from 0 to -0.9 in increments of -0.1, where $r = 0$ means there is no correlation between the attacker’s and screener’s payoffs and $r = -1$ means the game is zero-sum (Note: -1 is not tested as there is a specialized algorithm to deal with that case where our techniques are not useful). In previous experiments we do not explicitly set a correlation between the adversary’s and screener’s payoffs although it may be the case. On the x-axis is the r -coefficient and the y-axis shows the screener’s utility. For example, when $r = 0$ the *MarginalStrategyMILP* returns a screener utility -0.889, GATE-H-BG-30 returns a screener utility -0.917 and GATE-H-30 returns a solution quality of -0.931. In this

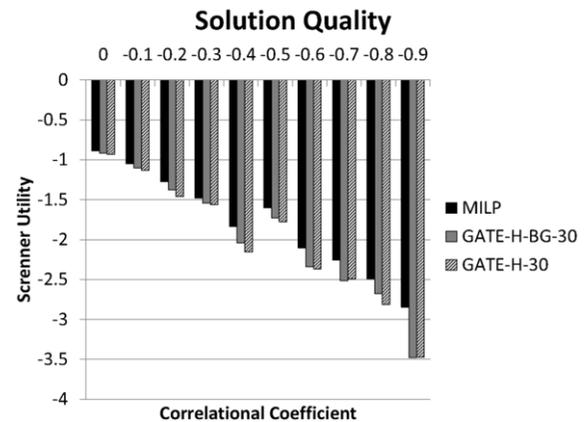


Figure 7. Solution Quality - Moving to Zero Sum

experiment an interesting trend appears. As we move toward zero-sum games the relative performance of both GATE-H-BG-30 and GATE-H-30 progressively worsens. However, until the game payoffs are nearly zero-sum ($r = -0.9$) both GATE-H variations do have a solution quality loss greater than 11.385%. This experiment again shows that both GATE-H-BG-30 and GATE-H-30 provide good approximations in general-sum TSGs. (A careful reader might notice in Figures 5 and 5 that there are slight differences in solution quality between GATE-H-BG-30 and GATE-H-30, however these are not statistically significant.)

8 Summary

The TSG model provides an extensible and adaptable model for game-theoretic screening in the real world. It improves upon previous models in security games that fail to capture important properties of the screening domain, e.g., the presence of non-player screenees in the game and complex team capacity constraints. The model also improves on work done on threat screening, such as screening stadium patrons [15], cargo container screening [2], and screening airport passengers [12, 11].

Previous work done on TSGs [4] focused on the Bayesian zero-sum case and in this paper we extend TSGs to the Bayesian general-sum case. We provide four contributions to accomplish this task: (1) the GATE algorithm which efficiently solves large scale Bayesian general-sum TSGs, (2) the Branch-and-Guide approach which combines branch-and-bound search and MGA in order to efficiently solve nodes in the hierarchical tree, (3) heuristics that speed up the computation of GATE, and (4) experimental evaluation of GATE showing the scalability of our algorithm.

Using the aforementioned contributions this paper presents a practical approach for solving Bayesian general-sum TSGs that scales up to problem sizes encountered in the real world. Thus, with this paper we hope to increase the applicability of TSGs by providing techniques for solving large scale Bayesian general-sum TSGs.

Acknowledgments: This research was supported by the United States Department of Homeland Security through the National Center for Risk and Economic Analysis of Terrorism Events (CREATE) under award number 2010-ST-061-RE0001 and by MURI grant W911NF-11-1-0332

REFERENCES

- [1] AAAE, 'Transportation security policy', Technical report, American Association of Airport Executives, (2014).
- [2] Saket Anand, David Madigan, Richard Mammone, Saumit Pathak, and Fred Roberts, 'Experimental analysis of sequential decision making algorithms for port of entry inspection procedures', in *Intelligence and Security Informatics*, 319–330, Springer, (2006).
- [3] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance, 'Branch-and-price: Column generation for solving huge integer programs', *Operations research*, **46**(3), 316–329, (1998).
- [4] Matthew Brown, Arunesh Sinha, Aaron Schlenker, and Milind Tambe, 'One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats', in *AAAI conference on Artificial Intelligence (AAAI)*, (2016).
- [5] Vincent Conitzer and Tuomas Sandholm, 'Computing the optimal strategy to commit to', in *Proceedings of the 7th ACM conference on Electronic commerce*, pp. 82–90. ACM, (2006).
- [6] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordóñez, and Milind Tambe, 'Security games with arbitrary schedules: A branch and price approach.', in *AAAI*, (2010).
- [7] Manish Jain, Kevin Leyton-Brown, and Milind Tambe, 'The deployment-to-saturation ratio in security games', *Target*, **1**(5), 5, (2012).
- [8] Manish Jain, Milind Tambe, and Christopher Kiekintveld, 'Quality-bounded solutions for finite bayesian stackelberg games: Scaling up', in *International Conference on Autonomous Agents and Multiagent Systems*, (2011).
- [9] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordez, and Milind Tambe, 'Computing optimal randomized resource allocations for massive security games', in *The Eighth International Conference on Autonomous Agents and Multiagent Systems*, (2009).
- [10] Joshua Letchford and Vincent Conitzer, 'Solving security games on graphs via marginal probabilities.', in *AAAI*. Citeseer, (2013).
- [11] Laura A McLay, Adrian J Lee, and Sheldon H Jacobson, 'Risk-based policies for airport security checkpoint screening', *Transportation science*, **44**(3), 333–349, (2010).
- [12] Xiaofeng Nie, Rajan Batta, Colin G Drury, and Li Lin, 'Passenger grouping with risk levels in an airport security system', *European Journal of Operational Research*, **194**(2), 574–584, (2009).
- [13] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus, 'Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games', in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pp. 895–902. International Foundation for Autonomous Agents and Multiagent Systems, (2008).
- [14] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus, 'Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport', in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pp. 125–132. International Foundation for Autonomous Agents and Multiagent Systems, (2008).
- [15] Brian C Ricks, Brian Nakamura, Alper Almaz, Robert DeMarco, Cindy Hui, Paul Kantor, Alisa Matlin, Christie Nelson, Holly Powell, Fred Roberts, et al., 'Modeling the impact of patron screening at an nfl stadium', in *IIE Annual Conference. Proceedings*, p. 3086. Institute of Industrial Engineers-Publisher, (2014).
- [16] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer, 'Protect: A deployed game theoretic system to protect the ports of the united states', in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 13–20. International Foundation for Autonomous Agents and Multiagent Systems, (2012).
- [17] Eric Shieh, Albert Xin Jiang, Amulya Yadav, Pradeep Varakantham, and Milind Tambe, 'Unleashing dec-mdps in security games: Enabling effective defender teamwork', in *European Conference on Artificial Intelligence (ECAI)*, (2014).
- [18] Jason Tsai, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Shyamsunder Rathi, 'Iris-a tool for strategic security allocation in transportation networks', (2009).
- [19] Xiaowen Wang, Cen Song, and Jun Zhuang, 'Simulating a multi-stage screening network: A queueing theory and game theory application', in *Game Theoretic Analysis of Congestion, Safety and Security*, 55–80, Springer, (2015).
- [20] Rong Yang, Albert Xin Jiang, Milind Tambe, and Fernando Ordóñez, 'Scaling-up security games with boundedly rational adversaries: A cutting-plane approach', in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. AAAI Press, (2013).

False-Name-Proof Mechanisms for Path Auctions in Social Networks

Lei Zhang¹ and Haibin Chen^{1,2} and Jun Wu¹ and Chong-Jun Wang¹ and Junyuan Xie¹

Abstract. We study path auction mechanisms for buying path between two given nodes in a social network, where edges are owned by strategic agents. The well known VCG mechanism is the unique solution that guarantees both truthfulness and efficiency. However, in social network environments, the mechanism is vulnerable to false-name manipulations where agents can profit from placing multiple bids under fictitious names. Moreover, the VCG mechanism often leads to high overpayment. In this paper, we present core-selecting path mechanisms that are robust against false-name bids and address the overpayment problem. Specifically, we provide a new formulation for the core, which greatly reduces the number of core constraints. Based on the new formulation, we present a Vickery-nearest pricing rule, which finds the core payment profile that minimizes the L_∞ distance to the VCG payment profile. We prove that the Vickery-nearest core payments can be computed in polynomial time by solving linear programs. Our experiment results on real network datasets and reported cost dataset show that our Vickery-nearest core-selecting path mechanism can reduce VCG’s overpayment by about 20%.

1 Introduction

We consider the problem of buying shortest paths between two given nodes in a social network. For example, in professional networks like LinkedIn, job seekers might want to buy short paths to potential employers; in social media networks, advertisers might want to buy short paths to target customers. In these examples, edges in the networks are owned by strategic agents, and each agent i has a private cost c_i of being included in the path. The goal is to design a path auction mechanism to determine which path to buy, and how much each agent is paid.

The problem is hard because agents may lie about their costs if lying could increase their utilities. Previous work on path auctions have focused on the well known Vickery-Clark-Groves (VCG) Mechanism [24]. The mechanism pays each agent on the shortest path an amount equal to the highest bid with which the agent could have won. It can be shown that the VCG mechanism is the unique efficient and dominant-strategy truthful path mechanism.

However, the VCG mechanism suffers from two economic problems that make it rarely used in practice. The first is that it can lead to significant overpayment. For example, in Figure 1, VCG selects the bottom path and pays 24, while the cost of the shortest path is only 3. Previous work shows that all truthful path mechanisms can be forced to make arbitrarily high overpayment in the worst case [14].

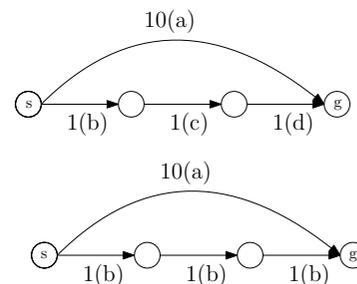


Figure 1: Problems with the VCG path mechanism. Top: VCG pays 8 to agent b, c and d , and the total payment is 24; Bottom: VCG pays 10 to agent b for his three edges, but if agent b places three bids under three different names, he will get $3 * 8 = 24$.

The second economic problem is that the VCG mechanism is vulnerable to false name manipulations [31]. In real world networks, an agent may own multiple edges or even a whole sub-network. Meanwhile, it is also very easy for agents to create fake accounts in social network environments. Then those agents that own many edges can profit from false name manipulations where they place multiple bids under these fictitious names. For example, in Figure 1, VCG pays 10 to agent b for the three edges in the bottom path, however, if agent b submits three separate bids under three different names, he will get a total payment of 24. Therefore, the VCG mechanism is not false-name proof.

In this paper, we are interested in designing efficient path mechanisms to address the economic problems of the VCG mechanism. Based on the framework of core-selecting auctions, we present core-selecting path mechanisms which relax dominant-strategy truthfulness and use the core as the solution concept. We prove that core-selecting path mechanisms are more frugal than the VCG mechanism and are robust against false-name bids. We then give a new core formulation that greatly reduces the number of core constraints, which allows us to compute Vickery-nearest core payments use linear programming techniques. We further show that the Vickery-nearest core payments can be computed in polynomial time. We then evaluate core-selecting path mechanisms on real network data and show that the Vickery-nearest core-selecting path mechanism can greatly reduce VCG’s overpayment under realistic bid (reported cost) distributions.

Our main contributions in this work are: (1) We present a new core formulation that reduces the number of core constraints to 2^k , where k is the network diameter. (2) We give a linear program to compute the Vickery-nearest core payments which minimize the L_∞ distance to the VCG payments.

¹ National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China, email: {zhangl,wujun,chjwang,jyxie}@nju.edu.cn

² Department of Electronic Technology, Engineering University of CAPF, Xian, China, email: hbchenwj@163.com

2 Related Work

The problem of designing mechanisms for path auctions was first studied in [24], where edges in a network are owned by strategic agents, and the cost of the edges is private information of the agent owning the edges. The VCG mechanism is applied to find the shortest paths. The mechanism is shown to be dominant strategy truthful, which means all agents reporting their true costs is a dominant strategy equilibrium. It is also shown that the VCG payments can be computed using n runs of Dijkstras algorithm in $O(nm + n^2 \log n)$ time. It is later shown that if the graph is undirected then the VCG payments can be computed in only $O(m + n \log n)$ time [17].

Although the VCG mechanism for path auction is efficient and truthful, it is found that the VCG mechanism has some undesirable properties. Previous work has found that VCG path mechanism can be forced to make arbitrarily high overpayment in the worst case, in fact the result can be generalized to include all truthful path mechanisms [14, 20]. This led to the study of frugal path mechanisms [2].

Previous work have also studied the VCG overpayment in the Internet inter-domain routing graph [15] and large random graphs [19]. The results show that the VCG overpayment can be intriguingly low when these graphs have unit edge costs. It is also noted the VCG payments can be reduced by removing edges in the graph, but it is NP-hard to determine the optimal set of edges to remove in order to get the lowest VCG payments [13].

False-name manipulations have been studied in a number of anonymous environments, including combinatorial auctions [30, 29, 26, 1], voting [28, 4, 3], matching [25] and social networks [7, 5]. Previous work have also studied first-price path auction mechanism which can be shown to be false-name proof [12, 18].

In addition to the literature on mentioned above, our work is also related to the literature on core-selecting auctions [22, 9, 11, 10, 8]. In particular, we use techniques from [8] to prove that core-selecting path mechanisms are robust against false-name bids.

3 Preliminary

We consider a setting where there is a social network represented by a graph $G = (V, E)$, with $|E| = n$. Each edge in the graph represents a strategic agent e that has a cost $c(e) \in R_{\geq 0}$ of being included in the path, and this cost is the private information of agent e . Given a start node s and a goal node g , the goal is to buy the shortest (least-cost) path from s to g . A solution to this problem is a subset of edges and a payment profile that describes the payments to the agents in the subset.

Since the costs c is private information, we set this up as a mechanism design problem. Each agent e who is a candidate for the path will make a bid $b_e > 0$ and the path mechanism will use an allocation rule $x_e(\mathbf{b}) \in \{0, 1\}$ and a payment rule $t_e(\mathbf{b}) \geq 0$ to determine whether or not agent e is selected, and the payment to agent e , respectively.

We assume agents are rational and strategic, they will choose bids to maximize their own utilities and may lie about their cost if lying can increase their utilities. Let allocation $\mathbf{x} = (x_1, \dots, x_n)$ and payment profile $\mathbf{t} = (t_1, \dots, t_n)$ denote the outcome of a path mechanism. The utility of agent e is defined through the following quasi-linear function,

$$\pi_e = \begin{cases} t_e - c_e & \text{if agent } e \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that agents are individual rational, which means that they are willing to participate in a path mechanism only if they are guaranteed a non-negative utility, so we have $\forall e \in E, t_e \geq b_e$.

Denote the auctioneer by 0, its utility $\pi_0 = -\sum_{e \in E} t_e$. The social welfare of $N = E \cup \{0\}$ is then the cost of the chosen path,

$$w(N) = \pi_0 + \sum_{e \in E} \pi_e = -\sum_{e \in E} x_e c_e \quad (1)$$

A path auction mechanism is efficient if it selects a minimum cost path from s to g . The minimum cost is denoted as $d(s, g, G)$. Then the social welfare of an efficient mechanism is: $w^*(N) = -d(s, g, G)$.

3.1 The VCG Mechanism for Path Auctions

One appealing mechanism for the path auction problem is the VCG mechanism [27, 6, 16]. The VCG mechanism is efficient, it chooses a shortest path so that the social welfare is maximized. Bidders that are not included in the chosen path are paid 0. For bidder e in the chosen path, its utility is given by,

$$\pi_{\text{VCG},e} = w^*(N) - w^*(N - \{e\}) \quad (2)$$

where $w^*(N - e)$ is the social welfare if bidder e 's bid is ignored.

By the definition of π_e , the VCG payment to agent e can be computed as follows,

$$\begin{aligned} t_{\text{VCG},e} &= \pi_{\text{VCG},e} + b_e \\ &= w^*(N) - w^*(N - \{e\}) + b_e \\ &= -d(s, g, G) + d(s, g, G - \{e\}) + b_e \end{aligned} \quad (3)$$

where $G - \{e\}$ stands for the graph G with edge e removed and $d(s, g, G - \{e\})$ is the cost of the shortest path from s to g in $G - \{e\}$. Alternatively, $d(s, g, G - \{e\})$ can also be considered as the cost of shortest path from s to g in the graph G if we set $c_e = \infty$.

The VCG mechanism is dominant strategy truthful. Note that in Equation 3, as bidder e is in the chosen path, its bid b_e also appears in $d(s, g, G)$, so it can be cancelled with the last term. The VCG payment t_e is thus not dependent on bidder e 's reported cost b_e . Therefore, it is a weakly dominant strategy for bidders to report their true costs: $\forall e \in E, b_e = c_e$.

3.2 Problems with the VCG Mechanism

The VCG mechanism is appealing because it is the only auction mechanism that is both truthful and efficient. However, it suffers from two problems that make it rarely used in practice.

The first is its overpayment problem. Consider a graph with two disjoint paths from s to g , the shortest path p_1 with cost 0 and the second shortest path p_2 with cost 1. The VCG mechanism pays 1 to each agent in p_1 . If there is $n - 1$ edges in p_1 , then VCG will pay $n - 1$ for p_1 , which is a $\Theta(n)$ factor more than the cost of the second cheapest path.

In fact, using results from single-parameter mechanism design [23], it can be shown that such a worst-case overpayment is an intrinsic property of any truthful path mechanism [14].

The second problem of the VCG mechanism is that it is vulnerable to false-name bidding or shill bidding, where bidders create fake names and submit multiple bids under these names [31, 12]. This kind of strategic bidding is easy to implement in path auctions on social networks because it is hard to verify all the bidders' identities.

4 Core-Selecting Path Mechanisms

In this section, we propose core-selecting path mechanisms to address the overpayment problem in the VCG mechanism and provide robustness against false-name bids. Since VCG is the unique mechanism to guarantee both allocation efficiency and dominant strategy truthfulness, we have to relax dominant-strategy truthfulness and use alternative solution concepts. The idea is to model path auction as a cooperative game (N, W) and use the core as our solution concept.

The set of agents in the cooperative game is $N = E \cup \{0\}$, which includes all bidders in E and the buyer 0. For a coalition $L \subseteq N$, its coalition value function $W(L)$ is defined as,

$$W(L) = \begin{cases} -d(s, g, L) & \text{if } 0 \in L, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

where $d(s, g, L)$ is the cost of the optimal s - g path in the graph formed by L . Note that if a coalition does not include the buyer, then its coalition value equals 0.

We can now define the concept of the core. An outcome is in the core when the total utility of N equals $W(N)$, and the total utility to every coalition L of agents is at least $W(L)$.

Definition 1 (Core outcome). *A core outcome in a path auction mechanism is an allocation and payment profile such that the utility profile $\pi = (\pi_1, \dots, \pi_n)$ satisfies:*

$$(C0): W(N) = \sum_{i \in N} \pi_i \quad (5)$$

$$(C1): W(L) \leq \sum_{i \in L} \pi_i \quad \forall L \subseteq N \quad (6)$$

The first core constraint (C0) requires that the total utility of N equals $-d(s, g, N)$, which means that the optimal path is always selected. For any coalition L , the second core constraint set (C1) requires that the total utility to L is no less than the value it could obtain, which equals $-d(s, g, L)$.

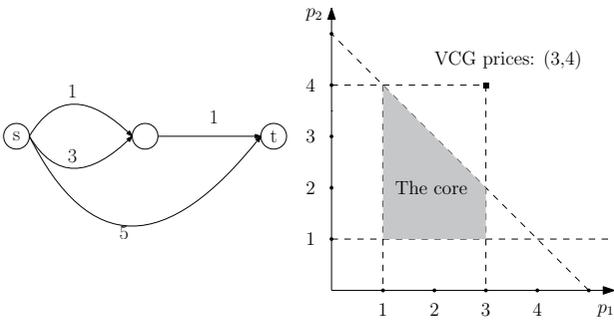


Figure 2: The core of a path auction. Left: There are four bidders with cost 1, 1, 3, 5. Right: Core payments for bidders with cost 1. The VCG payment profile $(3, 4, 0, 0)$ is not in the core.

Given a bid profile \mathbf{b} , let $Core(\mathbf{b})$ be the set of payment profiles in the core when the bidders' cost profile \mathbf{c} equals \mathbf{b} . It can be verified that \mathbf{b} is always in the core: $\mathbf{b} \in Core(\mathbf{b})$, so the set $Core(\mathbf{b})$ is always non-empty. This means that the first-price path auction is core-selecting. We can now define the concept of core-selecting path mechanisms.

Definition 2 (Core-selecting path mechanism). *A path auction mechanism is core-selecting if (1) it selects the optimal path; and (2) the payment profile \mathbf{t} is computed so that $\mathbf{t} \in Core(\mathbf{b})$.*

Example 1. *A path auction is shown in Figure 2. There are four agents, the cost profile $\mathbf{c} = (1, 1, 3, 5)$. The optimal path with cost 2 is selected in both the VCG mechanism and core-selecting mechanisms. The VCG payment profile $\mathbf{t}_{VCG} = (3, 4, 0, 0)$. The core payments $Core(\mathbf{b})$ is the convex hull of the set $\{(1, 1, 0, 0), (1, 4, 0, 0), (3, 1, 0, 0), (3, 2, 0, 0)\}$. We can see that the VCG payment profile $\mathbf{t}_{VCG} \notin Core(\mathbf{b})$, therefore the VCG mechanism is not a core-selecting path mechanism.*

5 Theoretical Results

In this section, we give several theoretical results for core-selecting path mechanisms. The first is that they are always more frugal than the VCG mechanism, specifically, core payments are never higher than VCG payments.

Theorem 1. *The VCG payment to bidder i equals its highest core payment over all payment profiles in the core. That is,*

$$\forall i \in E, \quad \max\{t_i | \mathbf{t} \in Core(\mathbf{b})\} = t_{VCG,i} \quad (7)$$

Proof. First, note that there is a core outcome in which bidder i gets its VCG payment $t_{VCG,i} = W(N) - W(N - i) + b_i$, while other bidders are paid their cost and get a utility of 0. The buyer's utility $\pi_0 = W(N - i)$. We can verify that this utility profile is in the core. Therefore, we have

$$\max\{t_i | \mathbf{t} \in Core\} \geq t_{VCG,i} \quad (8)$$

Meanwhile, suppose that in some core outcome bidder i is paid strictly more than its VCG payment, then its utility $\pi_i > t_{VCG,i} - b_i$. For coalition $N - i$, we have

$$\sum_{k \in N-i} \pi_k = W(N) - \pi_i < W(N - i) + b_i \leq W(N - i) \quad (9)$$

The core constraint corresponding to coalition $N - i$ is violated, so we have

$$\max\{t_i | \mathbf{t} \in Core\} \leq t_{VCG,i} \quad (10)$$

The proof is completed by combining (8) and (10). \square

The second result is that core-selecting path mechanisms are robust against false-name bidding. In fact, we show that they are the only type of efficient path mechanisms that are robust against false-name bidding.

Theorem 2. *An efficient path auction mechanism has the property that no bidder can earn more than its VCG payment by bidding with false names if and only if it is a core-selecting mechanism.*

Proof. Let $L \subseteq N$ be a coalition of bidders, it's possible that these bidders are false-name bidders. The condition requires that these bidders can not get more payment than if they were to submit their merged bid in a VCG mechanism, which means $\sum_{i \in L} t_i \leq t_{VCG,L}$. In the VCG mechanism, the payment for coalition L is,

$$t_{VCG,L} = W(N) - W(N - L) + \sum_{i \in L} b_i \quad (11)$$

The condition is therefore

$$\sum_{i \in L} t_i \leq t_{VCG,L} = W(N) - W(N - L) + \sum_{i \in L} b_i \quad (12)$$

Since the path auction mechanism is efficient, we have $W(N) = \pi_0 + \sum_{i \in N} \pi_i$. The condition can be written as,

$$\sum_{i \in (N-L) \cup \{0\}} \pi_i \geq W(N-L) \quad (13)$$

Since L is an arbitrary coalition of bidders, we have that for any coalition $T = N - L$,

$$\sum_{i \in T \cup \{0\}} \pi_i \geq W(T) \quad (14)$$

Therefore all core constraints in $C1$ are satisfied. Combining this with efficiency, we have $\mathbf{t} \in \text{Core}(\mathbf{b})$. \square

5.1 Core constraint set formulation

The core of a path auction is defined in terms of coalitions. For a network with n edges, the number of core constraints in $C1$ is 2^n . However, many constraints are redundant, for example, if a coalition L doesn't contain $s-g$ path, then $W(L) = -\infty$, the corresponding constraint is redundant and can be removed. We now formulate the core constraint set $C1$ in terms of paths.

$$(C2) : \sum_{e \in p-p^*} c(e) \geq \sum_{e \in p^*-p} t(e) \quad \forall p \in P \quad (15)$$

where P is the set of all paths from s to g , p^* is the optimal path, and $p-p^* = \{e \in p | e \notin p^*\}$, $p^*-p = \{e \in p^* | e \notin p\}$.

The following proposition shows that constraint set $C1$ and $C2$ describe the same set of core payments.

Proposition 1. *The two sets of constraints $C1$ and $C2$ describe the same core.*

Proof. Let $L \subseteq N$ be a coalition of bidders. By $C1$, we have

$$W(L \cup \{0\}) \leq \pi_0 + \sum_{i \in L} \pi_i \quad (16)$$

Assume that the optimal $s-g$ path in the sub-graph formed by L is p . Since all bidders not in p^* are not selected and have utility 0, the constraint can be rewritten as

$$-\sum_{e \in p} c(e) \leq -\sum_{e \in p^*} t(e) + \sum_{e \in p \cap p^*} [t(e) - c(e)] \quad (17)$$

After rearranging the terms, we get

$$\sum_{e \in p^*} t(e) - \sum_{e \in p \cap p^*} t(e) \leq \sum_{e \in p} c(e) - \sum_{e \in p \cap p^*} c(e)$$

which is equivalent to the constraint in $C2$. \square

The size of core constraint set is reduced to the number of $s-g$ paths. However, there exists graphs with exponentially many $s-g$ paths, so there might still be an exponential number of core constraints in $C2$ in the worst case. In fact, it is #P-complete to count the number of $s-g$ paths in a general graph.

Note that p^*-p is a subset of p^* , we can enumerate all p^*-p and create a new formulation for the core constraint set in terms of p^* 's subsets.

$$(C3) : d(s, g, G-x) - d(s, g, G) + \sum_{e \in x} c(e) \geq \sum_{e \in x} t(e) \quad \forall x \subseteq p^* \quad (18)$$

where $d(s, g, G-x)$ is the cost of the optimal $s-g$ path in the graph G with edges in x removed. Theorem 3 shows that $C3$ still describes the same set of core payments as $C1$ and $C2$.

Theorem 3. *The two sets of constraints $C2$ and $C3$ describe the same core.*

Proof. We will show that for each constraint in one set, there is a constraint in the other set that implies it. Therefore, the two sets describe the same core.

$(C3) \implies (C2)$: Every constraint in $(C2)$ corresponds to a path p . For every such path p , there exists a subset x of p^* such that $x = p^* - p$. By $(C3)$, we have

$$\sum_{e \in p^*-p} t(e) \leq d(s, g, G-(p^*-p)) - d(s, g, G) + \sum_{e \in p^*-p} c(e) \quad (19)$$

As $p^* - p = p^* - (p^* \cap p)$, we get

$$-d(s, g, G) + \sum_{e \in p^*-p} c(e) = -\sum_{e \in p^* \cap p} c(e) \quad (20)$$

Since $p \subset (G - (p^* - p))$, p is a valid path in $G - (p^* - p)$, we get

$$d(s, g, G - (p^* - p)) \leq \sum_{e \in p} c(e) \quad (21)$$

Combining (19), (20), and (21), we have

$$\sum_{e \in p^*-p} t(e) \leq \sum_{e \in p} c(e) - \sum_{e \in p^* \cap p} c(e) = \sum_{e \in p-p^*} c(e) \quad (22)$$

which is p 's corresponding constraint in $(C2)$. Therefore every constraint in $(C2)$ also exists in $(C3)$.

$(C2) \implies (C3)$: Every constraint in $(C3)$ corresponds to a subset x of the optimal path p^* . For every such subset x , there exists a path p such that p is the optimal path in the graph $G - x$, so we have $\sum_{e \in p} c(e) = d(s, g, G-x)$.

As $x \subseteq p^*$ and $x \cap p = \emptyset$, we get

$$\sum_{e \in p^*-p} t(e) = \sum_{e \in x} t(e) + \sum_{e \in (p^*-p)-x} t(e) \quad (23)$$

By $C2$, we have

$$\sum_{e \in x} t(e) + \sum_{e \in (p^*-p)-x} t(e) \leq \sum_{e \in p} c(e) - \sum_{e \in p \cap p^*} c(e) \quad (24)$$

For each $e \in p^*$, we have $c(e) \leq t(e)$,

$$\sum_{e \in x} t(e) \leq \sum_{e \in p} c(e) - \sum_{e \in p \cap p^*} c(e) - \sum_{e \in (p^*-p)-x} c(e) \quad (25)$$

$$= \sum_{e \in p} c(e) - \sum_{e \in p^*} c(e) + \sum_{e \in p^*-p} c(e) - \sum_{e \in (p^*-p)-x} c(e) \quad (26)$$

$$= d(s, g, G-x) - d(s, g, G) + \sum_{e \in x} c(e) \quad (27)$$

Therefore, the constraint corresponding to p in $C2$ implies the constraint corresponding to x in $C3$. \square

The theorem indicates that for a shortest path with k edges, the number of core constraints in $(C3)$ is 2^k . Real social networks often have small diameter (longest shortest path length k), which allows us to compute core payments efficiently using linear programming techniques.

6 Pricing Algorithms

As core-selecting path mechanisms are not dominant-strategy truthful, it is important to provide incentives for bidders to bid truthfully. In this section, we present Vickrey-nearest pricing rule for core-selecting path mechanisms, which is shown to have good incentive properties. The idea is to find core payments that maximize the total payment and are as close to the VCG payments as possible.

6.1 Maximum Payment

We employ a two-step optimization algorithm to determine core payments. First, we maximize the total payment over the core. Recall the core constraint set $C3$, for each subset x of the optimal path p^* , we have

$$\sum_{e \in x} t(e) \leq d(s, g, G - x) - d(s, g, G) + \sum_{e \in x} c(e) \quad (28)$$

Let $\beta_x = d(s, g, G - x) - d(s, g, G) + \sum_{e \in x} c(e)$, and denote the vector of all β_x values as β , we have

$$At \leq \beta \quad (29)$$

where A is a $2^k \times k$ matrix, k is the number of edges in p^* . A_{ij} equals 1 if bidder j is in the i -th subset and equals 0 otherwise. The maximum total payment α can then be found using the following linear program (LP-0):

$$\begin{aligned} \text{(LP-0): } \alpha &= \max t \cdot \mathbf{1} \\ \text{subject to: } At &\leq \beta \\ t &\geq c \end{aligned} \quad (30)$$

The linear program LP-0 has $2^k + k$ constraints, which is exponential in the number of variables in LP-0, however, we can still prove that LP-0 can be solved in polynomial time.

Proposition 2. *The maximum core payment α can be computed in time polynomial in k by solving the linear program LP-0.*

Proof. We first show that there is a polynomial time separation oracle, which given a core payment profile t , answers that t satisfies $At \leq \beta$ and $t \geq c$, or returns an inequality that is not satisfied by t .

The second set of constraints $t \geq c$ is easy to verify. To check that whether t satisfied the first set of constraints $At \leq \beta$, we set the cost of edges in the original optimal path p^* to be t and then compute a new optimal path p' . If the optimal path stays the same, $p^* = p'$, then the first set of constraints is satisfied; otherwise assume that $pp = p^* - p'$, then the following constraint corresponding to pp is not satisfied: $\sum_{e \in pp} t(e) \leq \beta_{pp}$.

Since the separation oracle runs in polynomial time, the ellipsoid method can give solutions to LP-0 in time polynomial in k . \square

6.2 VCG-Nearest Payments

In the second step, we find core payments to minimize the L_∞ distance to the VCG payments. The L_∞ between two payment vectors x and y is defined as

$$L_\infty(x, y) = \max\{|x_1 - y_1|, |x_2 - y_2|, \dots, |x_k - y_k|\}$$

Besides the core constraints in (LP-0), we add one more constraint that the sum of payments should equal the maximum total payment:

$t \cdot \mathbf{1} = \alpha$. Then we can find the Vickrey-nearest payments by solving the following problem:

$$\begin{aligned} \text{(NLP-1): } r &= \min \|t - t_{\text{VCG}}\|_\infty \\ \text{subject to: } At &\leq \beta \\ t &\geq c \\ t \cdot \mathbf{1} &= \alpha \end{aligned} \quad (31)$$

It is hard to optimize (31) directly. We reformulate it as a linear program by adding a new variable y . Meanwhile, for each edge i in the optimal path, we add a new constraint: $|t_{\text{VCG},i} - t_i| \leq y$. y can be understood as the the maximum difference between t_i^{VCG} and t_i . As $t_i \leq t_{\text{VCG},i}$, the new constraints can be rewritten as: $t + y\mathbf{1} \geq t_{\text{VCG}}$. Then we can find the Vickrey-nearest payments t by solving the following linear program:

$$\begin{aligned} \text{(LP-2): } r &= \min y \\ \text{subject to: } At &\leq \beta \\ t &\geq c \\ t \cdot \mathbf{1} &= \alpha \\ t + y\mathbf{1} &\geq t_{\text{VCG}} \end{aligned} \quad (32)$$

For an optimal path with k edges, the linear program (LP-2) has k decision variables and $2^k + 2k + 1$ constraints.

Theorem 4. *The Vickrey-nearest core payments can be computed in polynomial time by solving linear program LP-2.*

Proof. Given a core payment profile t and y , we already give a polynomial time separation oracle for $At \leq \beta$ and $t \geq c$ in Proposition 2. The third set of constraints $t \cdot \mathbf{1} = \alpha$ is trivial to check. The last set of constraints can also be sequentially verified in $O(k)$ time. Therefore, we can also solve LP-2 in polynomial time using the ellipsoid method. \square

For the path auction in Figure 2, the maximum total payment α is 5, and the Vickrey-nearest core payments is $(2, 3, 0, 0)$.

To analyze the incentive property of the Vickrey-nearest pricing rule, we assume bidders know each others' costs and analyze the Nash equilibrium. Denote \hat{b}_{-i} as the bids of bidders other than i . Given any \hat{b}_{-i} , let $\hat{t}_{\text{VCG},i}$ be the VCG payment to bidder i when i is truthful. Then we have the following result,

Proposition 3. *$\hat{t}_{\text{VCG},i}$ is a best response by bidder i to the bids \hat{b}_{-i} of others.*

Proposition 3 holds because if i 's bid is more than $\hat{t}_{\text{VCG},i}$, then the core constraint corresponding to coalition $N - \{i\}$ is violated. Define the regret of a bidder as the difference between his utility submitting a best-response bid to the bids of others and his utility when bidding truthfully. We can prove the following result,

Theorem 5. *The Vickrey-nearest core-selecting path mechanism minimizes the maximum regret for bidders across all core-selecting path mechanisms.*

Proof. Assume that all bidders are truthful. Fixing bids b_{-i} of others, the best response of bidder i is to bid his VCG payment $t_{\text{VCG},i}$. Then bidder i 's regret is $t_{\text{VCG},i} - t_i$, where t_i is the VCG-nearest core payment for bidder i . As the Vickrey-nearest payments t minimizes the L_∞ distance to the VCG payments t_{VCG} , the maximum regret for all bidders is minimized. \square

7 Experiment Results

In this section, we evaluate the performance of false-name proof path mechanisms.

7.1 Datasets and Problem Generation

First we describe the network and cost(bidding) datasets used in the following experiments. The network datasets we used are from the SNAP datasets [21], which consist of four social networks:

- Facebook network. The dataset consists of friends lists from Facebook. The data was collected from survey participants using Facebook app³.
- Google+ network. This dataset consists of circles (friends lists) from Google. The data was collected from users who had manually shared their circles.
- Twitter network. This dataset consists of friend list from Twitter. The data was crawled from public sources.
- Wikipedia voting network. The network contains voting data for Wikipedia administrators elections. Nodes in the network represent wikipedia users and a directed edge from node i to node j represents that user i voted on user j .

The detailed network statistics are given in table 1.

Networks	Nodes	Edges	d_{\max}	90-percentile d_{\max}
Facebook	4,039	88,234	8	4.7
Google+	107,614	13,673,453	6	3.0
Twitter	81,306	1,768,149	7	4.5
Wiki-Vote	7,115	103,689	7	3.8

Table 1: Network statistics, d_{\max} is the network diameter, or the maximum shortest path length, the last column is the 90-th percentile of shortest path length distribution.

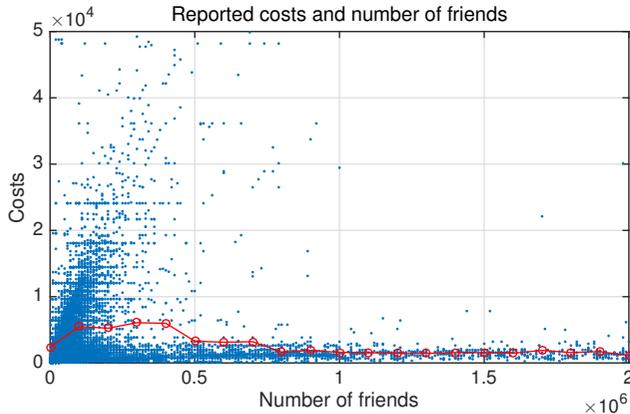


Figure 3: Reported costs plotted against number of friends. The red line shows average reported costs.

Compared with network data, the true costs of bidders are difficult to obtain. In our experiments, we use reported cost data from a microblog advertising platform weibo⁴, where microbloggers are asked to report their costs to make recommendations to friends in their social network. Figure 3 shows reported costs and the number

of friends for 15082 microbloggers. As the correlation between reported costs and the number of friends is rather small (-0.005), we use a simulation approach to generate realistic bids. Each time we take a cost at random from the microblogger bidding data and assign it to an edge from our networks. We also use another unit cost distribution where all costs are set to 1.

The compared path mechanisms include the VCG mechanism (VCG) and the Vickrey-nearest core-selecting mechanism (VNC). For each network and each path mechanism, we generate 3000 problem instances where the start node s and goal node g are selected uniformly at random from all nodes. All problem instances are executed using CPLEX 12.6 on a 3.1 GHz Intel Core i5 processor.

Networks	Avg. shortest path cost	Avg. VCG payments	Avg. VNC payments	Avg. maximum regret
Facebook	3.63	5.66	5.03	0.26
Wikipedia	2.99	3.78	3.43	0.18
Google+	3.19	3.79	3.52	0.08
Twitter	4.81	6.07	5.46	0.18

Table 2: Average payment and average maximum regret under unit cost distribution.

Networks	Avg. shortest path cost	Avg. VCG payments	Avg. VNC payments	Avg. maximum regret
Facebook	871.07	2113.19	1697.70	81.50
Wikipedia	1038.11	2982.92	2599.35	106.56
Google+	673.71	1607.24	1479.95	35.16
Twitter	1380.82	3584.91	2982.22	162.17

Table 3: Average payment and average maximum regret under reported cost distribution.

7.2 Payment Performance

We first study the payment performance of path mechanisms. In Table 2 and 3, we show average payments for the unit cost distribution and the reported cost distribution respectively. We can see that VNC payments are always less than VCG payments. Under the unit cost distribution, average overpayment is not very high for both mechanisms. However, under the reported cost distribution, VCG overpayments become significant in all four networks. In particular, the VCG mechanism overpays by 2304 in the twitter network when the shortest paths cost only 1380.

The performance measure we used is the overpayment factors. The overpayment factor of a path mechanism M is defined as the ratio between its total payment and the true cost of the shortest path p^* .

$$OF = \frac{\sum_{e \in p^*} t_M(e)}{\text{cost}(p^*)} \quad (33)$$

We give detailed average overpayment factor results in Figure 4, where average overpayment factors are plotted against the number of edges in p^* . We can see that under the reported distribution, the VCG mechanism overpays by a factor of 2.5 in the Facebook, Google+ and Twitter networks, and overpays by a factor of 3 in the Wikipedia voting network. Meanwhile, the VNC mechanism only overpays by a factor of 2 in these networks. Meanwhile, under unit cost distribution, there is little difference between the overpayment factors for VCG and VNC mechanisms, this is because the overpayments are already very low. Note that when the shortest path contains only one edge ($k = 1$), the VNC payment is equal to the VCG payment, so

³ <https://www.facebook.com/apps/application.php?id=201704403232744>

⁴ <http://www.weibo.com>

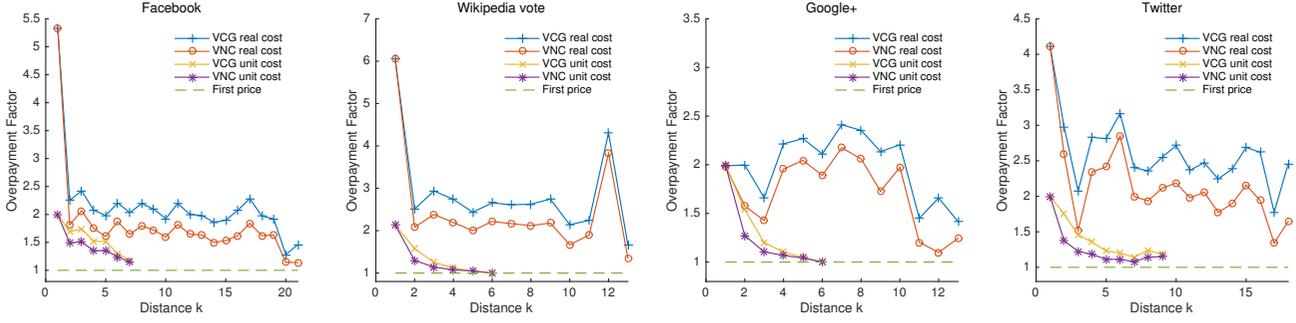


Figure 4: Comparison of average overpayment factors under different cost distributions.

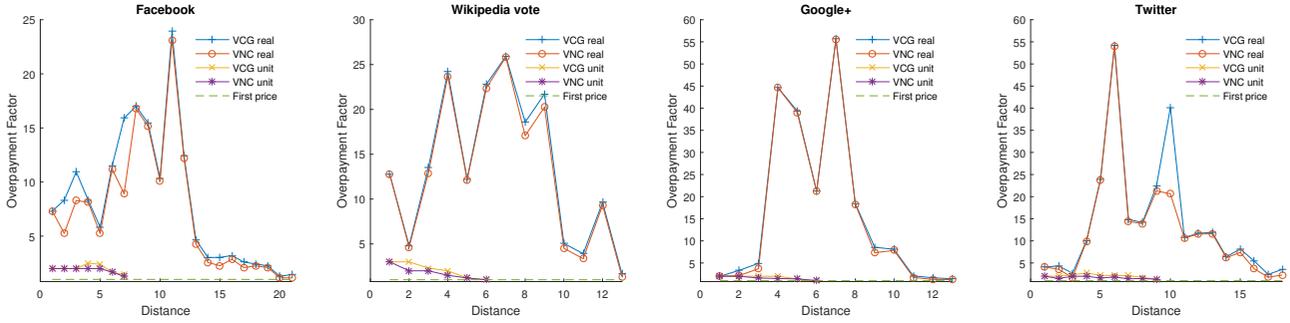


Figure 5: Comparison of worst case overpayment factors under different cost distributions.

the overpayment factors are the same, this is confirmed by Figure 4. We also plot worst case overpayment factor results in Figure 5. We can see that under reported cost distributions, VNC’s overpayment factors and VCG’s overpayment factors are very close in the worst case.

We next evaluate bidders’ maximum regrets under the two cost distributions. Define the maximum regret ratio as the ratio between the maximum regret and the shortest path cost. Under the unit cost distribution, the maximum regret ratio is less than 3% in the Google+ network and around 6% in the others. Under the reported cost distribution, the maximum regret ratio is around 5% in the Google+ network and around 10% in the others.

Networks	first-price	VCG	maxpay	VNC
Facebook	0.004	0.050	0.102	0.104
Google+	0.156	5.382	6.921	6.922
Twitter	0.026	0.790	1.309	1.310
Wiki-Vote	0.001	0.042	0.055	0.056

Table 4: Average time performance (in seconds) under unit cost for first-price mechanism (first-price), the VCG mechanism (VCG), maximum payments (maxpay) and Vickrey-nearest payments (VNC).

7.3 Time Performance

In the next, we evaluate the time performance of path mechanisms. Table 4 shows the average time performance of four different path mechanisms under unit cost. We can see that the time performance of the maximum core payment and the Vickrey-nearest payment is very close. Moreover, in the Facebook network and the Twitter network, core-selecting path mechanisms have comparable time performance with the first-price mechanism and the VCG mechanism.

In the Google+ network and the Wikipedia voting network, core-selecting path mechanisms use about twice the time compared with the VCG mechanism, however, note that all mechanisms use less than 1 seconds.

Networks	first-price	VCG	maxpay	VNC
Facebook	0.005	0.073	12.892	27.530
Google+	0.220	6.220	29.420	29.431
Twitter	0.040	1.001	151.043	175.300
Wiki-Vote	0.001	0.054	0.330	0.382

Table 5: Average time performance (in seconds) under reported cost for first-price mechanism (first-price), the VCG mechanism (VCG), maximum payments (maxpay) and Vickrey-nearest payments (VNC).

Table 5 shows the average time performance of four different path mechanisms under the reported cost distribution. We can see from the table that the first-price path mechanism and the VCG path mechanism only need less than 1 seconds in all networks except the Google+ network, where it take 6 seconds on average to compute the payments. Meanwhile, the maximum core payments and VCG-nearest core payments need much more time to compute, in the Wikipedia voting network the time performance is comparable with that of the first-price mechanism and the VCG mechanism. In the Twitter network, the core-selecting mechanisms have a very bad time performance, where it takes more than 2 minutes on average to compute the core payments. We believe the reason is that some agent reported a very high cost, which make the linear program LP-0 very hard to solve by CPLEX. This indicates one possible future research direction that we may need to find more efficient algorithms for computing the maximum core payments when very high cost (bids) can be reported.

8 Conclusions

In this paper, we propose false-name-proof path mechanisms to address the overpayment problem of the VCG path mechanism and provide robustness against false-name manipulations. Based on a novel core constraint formulation, we give a polynomial time core-selecting path mechanism which finds the core payment profile that minimizes the L_∞ distance to the VCG payment profile. We show that the Vickrey-nearest core-selecting path mechanism has good incentive properties. Experiment results on real network and reported cost data show that the overpayment of the VCG mechanism can be very high, while the Vickrey-nearest core-selecting path mechanism can reduce VCG's overpayment by about 20%. Therefore, Vickrey-nearest core-selecting path mechanism can be considered as an appealing candidate mechanism for path auctions in social network environments where false-name manipulations are too common to ignore.

In this paper, we assume that all possible subsets of agents can form coalitions, however, in real world applications it is reasonable to assume that only connected agents can form coalitions. It is an interesting direction to devise efficient payment algorithms for this kind of environment.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their helpful suggestions. This paper is supported by the National Natural Science Foundation of China (Grant No. 61375069,61403156, 61502227) and this research is partially supported by the Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing University.

REFERENCES

- [1] Colleen Alkalay-Houlihan and Adrian Vetta, 'False-name bidding and economic efficiency in combinatorial auctions', in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pp. 538–544. AAAI Press, (2014).
- [2] Aaron Archer and Éva Tardos, 'Frugal path mechanisms', *ACM Transactions on Algorithms (TALG)*, 3(1), 3, (2007).
- [3] Haris Aziz and Mike Paterson, 'False name manipulations in weighted voting games: Splitting, merging and annexation', in *Proceedings of The 8th International Conference on Autonomous Agents and Multi-agent Systems - Volume 1*, AAMAS '09, pp. 409–416, Richland, SC, (2009). International Foundation for Autonomous Agents and Multiagent Systems.
- [4] Yoram Bachrach and Edith Elkind, 'Divide and conquer: False-name manipulations in weighted voting games', in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '08, pp. 975–982, Richland, SC, (2008). International Foundation for Autonomous Agents and Multiagent Systems.
- [5] Markus Brill, Vincent Conitzer, Rupert Freeman, and Nisarg Shah, 'False-name-proof recommendations in social networks', in *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '16, pp. 332–340, Richland, SC, (2016). International Foundation for Autonomous Agents and Multiagent Systems.
- [6] Edward H Clarke, 'Multipart pricing of public goods', *Public choice*, 11(1), 17–33, (1971).
- [7] Vincent Conitzer, Nicole Immorlica, Joshua Letchford, Kamesh Munagala, and Liad Wagman, *False-Name-Proofness in Social Networks*, 209–221, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [8] B Day and Paul Milgrom, 'Optimal incentives in core-selecting auctions', *Handbook of Market Design*, (2010).
- [9] Robert Day and Paul Milgrom, 'Core-selecting auctions', *International Journal of Game Theory*, July, (2007).
- [10] Robert Day and Paul Milgrom, 'Core-selecting package auctions', *International Journal of Game Theory*, 36(3-4), 393–407, (2008).
- [11] Robert W Day and Subramanian Raghavan, 'Fair payments for efficient allocations in public sector combinatorial auctions', *Management Science*, 53(9), 1389–1406, (2007).
- [12] Ye Du, Rahul Sami, and Yaoyun Shi, 'Path auctions with multiple edge ownership', *Theoretical Computer Science*, 411(1), 293 – 300, (2010).
- [13] Edith Elkind, 'True costs of cheap labor are hard to measure: Edge deletion and vcg payments in graphs', in *Proceedings of the 6th ACM Conference on Electronic Commerce*, EC '05, pp. 108–116, New York, NY, USA, (2005). ACM.
- [14] Edith Elkind, Amit Sahai, and Ken Steiglitz, 'Frugality in path auctions', in *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 701–709. Society for Industrial and Applied Mathematics, (2004).
- [15] Joan Feigenbaum, Christos Papadimitriou, Rahul Sami, and Scott Shenker, 'A bgp-based mechanism for lowest-cost routing', *Distributed Computing*, 18(1), 61–72, (2005).
- [16] Theodore Groves, 'Incentives in teams', *Econometrica: Journal of the Econometric Society*, 617–631, (1973).
- [17] John Hershberger and Subhash Suri, 'Vickrey prices and shortest paths: What is an edge worth?', in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pp. 252–259. IEEE, (2001).
- [18] Nicole Immorlica, David Karger, Evdokia Nikolova, and Rahul Sami, 'First-price path auctions', in *Proceedings of the 6th ACM Conference on Electronic Commerce*, EC '05, pp. 203–212, New York, NY, USA, (2005). ACM.
- [19] D.R. Karger and E. Nikolova, 'On the expected vcg overpayment in large networks', in *Decision and Control, 2006 45th IEEE Conference on*, pp. 2831–2836, (Dec 2006).
- [20] Anna R. Karlin, David Kempe, and Tami Tamir, 'Beyond vcg: Frugality of truthful mechanisms', in *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pp. 615–626, Washington, DC, USA, (2005). IEEE Computer Society.
- [21] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.
- [22] Paul Robert Milgrom, *Putting auction theory to work*, Cambridge University Press, 2004.
- [23] Roger B Myerson, 'Optimal auction design', *Mathematics of operations research*, 6(1), 58–73, (1981).
- [24] Noam Nisan and Amir Ronen, 'Algorithmic mechanism design', in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 129–140. ACM, (1999).
- [25] Taiki Todo and Vincent Conitzer, 'False-name-proof matching', in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pp. 311–318, Richland, SC, (2013). International Foundation for Autonomous Agents and Multiagent Systems.
- [26] Taiki Todo, Atsushi Iwasaki, Makoto Yokoo, and Yuko Sakurai, 'Characterizing false-name-proof allocation rules in combinatorial auctions', in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pp. 265–272, Richland, SC, (2009). International Foundation for Autonomous Agents and Multiagent Systems.
- [27] William Vickrey, 'Counterspeculation, auctions, and competitive sealed tenders', *The Journal of finance*, 16(1), 8–37, (1961).
- [28] Liad Wagman and Vincent Conitzer, 'Optimal false-name-proof voting rules with costly voting', in *Twenty-Third AAAI Conference on Artificial Intelligence*, AAAI 2008, pp. 190–195, (2008).
- [29] Makoto Yokoo, 'Characterization of strategy/false-name proof combinatorial auction protocols: Price-oriented, rationing-free protocol', in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pp. 733–739, San Francisco, CA, USA, (2003). Morgan Kaufmann Publishers Inc.
- [30] Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara, 'Robust combinatorial auction protocol against false-name bids', *Artificial Intelligence*, 130(2), 167 – 181, (2001).
- [31] Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara, 'The effect of false-name bids in combinatorial auctions: New fraud in internet auctions', *Games and Economic Behavior*, 46(1), 174–188, (2004).

Multi-Robot Adversarial Coverage

Roi Yehoshua and Noa Agmon¹

Abstract. This work discusses the problem of adversarial coverage, in which one or more robots are required to visit every point of a given area, which contains threats that might stop the robots. The objective of the robots is to cover the target area as quickly as possible, while maximizing the percentage of covered area before they are stopped. This problem has many real-world applications, from performing coverage missions in hazardous fields such as nuclear power plants, to surveillance of enemy forces in the battlefield and field demining. Previous studies of the problem dealt with single-robot coverage. Using a multi-robot team for the coverage has clear advantages in terms of both coverage time and robustness: even if one robot is totally damaged, others may take over its coverage subtask. Hence, in this paper we describe a multi-robot coverage algorithm for adversarial environments that tries to maximize the percentage of covered area before the team is stopped, while minimizing the coverage time. We analytically show that the algorithm is robust, in that as long as a single robot is able to move, the coverage will be completed. We also establish theoretical bounds on the minimum covered area guaranteed by the algorithm and on the coverage time. Lastly, we evaluate the effectiveness of the algorithm in an extensive set of environments and settings.

1 Introduction

Coverage path planning is one of the fundamental problems in robotics. The goal of coverage path planning is to find a sequence of world locations which allows the robot(s) to visit every part of the target area while optimizing some criteria, usually minimizing travel cost while avoiding obstacles. This problem has many real-world applications, from automatic floor cleaning [2] and coating in supermarkets [4], to field demining [14] and surveillance by unmanned aerial vehicles (UAVs) [7].

In a recently introduced version of the problem, *adversarial coverage* (e.g., [20]), the robot has to cover the given terrain without being stopped by an adversary. Each point in the area is associated with a probability of the robot being stopped at that point. The objective of the robot is to cover the entire target area (including the threat points) as quickly as possible while minimizing the probability that it will be stopped before completing the coverage. This problem is a generalization of the original problem of coverage in neutral environments (without adversarial presence), where risks do not exist, thus are not accounted for, and the only goal is to minimize coverage time [13], [5], [8].

Previous work of adversarial coverage dealt with the single-robot version of the problem. There are obvious advantages in using multiple robots in the adversarial coverage task. Using multiple robots

clearly decreases the time to complete the task due to workload division, as seen in the original multi-robot coverage problem [8], [1]. Additionally, using multiple robots improves robustness, as failure of some members of the robot team can be compensated by others. Therefore, in this paper we extend adversarial coverage to multi-robot systems. We focus on coverage using a map of the work-area (known as offline coverage [6]).

First, we formally define the multi-robot version of adversarial coverage, and the problem of finding the safest coverage path for a multi-robot team. In this paper we are mainly concerned about the survivability of the team and not the coverage time. Nevertheless, the algorithm we propose also tries to minimize the coverage time, as long as the robots' safety is not compromised. Clearly, there is a tradeoff between the two objectives of survivability and coverage time: trying to minimize the risk to the robots along their coverage paths typically means making some redundant steps, which in turn can make their coverage paths longer, and thus increase the risks involved, as well as the coverage time.

Second, we describe an efficient distributed multi-robot adversarial coverage algorithm, that tries to maximize the survivability of the team while optimizing the coverage time. The algorithm is based on decomposition of the target area into connected areas of safe and dangerous locations, and prioritization of the coverage such that coverage of safer areas comes before coverage of more dangerous ones. Our method also utilizes graph partitioning techniques in cases where more than one robot is assigned to the coverage of a given area, in order to speed up its coverage. We provide a theoretical bound on the expected coverage percentage guaranteed by the algorithm, and also analyze the best-case and worst-case completion times for the algorithm.

Finally, we evaluate our method in an extensive set of experiments. The results show that adding more robots to the coverage task can significantly increase the percentage of the area covered as well as reduce the coverage time.

2 Related Work

The problem of single and multi-robot coverage has been extensively discussed in the robotic literature (see Galceran and Carreras [6] for a recent survey). Most approaches to multi-robot coverage extend single-robot ideas to multiple robots by using a strategy to divide the workload. Hazon and Kaminka [8] generalized the STC method to multi-robot teams in the family of Multi Robot Spanning Tree Coverage (MSTC) algorithms. Their solution, along with decreasing the total coverage time, achieved robustness in the sense that as long as one robot works properly, the coverage of the terrain is guaranteed. They have also shown that in multi-robot teams redundancy might be necessary for more efficiency. Agmon et al. [1] proposed a spanning

¹ Bar Ilan University, Israel, email: yehoshr1@cs.biu.ac.il, agmon@cs.biu.ac.il

tree construction algorithm that provides efficient paths in terms of distance, and can be used as a basis for MSTC.

Rekleitis et al. [16] presented a collection of algorithms for the coverage planning problem using a team of mobile robots on an unknown environment, based on an exact cellular decomposition. To achieve coverage in line-of-sight-only communications, the robots take two roles: some members, called explorers, cover the boundaries of the current target cell, while the other members, called coverers, perform simple back-and-forth motions to cover the remainder of the cell. For task/cell allocation among the robots, a greedy auction mechanism is used.

Other approaches to multi-robot coverage found in the literature include methods inspired by biological behaviors found in nature. For example, Luo and Yang [12] presented a biologically inspired neural network approach for coverage tasks to multi-robot scenarios where the robots see each other as moving obstacles. Wagner and Bruckstein [17] explored the problem of room cleaning by a group of robots, and proposed a robust algorithm for complete coverage of a terrain. In their case, the robots have limited capabilities and communicate with each other mainly using pheromones.

The offline single-robot adversarial coverage problem was formally defined in [21], in which we proposed a simple heuristic algorithm for generating a coverage path aiming at minimizing a cost composed of both the survivability of the robot and the coverage path length. The heuristic algorithm worked only for obstacle-free areas, and without any guarantees. In a follow-up paper [22] we have addressed a more specific version of the problem, namely, finding the safest coverage path. There we suggested two heuristic algorithms: STAC, a spanning-tree based coverage algorithm, and GAC, which follows a greedy approach. We have shown that while STAC tends to achieve higher expected coverage, GAC produces shorter coverage paths with lower accumulated risk. In [18] we have built a more sophisticated model of the adversary, in which it can choose the best locations of the threat points, such that the probability of stopping the covering robot is maximized. Lastly, the online single-robot version of the problem was presented in [19].

In the related patrol problem ([15], [3]), a multi-robot team needs to patrol around a closed area with the existence of an adversary attempting to penetrate into the area. The patrol problem resembles the coverage problem in the sense that both require the robot or group of robots to visit all points in the given terrain. However, while coverage seeks to minimize the number of visits to each point (ideally, visiting it only once), patrolling seeks to maximize it (while still visiting all points).

3 Problem Formulation

A team of k robots $\mathcal{R} = \{R_1, \dots, R_k\}$ needs to cover a given area. The area contains threats that may stop the robots, as well as obstacles. In contrast to obstacles which the robots cannot go through, threat locations are places that the robots must visit, but might be stopped at. The robots are given a map of the environment in advance. We assume that communication between the robots is available without any restrictions. Each robot autonomously covers the area it is assigned, keeping track of all the covered and uncovered space by communicating with the other robots (see section 4.2 for the communication requirements).

Furthermore, we assume that the given area can be decomposed into a regular grid with n cells. Let us denote this grid by G . G contains two types of cells: free cells and cells that are occupied by obstacles. Some of the free cells contain threats. Each free cell

i is associated with a threat probability p_i , which measures the likelihood that a threat in that cell will stop a robot visiting it. We define *safe* cells as cells in which the threat probability is $p_i = 0$, while *dangerous* cells are those in which $p_i > 0$. The robots can move in the four basic directions (North, South, East, West), and can locate themselves within the work-area to a specific cell.

Figure 1 shows an example world map of size 20×20 , with 8 robots located at the upper-left corner of the area. Obstacles are represented by black cells, safe cells are colored white, and dangerous cells are represented by 5 different shades of magenta. Darker shades represent higher values of p_i (more dangerous areas).

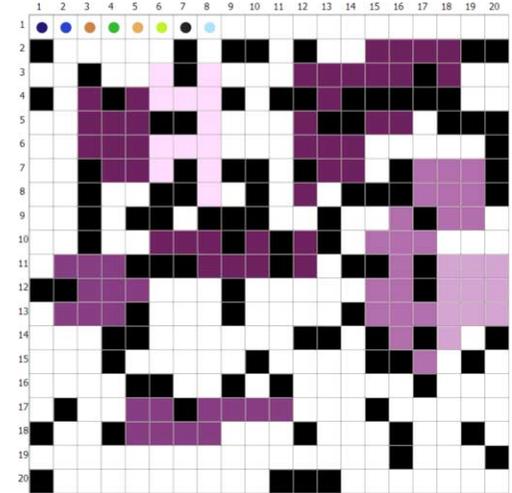


Figure 1. A sample map with 8 robots located at the upper-left corner of the environment. Darker shades represent more dangerous areas.

We assume that robots that have been stopped by threats do not block live robots, i.e. other robots can move through their cells, and that the threats remain there. An example for a real-world scenario in which this assumption holds is when the covering robots are UAVs, and the threats are constant in time (e.g., an anti-drone weapon aimed at a particular location).

The survivability measure can be defined in two levels: the survivability of each single robot, and the survivability of the team. We say that a robot survives the coverage, if it manages to finish its coverage task unharmed. To formally define the survivability of a robot R_i , we first denote the coverage path that it follows by $P_i = (c_1^i, c_2^i, \dots, c_{n_i}^i)$, where c_j^i is the cell that robot i is located at in time step j . Thus, the probability that a robot R_i survives the coverage is:

$$Surv(R_i) = \prod_{j=1}^{n_i} (1 - p_j^i) \quad (1)$$

where p_j^i denotes the probability that the robot is stopped by a threat in cell c_j^i .

We say that a team of robots survives the coverage, if at least one of the robots in the team survives until all cells in the area are visited (coverage completed). Thus, if it takes t time steps to cover the area, the probability that a team of k robots $\mathcal{R} = \{R_1, \dots, R_k\}$ is able to cover this area is:

$$Surv(\mathcal{R}) = \prod_{j=1}^t \left[1 - \prod_{i=1}^k p_j^i \right] \quad (2)$$

Following these definitions, we can now define the Multi-Robot Safe Adversarial Coverage Problem (MRSACP) as follows:

Definition 1 *Multi-Robot Safe Adversarial Coverage Problem (MRSACP):* Given a team of k robots and a grid representation of a world that contains obstacles and threat points, find a coverage path of the grid that maximizes the survivability of the team.

The \mathcal{NP} -hardness of MRSACP follows directly from the \mathcal{NP} -hardness of the single-robot safest coverage path problem [22].

4 Multi-Robot Adversarial Coverage Algorithm

The Multi-Robot Adversarial Coverage algorithm (MRAC, described in Algorithm 1) uses a layered-based approach. It first tries to cover all the safe cells in the target area as efficiently as possible, using the given k robots. Then it covers the dangerous areas from the least dangerous ones to the most dangerous ones. This way the algorithm tries to maximize the coverage percentage before the entire group of robots is stopped.

Before describing the algorithm in detail, let us introduce the following definitions:

Definition 2 A *connected area* is a connected subset of cells in the grid that belong to the same threat level.

We also use the terms *safe areas* and *dangerous areas* to refer to connected areas that are composed of only safe cells or only dangerous cells. Figure 2 shows an example of a grid containing two safe areas and five dangerous areas that belong to two different threat levels. The two low-threat-level areas are outlined by yellow lines and the three high-threat-level areas are outlined by blue lines.

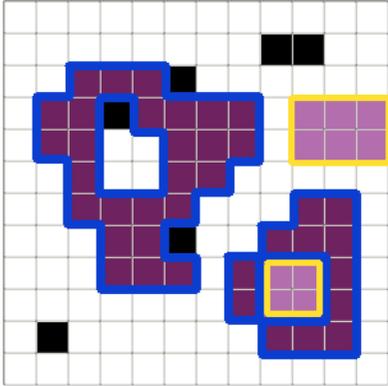


Figure 2. An example of a grid containing two safe areas and five dangerous areas that belong to two different threat levels.

In addition, we denote the distinct threat probabilities that exist in the grid by p_0, \dots, p_l ($p_0 = 0$) (l is finite, since the number of cells in the grid is finite), and define *threat level* i as the group of all the cells that contain threats with probability p_i of stopping the robot.

The main steps of the multi-robot coverage algorithm are:

1. Split the target area into layers according to the threat levels, i.e. layer i contains all the cells that belong to threat level i .
2. For each threat level i , denote by $A_1^i, A_2^i, \dots, A_{n_i}^i$ the connected areas that belong to level i .

3. Each robot is assigned to a safe area $A \in \{A_1^0, \dots, A_{n_0}^0\}$, which has the safest path from its current location. Areas with more than one robot assigned to them are split between their assigned robots.
4. For the coverage of each area, we use a modified version of GAC (Greedy Adversarial Coverage), the state-of-the-art solution to the single-robot adversarial coverage problem [20].
5. When a robot R_i completes its current coverage task, it is allocated an uncovered area from the safest threat level which has not been completed yet. If more than one such area exists, the area with the minimum-risk path from the robot's current location is chosen. If all areas have been allocated, then the robot is added to an area that is already being covered by another robot. This area is split into two, where each robot is assigned to the sub-area with the safest path from its current location.
6. If a robot is hit by a threat during the coverage of its allocated area, then this area is returned to the pool of unassigned areas.
7. If all the robots have completed their area coverage, and there are no uncovered cells, then the robots declare the environment covered.

The algorithm consists of two main phases: the first phase (steps 1–3) is executed prior to the coverage and computes the initial allocations of areas to the robots. This computation can be performed either offline or online by one of the robots, and then its results can be transmitted to all the other robots in the team. The second phase (steps 4–7) is executed independently by each of the robots in a distributed manner during the coverage process itself.

4.1 Multi-level Graph Partitioning

There are several places in the algorithm where we need to use a graph partitioning scheme, in order to divide a given area between several robots. More specifically, we need to apply graph partitioning in the initial allocation of areas to robots, and at the end of the coverage when all the areas have been allocated and there are some idle robots that can help their teammates finish their coverage task.

The problem of partitioning a graph into k equally-sized components is known to be \mathcal{NP} -Hard [10]. Therefore, practical solutions are based on heuristics. Here we use a multi-level graph partitioning algorithm [9], that consists of three main phases: coarsening, partitioning, and uncoarsening (Figure 3). In the coarsening phase, a sequence of smaller graphs, each with fewer vertices is obtained by collapsing vertices and edges into single vertices of the next level, which are called multi-nodes. Then, in the partitioning phase, the coarse graph obtained is partitioned. Lastly, in the uncoarsening phase, the partitioning is refined while the original graph is restored.

4.2 Data Structures

The algorithm maintains a list of connected areas, denoted by \mathcal{A} . Each area can be in one of the following states:

1. $S_{unassigned}$ - unassigned to any robot
2. $S_{assigned}$ - assigned to a robot and not completely covered yet
3. $S_{covered}$ - covered

An area that has been assigned to a robot can change its state to either being covered (if the robot has successfully finished covering it) or to unassigned (if the robot has been stopped by a threat during its coverage). An area that has been completely covered cannot change its state.

For each area A , we maintain the following fields:

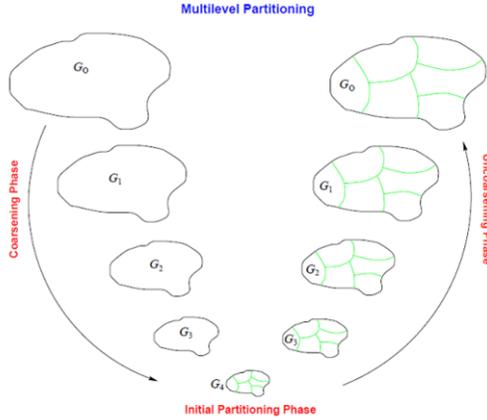


Figure 3. The three phases of multilevel k -way graph partitioning. [9]

- $A.level$ - the threat level this area belongs to
- $A.cells$ - the group of cells that belong to this area
- $A.state$ - the current state of the area
- $A.initial_robots$ - the group of robots initially assigned to this area. If the group contains more than one robot, then this area would need to split between them.
- $A.robot$ - the robot currently assigned to this area

Each robot in the team can be in one of the following states:

1. S_{idle} - waiting for a task assignment. This is the initial state of each robot.
2. $S_{traveling}$ - traveling to the next area that the robot has to cover.
3. $S_{covering}$ - in the process of covering a given area.
4. S_{done} - the robot has finished covering its allocated area and there are no more areas to be covered.
5. S_{dead} - the robot has been hit by a threat.

The transitions between the robot's states are described in section 4.4. For each robot R , we maintain the following fields:

- $R.state$ - the current state of the robot
- $R.area$ - the current area the robot is assigned to
- $R.location$ - the robot's location
- $R.path$ - the path the robot is currently following. This path can be either the transition path to the robot's next allocated area or the coverage path of the area it is currently covering.

In addition to the global map, we assume that the following data structures are shared (and synchronized) between the robots:

1. The status of each cell in the map (visited or not).
2. The list of connected areas and their states.
3. The states of all the robots, and the areas that they are assigned to.

4.3 Initial Allocation

Algorithm 1 is run prior to the coverage itself. It is responsible for pre-processing of the map and the initial allocation of coverage tasks to the robots.

The procedure `Allocate_Areas_To_Robots` allocates the initial areas to the robots. The idea is to initially cover all the safe areas as fast as possible, using all the robots available, before moving to the dangerous areas. The procedure first computes the safest paths of each

Algorithm 1 `Multi_Robot_Adversarial_Coverage`(G, \mathcal{R}, d)

input: G - a grid representing the target area, \mathcal{R} - group of k robots, d - maximal area density

output: \mathcal{A} - list of connected areas

- 1: $\mathcal{A} \leftarrow \emptyset$
 - 2: Group the cells in G into $l + 1$ threat levels, T_0, \dots, T_l
 - 3: **for each** threat level i , $0 \leq i \leq l$ **do**
 - 4: Build the graph H_i induced from the cells in T_i
 - 5: Find the connected components (areas) of H_i using DFS
 - 6: Let A_1, \dots, A_k be the connected areas of H_i
 - 7: **for each** area A_j , $1 \leq j \leq k$ **do**
 - 8: $A_j.state \leftarrow S_{unassigned}$
 - 9: $A_j.level \leftarrow i$
 - 10: Add A_j to \mathcal{A}
 - 11: `Allocate_Areas_To_Robots`($\mathcal{A}, \mathcal{R}, d$)
-

robot to every safe area. Then, it assigns each robot to the safe area with the safest path from its current location, if this area is not too dense with robots. We define a *dense* area as an area whose number of cells is less than d times the number of robots assigned to it, i.e., each robot has less than d cells to cover on average (in experiments we have found that $d = 4$ gave the best results). If the area chosen for the robot is already too dense with robots, then the next safest non-dense area will be allocated to it. After the allocation of areas for all the robots, every area that is assigned to more than one robot is split between the robots using the graph partitioning method described in section 4.1.

1: `procedure` `ALLOCATE_AREAS_TO_ROBOTS`($\mathcal{A}, \mathcal{R}, d$)

- input:** \mathcal{A} - list of connected areas, \mathcal{R} - the group of robots, d - maximal area density
- 2: $\mathcal{S} \leftarrow \{A \mid A \in \mathcal{A} \wedge A.level = 0\}$ ▷ the safe areas
 - 3: **for each** robot $R \in \mathcal{R}$ **do**
 - 4: {Find safest paths to all safe areas}
 - 5: **for each** area $A \in \mathcal{S}$ **do**
 - 6: $P_A \leftarrow \text{Find_Safest_Path_To_Area}(A, R.location)$
 - 7: {Find safest non-dense area}
 - 8: Let A_1, \dots, A_k be the areas sorted by $P_A.cost$
 - 9: $i \leftarrow 1$
 - 10: **while** $i \leq k$ **and** $R.area = null$ **do**
 - 11: **if** $|A.initial_robots| \cdot d \leq |A.cells|$ **then** ▷ check if area is not too dense with robots
 - 12: Add R to $A.initial_robots$
 - 13: **else**
 - 14: $i \leftarrow i + 1$
 - 15: {Split the areas that are allocated to multiple robots}
 - 16: **for each** area $A \in \mathcal{S}$ **do**
 - 17: **if** $|A.initial_robots| > 1$ **then**
 - 18: `Split_Area_Between_Robots`(A)
 - 19: **else if** $|A.initial_robots| = 1$ **then**
 - 20: $R \leftarrow A.initial_robots[0]$
 - 21: $R.path \leftarrow P_A$
 - 22: `Assign_Area_To_Robot`(A, R)
-

The procedure `Find_Safest_Path_To_Area`² searches for the safest path from the current location of the robot to any of the cells that belong to the given area. For that purpose, it runs Dijkstra's shortest paths algorithm on the graph induced from the grid's cells, using the following edge weights:

$$w_{ij} = \begin{cases} p_j/p_{min} & \text{if cell } j \text{ contains a threat} \\ 1/n & \text{otherwise} \end{cases} \quad (3)$$

This weight function ensures that $w_{ij} \geq 1$ for edges that target a dangerous cell, while $w_{ij} = 1/n$ (where n is the grid size) for edges

² The pseudocode of some of the procedures is omitted due to space constraints. The full pseudocode of these procedures can be found at <http://goo.gl/mfYn2Y>

that target a safe cell. This way, the cost of visiting one dangerous cell is greater than the cost of visiting all the safe cells in the grid. Thus, only when there are two equally safe paths, the robot will prefer the shorter one.

The procedure `Assign_Area_To_Robot` updates the data structures to indicate that an area A has been assigned to robot R .

The procedure `Split_Area_Between_Robots` splits the given area into k connected sub-areas using the multi-level graph partitioning algorithm. Then it assigns each robot to the sub-area with the safest path from its current location. Since each sub-area can be allocated to only one robot, we use the Hungarian algorithm [11] for deciding the optimal assignment.

```

1: procedure SPLIT_AREA_BETWEEN_ROBOTS( $A, \mathcal{R}$ )
   input:  $A$  - a connected area,  $\mathcal{R}$  - the group of robots globals:  $\mathcal{A}$  - list of
   connected areas
2:   Build the graph  $G$  induced from the area  $A$ 's cells
3:    $k \leftarrow |\mathcal{R}|$ 
4:   Partition_Graph( $G, k$ )
5:   Let  $G_1, G_2, \dots, G_k$  be the subgraphs created from the partition
6:   for  $i \leftarrow 1$  to  $k$  do
7:     Create a subarea  $A_i$ 
8:      $A_i.cells \leftarrow$  the nodes in  $G_i$ 
9:      $A_i.level \leftarrow A.level$ 
10:    Add  $A_i$  to  $\mathcal{A}$ 
11:    {Compute the safest path of each robot to the sub-area}
12:    for  $j \leftarrow 1$  to  $k$  do
13:       $P_{ij} \leftarrow$  Find_Safest_Path_To_Area( $A_i, R_j.location$ )
14:       $C_{ij} \leftarrow P_{ij}.cost$  ▷ the cost matrix
15:     $O \leftarrow$  Hungarian_Method( $C$ ) ▷  $O$  is the optimal assignment of
    robots to sub-areas
16:     $O_i \leftarrow$  optimal assignment of robot  $i$ 
17:    for  $i \leftarrow 1$  to  $k$  do
18:       $R_i.path \leftarrow P_{O_i, i}$ 
19:      Assign_Area_To_Robot( $O_i, R_i$ )
20:    Remove  $A$  from  $\mathcal{A}$ 

```

4.4 The Coverage Algorithm

We now describe the algorithm that is executed by each robot independently, after the initial allocations have been performed. Algorithm 2 describes the action taken by each robot during one time step.

The robot typically starts in the state $S_{traveling}$, unless there were not enough areas to allocate to all the robots in the initial phase (e.g., there was only one area with 20 cells and there are 10 robots). In this case, the robot starts in S_{idle} and waits for a task assignment. When the robot arrives at the area that it needs to cover, its state changes to $S_{covering}$. If along the way to its designated area, the robot completes a coverage of another area (e.g., an area that consists of only one cell that resides on the connecting path between two larger areas), then this area's state is changed to $S_{completed}$, and is removed from the pool of available areas for coverage. Only when the robot arrives at its designated area, the coverage path of this area is computed. This is because by the time the robot gets to this area, some of its cells may have already been visited along the connecting paths of other robots.

When the robot finishes its coverage task, it is assigned a new area to cover from the pool of unassigned areas and its state changes back to $S_{traveling}$. If there are no more unassigned areas, it joins another covering robot to help it finish its coverage task. If there are no more areas that can be shared, then the robot's state changes to S_{done} and it waits until one of the areas becomes unassigned (e.g., when another robot is stopped). If the robot is stopped during the coverage of its

assigned area or on its way to it, the robot's state changes to S_{dead} and its allocated area returns to the pool of unassigned areas.

Algorithm 2 Robot_Action(R)

```

1: switch  $R.state$  do
2:   case  $S_{traveling}$ 
3:      $c \leftarrow$  next cell on  $R.path$ 
4:     Mark  $c$  as visited
5:      $A \leftarrow$  the area that contains  $c$ 
6:     if all cells in  $A$  are visited then
7:        $A.state \leftarrow S_{completed}$ 
8:     if robot was hit by a threat in  $c$  then
9:        $R.state \leftarrow S_{dead}$ 
10:      Reallocate_Area( $R.area$ )
11:    else if  $c$  is the last cell on  $R.path$  then
12:       $R.path \leftarrow$  Area_Coverage( $R.area, R.location$ )
13:       $R.state \leftarrow S_{covering}$ 
14:    case  $S_{covering}$ 
15:       $c \leftarrow$  next cell on  $R.path$ 
16:      Mark  $c$  as visited
17:      if robot was hit by a threat in  $c$  then
18:         $R.state \leftarrow S_{dead}$ 
19:        Reallocate_Area( $R.area$ )
20:      else if  $c$  is the last cell on  $R.path$  then
21:         $R.area.state \leftarrow S_{completed}$ 
22:        Allocate_Next_Area( $R$ )
23:    case  $S_{done}, S_{idle}$ 
24:      if an unassigned area exists in  $\mathcal{A}$  then
25:        Allocate_Next_Area( $R$ )
26:    case  $S_{dead}$ 
27:       $R.area.state \leftarrow S_{unassigned}$ 

```

The procedure `Allocate_Next_Area` allocates a new area to cover for a robot that has completed its coverage task. If there are any unassigned areas, the robot is assigned to the safest unassigned area with the safest path from its current location. If all the areas are already assigned, the robot joins another covering robot to help it finish its coverage task.

The procedure `Find_Area_To_Share` tries to find for a given idle robot an assigned area that it can help finish covering. If the path from the given robot's location to the designated area is longer than the number of unvisited cells in that area, then there is no point of sending the robot there, since by the time the robot arrives there, its coverage will have been completed. If an area that can be shared has been found, its connected unvisited parts are defined as new areas and added to the pool of unassigned areas instead of the original area. If there is only one such part (e.g., its assigned robot has not started covering it), then it is split into two balanced parts. Finally, both the assigned robot and the idle robot are (re-)assigned to the sub-areas with the safest paths from their current location.

The procedure `Assign_Robot_To_Safest_Area` finds the safest path from the robot's location to each of the given areas and assigns the robot to the area with the safest path.

For the coverage of each area, our algorithm is based on the Greedy Adversarial Coverage (GAC), the state-of-the-art solution to the single-robot adversarial coverage problem described in [20]. GAC follows a greedy approach, where in each step it leads the robot to the safest nearest cell to its current location which has not been covered yet.

We have modified the original coverage algorithm to take into account cells that have already been visited in the target area (see Algorithm 3). By the time the designated area is allocated to the robot and the robot reaches this area, other robots may already have visited some cells in this area on the way to their own designated areas. Thus, in order to avoid repeated coverage of these cells, we have changed the algorithm to cover only unvisited cells in the given area. When

transitioning between unvisited cells in the target area, the robot is allowed to visit cells that belong to other areas (if they make the connecting path safer).

Algorithm 3 Area_Coverage(A, s)

input: an area A , and a starting cell s

output: a coverage path P that covers all unvisited cells in A

```

1:  $P \leftarrow \emptyset$ 
2: Build the graph  $H$  induced from  $A$ 's cells with the weight function  $w$ 
   from eq. (3)
3: Add the starting cell  $s$  to  $P$ 
4: Mark  $s$  as visited
5: while there are unvisited cells in  $A$  do
6:   Run Dijkstra's shortest paths algorithm on  $H$  from  $s$ 
7:    $v \leftarrow$  an unvisited node in  $A$  with minimum weighted distance from
    $s$ 
8:   Add the path  $s \rightsquigarrow v$  to  $P$ 
9:   Mark  $v$  as visited
10:   $s \leftarrow v$ 
11: return  $P$ 

```

Finally, the procedure `Reallocate_Area` is used to reallocate an area whose coverage was stopped in the middle, because its assigned robot was hit by a threat. The procedure finds all the unvisited parts of the given area and creates new unassigned areas from them. Then these areas are added to the list of connected areas instead of the given area. The next idle robot will be assigned to one of these sub-areas in its next cycle (lines 24–25 in algorithm 2).

5 Analysis of the MRAC algorithm

We now analyze the MRAC algorithm. We first prove that the algorithm is complete, i.e., that it generates coverage paths that together cover all the accessible cells in the given area.

Theorem 1 (Completeness) *Algorithm MRAC generates paths for the robots that together cover every cell accessible from their starting locations.*

Proof. MRAC partitions the target area into k connected areas, whose union is equal to the target area. Each of these areas is eventually assigned to one of the robots, since no robot remains idle while there are more areas to be covered (lines 24–25 in algorithm 2). The areas are covered by using the GAC algorithm. Previous work has shown that GAC is complete, i.e. that it produces a path that covers all the accessible cells in its given area (Theorem 8 in [20]). Thus, the union of the coverage paths of the k connected areas covers every accessible cell in the target area. \square

As key motivation for using multiple robots comes from robustness concerns, we now prove that MRAC is robust to robotic failures.

Theorem 2 (Robustness) *Algorithm MRAC guarantees that the coverage will be completed as long as at least one robot remains active.*

Proof. The target area is divided into a set of connected areas. Each area is eventually assigned to one of the robots. If a robot is stopped while covering its assigned area, the uncovered parts of this area are returned to the pool of unassigned areas (in procedure `Reallocate_Area`). These sub-areas are eventually assigned to one of the remaining robots, since no robot remains idle while there are more areas to be covered (lines 24–25 in algorithm 2). \square

We now provide a bound on the minimum expected number of cells that the robots will cover when following the policy generated

by the MRAC algorithm. We will use the definition of expected coverage from [20]. Given a coverage path A of a robot R , we denote the sequence of new cells discovered along A by (b_1, \dots, b_n) , and the number of new cells visited by the robot until it is stopped by C_A . Furthermore, for each cell in the sequence b_i , we will denote the sub-path in A that leads from the origin cell a_1 to it by g_i . Then, under the threat probability function p , the expected number of new cells that robot R visits can be expressed as:

$$E(R) = \sum_{i \in (b_1, \dots, b_n)} \prod_{j \in g_i} (1 - p_j) \quad (4)$$

Theorem 7 in [20] provides a bound on the minimum expected number of cells that a single robot will be able to cover, given its coverage path A and the threat probability function p .

Let l be the number of dangerous threat levels and the threat probabilities of these levels be p_0, \dots, p_l ($p_0 = 0$). Let $A_{i,1}, \dots, A_{i,k_i}$ be the connected areas of threat level i , arranged in the order of their visit by the robot R . Let $|A_{i,j}|$ be the size of area $A_{i,j}$ and $m_{i,j}$ the number of cell visits needed to cover this area ($m_{i,j} \geq |A_{i,j}|$). Let $C_{i,j}$ be the set of cells on the connecting path between two consecutive areas $A_{i,j}$ and $A_{i,j+1}$, and C_{i,k_i} be the set of cells on the connecting path between the last area of threat level i and the first area of threat level $i + 1$. Denote by $P(C_{i,j})$ the probability to traverse the cells in $C_{i,j}$ without being hit by a threat. Then the expected number of cells robot R will be able to cover before it is stopped is at least:

$$E(R) \geq |A_{0,0}| + \sum_{i=0}^l \sum_{j=1}^{k_i} \left[\prod_{x=0}^{i-1} \left[(1 - p_x)^{\sum_{y=1}^{k_x} m_{x,y}} \prod_{y=1}^{k_x} P(C_{x,y}) \right] \cdot (1 - p_i)^{\sum_{y=1}^j m_{i,y}} \prod_{y=1}^{j-1} P(C_{i,y}) \right] |A_{i,j}| \quad (5)$$

We now use this theorem to provide a lower bound on the expected coverage that can be attained by the entire robotic team.

Theorem 3 *Let \mathcal{R} be a group of k robots $\mathcal{R} = \{R_1, \dots, R_k\}$. Let l be the number of dangerous threat levels. Let \mathcal{A} be the group of connected areas and let \mathcal{C} be the set of cells on the connecting path between two areas. Then the expected number of cells the team \mathcal{R} will be able to cover before all robots in \mathcal{R} are stopped is at least:*

$$E(\mathcal{R}) \geq \sum_{i=1}^k E(R_i) - |\mathcal{C}| \quad (6)$$

Proof. Since different robots in the team cover different areas (when one area is assigned to more than one robot, it is split between them), the expected number of cells that will be covered by the entire team is equal to the total number of cells that will be covered by each robot individually minus the number of cells on the connecting paths between areas, since those might be visited multiple times by different robots. \square

Notice that the ideal expected coverage is attained when the robots visit the cells precisely in increasing order of their threat probabilities, i.e., when they first visit all the safe cells, then all the cells with threat probability p_1 , etc. Thus, if we denote the cells that belong to threat level i by $c_{i,1}, \dots, c_{i,n_i}$, then the ideal expected coverage is:

$$\sum_{i=0}^l \sum_{j=1}^{n_i} \left[\prod_{x=0}^{i-1} (1 - p_x)^{n_x} \right] \cdot (1 - p_i)^{j-1} \quad (7)$$

In most environments the ideal expected coverage cannot be attained, since typically some of the cells that belong to a given threat level are disconnected, and the robots have to visit cells that belong to a higher threat level or revisit threat points that have already been covered in order to move between the disconnected cells. In these cases, the cell visits cannot precisely follow the order of the threat levels.

By comparing equations (6) and (7), we can see that the gap between the ideal expected coverage and the expected coverage achieved by MRAC depends on the quality of the coverage algorithm of each connected area (which determines the gap between n_x and $\sum_{y=1}^{k_x} m_{x,y}$). Finding an optimal solution to the coverage problem is \mathcal{NP} -Hard [20]. However, MRAC uses for the coverage of each area the algorithm GAC, which creates a close-to-optimal coverage paths [20]. In fact, MRAC can utilize any single-robot coverage algorithm for the internal coverage of each area.

The next theorem provides a bound on the coverage time that can be attained by the robots following the policy generated by MRAC.

Theorem 4 *The worst-case coverage time for k robots is equal to that of a single robot. The best-case coverage time for k robots is approximately $1/k$ the coverage time of a single robot.*

Proof. The worst-case scenario is where all the robots start at locations that are all inside dangerous areas, and thus they will be stopped by threats in the beginning of their coverage paths. In such case, using more than one robot will not improve the total coverage time. The best-case scenario is when the target area consists of only one contiguous safe area (or a few large safe areas with connecting paths that have very low risk). In such scenario, the area is split into k almost equally-sized components that are covered concurrently by the k different robots. In such scenario, the coverage time of MRAC highly depends on the quality of the graph partitioning algorithm, which has no theoretical guarantee, however, in practice, it almost always generated equally-sized sub-graphs. The coverage time in this case also depends on the initial locations of the robots, e.g., if the robots start the coverage at the same location, it will take them some time to move to their assigned areas. □

6 Experimental Results

We have fully implemented the MRAC algorithm and evaluated it in various types of simulated environments with varying parameters, such as map size, number of robots, number of threat levels, distribution of threats, number of obstacles, number of threats, etc. We provide here the most interesting results from our experiments.

First, we demonstrate the algorithm’s results on a specific grid map of size 20×20 (Figure 4). In this example, the maps contained 10 contiguous threat areas which were divided into 5 different threat levels with the following threat probabilities: 4%, 8%, 12%, 16%, and 20%. We ran the algorithm on this map with two team sizes of 4 robots and 10 robots. All the robots start in the upper-left corner of the area (as shown in Figure 1). The figure shows the locations of the robots after the coverage has finished, and the total number of times each cell in the grid has been visited by the robots. Using 10 robots instead of 4 robots has increased the coverage percentage from 80.33% to 97.33% and reduced the coverage time from 311 to 244

time steps. As can be seen in both maps, more dangerous areas are less frequently visited by the robots.

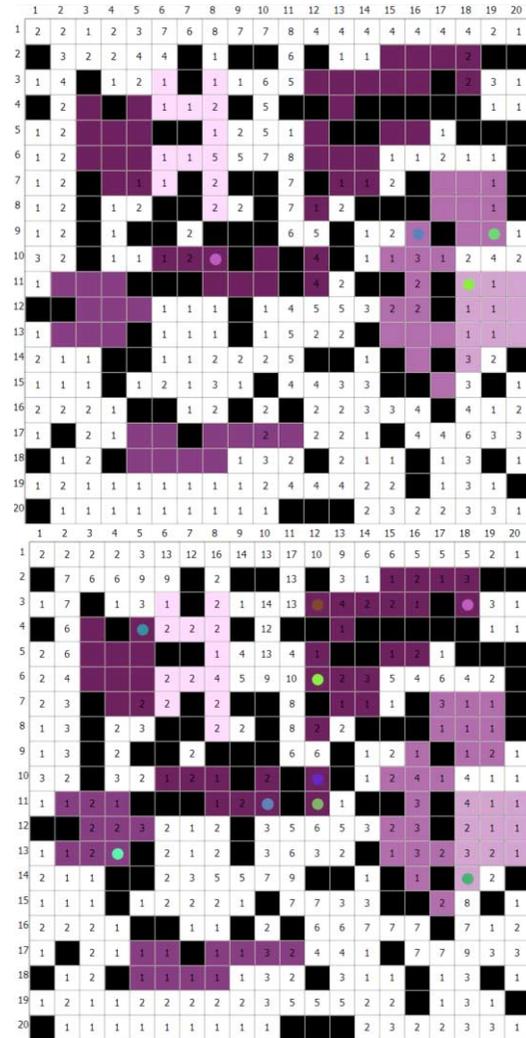


Figure 4. A sample map with the results of MRAC with 4 robots (upper figure) and 10 robots (lower figure). The number in each cell indicates the number of visits in that cell.

Second, we examined the effect of changing the team size on the performance of the algorithm. We used 500 randomly-generated maps of size 20×20 , with varying number of robots between 1 and 20. In all the experiments, the obstacles ratio and the threats ratio were both set to 25% of the cells. The maps contain 10 contiguous threat areas which are divided into 5 different threat levels with the following threat probabilities: 4%, 8%, 12%, 16%, and 20%. The locations of the obstacles were randomly chosen. Figure 5 shows the percentage of covered area and the coverage time for each team size.

As can be clearly seen in the graph, the coverage percentage increases and the coverage time decreases as we add more robots to the team. Although the area that the robots are able to cover increases as the team grows, they are able to cover it more quickly. The small increase in the coverage time in the transition from one robot to two robots is due to the fact that the area that two robots are able to cover is significantly larger than one robot (62% for two robots vs. 44% for

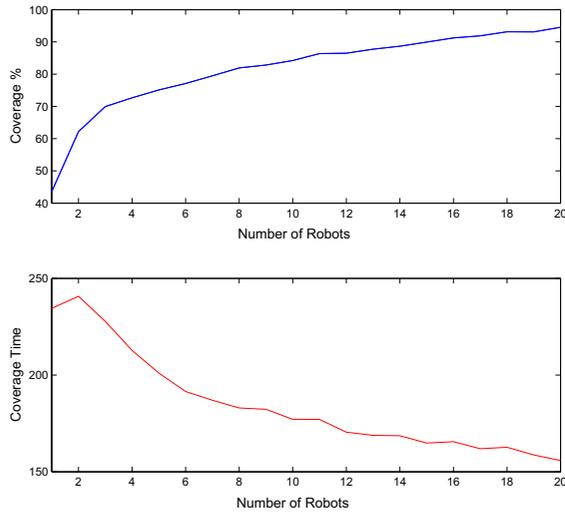


Figure 5. Comparing coverage percentage and coverage time for 1 to 20 robots in environments with contiguous areas of threats.

one robot), thus it takes more time until the two robots are stopped despite the division of the workload between them (e.g., one robot may be stopped much earlier than the second one).

Next, we examined environments where the threats are randomly scattered across the map. We kept all the other map settings as in the previous experiment (i.e., the same ratio of obstacles and threats, and the same number of threat levels). Figure 6 shows the results.

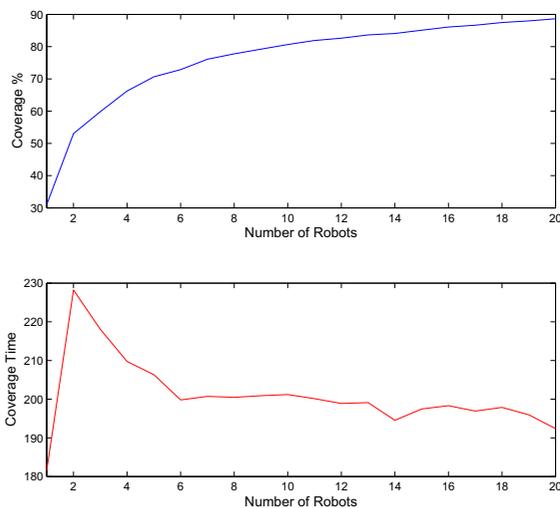


Figure 6. Comparing coverage percentage and coverage time for 1 to 20 robots in environments with randomly-scattered threats.

As previously, the coverage percentage increases and the coverage time decreases as we add more robots to the team. However, the coverage percentage that the team is able to achieve in this type of envi-

ronments is smaller than in environments with contiguous dangerous areas, and it takes longer time for the team to complete the coverage. When the threats are scattered, the effectiveness of the team is mitigated, since the robots spend much of their time in moving between cells with different threat levels than in covering large areas. As a consequence, locations that reside on connecting paths between cells that belong to different threat levels are repetitively visited by different robots.

Lastly, we have examined the effect of changing the threats ratio in the environment on the ability of the robotic team to cover it. Figure 7 shows the coverage percentage and coverage time for various threat ratios between 0% and 35%. In all the experiments we used teams of 8 robots to cover the area and the threats were concentrated in 10 dangerous areas. As expected, adding more threats the environment causes the team to cover smaller percentage of the area. Note that when the threats ratio is more than 15% the coverage time begins to decrease, since the area that needs to be covered gets smaller thus the team is able to cover it more quickly.

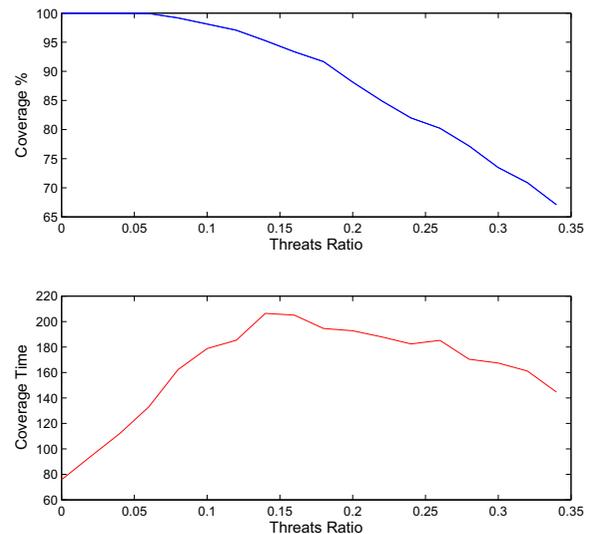


Figure 7. Comparing coverage percentage and coverage time for varying threat ratios between 0 and 0.35.

7 Conclusions and Future Work

In this paper, we described a multi-robot coverage algorithm for adversarial environments that is complete and robust in face of threats harming the robots. Our approach is based on decomposition of the target area into areas of different danger levels, and assignment of the robots to these areas based on the safety of the paths leading to them. In our approach no robot remains idle while there are areas to be covered. We examined the efficiency of the algorithm in terms of both the survivability of the team and the coverage time, and have shown that adding more robots to the team significantly improves both measures. For future work, we would like to extend the multi-robot coverage algorithm to online scenarios, in which the robots are not given a map of the area in advance. In addition, we would like to consider non-stationary environments, where the locations of the threat points may change over time.

REFERENCES

- [1] Noa Agmon, Noam Hazon, and Gal A Kaminka, 'Constructing spanning trees for efficient multi-robot coverage', in *IEEE International Conference on Robotics and Automation (ICRA-06)*, pp. 1698–1703, (2006).
- [2] J Colegrave and A Branch, 'A case study of autonomous household vacuum cleaner', *AIAA/NASA CIRFFSS*, 107, (1994).
- [3] Yehuda Elmaliach, Noa Agmon, and Gal A Kaminka, 'Multi-robot area patrol under frequency constraints', *Annals of Mathematics and Artificial Intelligence*, **57**(3-4), 293–320, (2009).
- [4] Hermann Endres, Wendelin Feiten, and Gisbert Lawitzky, 'Field test of a navigation system: Autonomous cleaning in supermarkets', in *IEEE International Conference on Robotics and Automation (ICRA-98)*, volume 2, pp. 1779–1781, (1998).
- [5] Yoav Gabriely and Elon Rimon, 'Competitive on-line coverage of grid environments by a mobile robot', *Computational Geometry*, **24**(3), 197–224, (2003).
- [6] Enric Galceran and Marc Carreras, 'A survey on coverage path planning for robotics', *Robotics and Autonomous Systems*, **61**(12), 1258–1276, (2013).
- [7] Anouck R Girard, Adam S Howell, and J Karl Hedrick, 'Border patrol and surveillance missions using multiple unmanned air vehicles', in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pp. 620–625, (2004).
- [8] Noam Hazon and Gal A Kaminka, 'On redundancy, efficiency, and robustness in coverage for multiple robots', *Robotics and Autonomous Systems*, **56**(12), 1102–1114, (2008).
- [9] George Karypis and Vipin Kumar, 'Multilevel k-way partitioning scheme for irregular graphs', *Journal of Parallel and Distributed computing*, **48**(1), 96–129, (1998).
- [10] Robert Krauthgamer, Joseph Seffi Naor, and Roy Schwartz, 'Partitioning graphs into balanced components', in *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 942–949, (2009).
- [11] Harold W Kuhn, 'The Hungarian method for the assignment problem', *Naval research logistics quarterly*, **2**(1-2), 83–97, (1955).
- [12] Chaomin Luo, Simon X Yang, and Deborah A Stacey, 'Real-time path planning with deadlock avoidance of multiple cleaning robots', in *IEEE International Conference on Robotics and Automation (ICRA-03)*, volume 3, pp. 4080–4085, (2003).
- [13] Chaomin Luo, Simon X Yang, Deborah A Stacey, and Jan C Jofriet, 'A solution to vicinity problem of obstacles in complete coverage path planning', in *IEEE International Conference on Robotics and Automation (ICRA-02)*, volume 1, pp. 612–617, (2002).
- [14] Jean Daniel Nicoud and Maki K Habib, 'The Pemex-B autonomous demining robot: perception and navigation strategies', in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 'Human Robot Interaction and Cooperative Robots'*, volume 1, pp. 419–424, (1995).
- [15] David Portugal and Rui Rocha, 'A survey on multi-robot patrolling algorithms', in *Technological Innovation for Sustainability*, 139–146, Springer, (2011).
- [16] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset, 'Efficient boustrophedon multi-robot coverage: an algorithmic approach', *Annals of Mathematics and Artificial Intelligence*, **52**(2-4), 109–142, (2008).
- [17] Israel A Wagner, Michael Lindenbaum, and Alfred M Bruckstein, 'Distributed covering by ant-robots using evaporating traces', *IEEE Transactions on Robotics and Automation*, **15**(5), 918–933, (1999).
- [18] Roi Yehoshua and Noa Agmon, 'Adversarial modeling in the robotic coverage problem', in *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-15)*, pp. 891–899, (2015).
- [19] Roi Yehoshua and Noa Agmon, 'Online robotic adversarial coverage', in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-15)*, pp. 3830–3835, (2015).
- [20] Roi Yehoshua, Noa Agmon, and Gal A Kaminka, 'Robotic adversarial coverage of known environments', *International Journal of Robotics Research*, Advance online publication. doi:10.1177/0278364915625785, (2016).
- [21] Roi Yehoshua, Noa Agmon, and Gal A Kaminka, 'Robotic adversarial coverage: Introduction and preliminary results', in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-13)*, pp. 6000–6005, (2013).
- [22] Roi Yehoshua, Noa Agmon, and Gal A Kaminka, 'Safest path adversarial coverage', in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-14)*, pp. 3027–3032, (2014).

Parameterized Complexity Results for the Kemeny Rule in Judgment Aggregation

Ronald de Haan¹

Abstract. We investigate the parameterized complexity of computing an outcome of the Kemeny rule in judgment aggregation, providing the first parameterized complexity results for this problem for any judgment aggregation procedure. As parameters, we consider (i) the number of issues, (ii) the maximum size of formulas used to represent issues, (iii) the size of the integrity constraint used to restrict the set of feasible opinions, (iv) the number of individuals, and (v) the maximum Hamming distance between any two individual opinions, as well as all possible combinations of these parameters. We provide parameterized complexity results for two judgment aggregation frameworks: formula-based judgment aggregation and constraint-based judgment aggregation. Whereas the classical complexity of computing an outcome of the Kemeny rule in these two frameworks coincides, the parameterized complexity results differ.

1 Introduction

The area of computational social choice places computational aspects of social choice procedures at a first-class level among selection criteria for the choice between such procedures. For example, since the seminal work of Bartholdi, Tovey and Trick [3], it has been well known that for many voting rules the problem of computing who won a particular election is computationally intractable. As a result, the computational complexity of this winner determination problem plays an important role in the selection of what voting scheme to use. Traditionally, classical computational complexity theory has been used to provide qualitative information about the difficulty of relevant computational problems for social choice procedures. For instance, completeness results for the classical complexity classes NP and Θ_2^P abound in the computational social choice literature (see, e.g., [3, 8, 9, 21, 29, 40, 41]).

However, classical complexity theory—being a worst-case framework that measures the running time of algorithms in terms of only the bit-size of the input—is mostly blind to what aspects of the input underlie negative complexity results. Consequently, negative complexity results tend to be interpreted overly pessimistically. An illustrative example of this concerns the winner determination problem for the Kemeny voting rule. This problem is Θ_2^P -complete in general (which rules out algorithms that work efficiently across the board), but efficient algorithms do exist for cases where the number of candidates is small [5, 6, 30, 33, 39].

The multidimensional framework of *parameterized complexity* [13, 14, 23, 38] offers a mathematically rigorous theory to analyze the computational complexity of problems on the basis of more than just the input size in bits. Therefore, using this framework, one can give much more informative complexity results that are sensitive to

various aspects of the problem input—in principle, any aspect of the input can be taken into account. In the analysis of voting procedures, parameterized complexity has been used widely, to give a more accurate picture of the complexity of many related computational problems, including the problem of winner determination (see, e.g., [5, 6, 33]). In the area of judgment aggregation, however, parameterized complexity has been used to analyze the computational complexity of problems only in very few cases [4, 19]. The fundamental problem of computing the outcome of judgment aggregation procedures, as of yet, remains uninvestigated from a parameterized complexity point of view.

We hope to initiate a structured parameterized complexity investigation of the problem of computing the outcome of judgment aggregation procedures. Judgment aggregation studies the process of combining individual judgments on a set of related propositions of the members of a group into a collective judgment reflecting the views of the group as a whole [15, 26, 34, 35]. Seen from a classical complexity point of view, computing the outcome of many judgment aggregation procedures is Θ_2^P -complete. However, as these negative results pertain only to the case where every possible input needs to be considered, there is a lot of room for relativizing these negative results by taking a parameterized complexity perspective and considering various (combinations of) reasonable restrictions on the inputs.

Contribution In this paper, we start the parameterized complexity investigation of judgment aggregation procedures by considering one of the most prominent procedures: the Kemeny procedure for judgment aggregation (or *Kemeny rule*, for short).² The unrestricted problem of computing an outcome for this rule is Θ_2^P -complete [17, 32]. We consider a number of natural parameters for this problem—capturing various aspects of the problem input that can reasonably be expected to be small in some applications—and we give a complete parameterized complexity classification for the problem of computing the outcome of the Kemeny rule, for every combination of these parameters. The parameters that we consider are:

- the number n of issues that the individuals (and the group) form an opinion on;
- the maximum size m of formulas used to represent the issues;
- the size c of the integrity constraint used to limit the set of feasible opinions;
- the number p of individuals; and
- the maximum (Hamming) distance h between any two individual opinions.

¹ Technische Universität Wien, email: dehaan@ac.tuwien.ac.at

² This procedure has also been called “MWA” [32], “Median rule” [37], “Simple scoring rule” [11], and “Prototype-Hamming rule” [36].

The results in this paper open up interesting and natural lines of future research. A similar parameterized complexity analysis can be performed for the problem of computing the outcome of other judgment aggregation procedures. Moreover, further parameters can be taken into account in future parameterized complexity analyses of the problem.

We develop parameterized complexity results for two formal frameworks for judgment aggregation: *formula-based judgment aggregation* and *constraint-based judgment aggregation* (the former is often simply called ‘judgment aggregation’ [12, 17, 32] and the latter is also called ‘binary aggregation with integrity constraints’ [24, 25])—we define these two frameworks in detail in Section 3. In general, the computational complexity of computing the outcome of a judgment aggregation procedure might differ for these two frameworks [16], but for the Kemeny rule this problem is Θ_2^p -complete in both frameworks [17, 24].

Nonstandard Parameterized Complexity Tools Since the invention of parameterized complexity theory, it has been applied mostly to problems that are in NP. As a result, the most commonly used parameterized complexity toolbox is insufficient to perform a complete parameterized complexity analysis of problems that are beyond NP (such as the Θ_2^p -complete problem of computing the outcome of the Kemeny procedure). Recently, various novel parameterized complexity tools have been developed that aid in analyzing the parameterized complexity of problems beyond NP [19, 20, 28]. The parameterized complexity results in this paper feature one of these innovative parameterized complexity tools: the class $\text{FPT}^{\text{NP}}[\text{few}]$, which consists of problems that can be solved by a fixed-parameter tractable that can query an NP oracle a small number of times (that is, the number of oracle queries depends only on the parameter value). We define this class in detail in Section 2, where we discuss relevant notions from parameterized complexity.

Parameters	parameterized complexity result	
c, n, m	in FPT	(Proposition 3)
h, p	in $\text{FPT}^{\text{NP}}[\text{few}]$	(Proposition 1)
n	in $\text{FPT}^{\text{NP}}[\text{few}]$	(Proposition 2)
h, n, m, p	$\text{FPT}^{\text{NP}}[\text{few}]$ -hard	(Proposition 8)
c, h, n, p	$\text{FPT}^{\text{NP}}[\text{few}]$ -hard	(Proposition 9)
c, h, m, p	$\text{FPT}^{\text{NP}}[\text{few}]$ -hard	(Proposition 10)
c, h, m	para- Θ_2^p -hard	(Corollary 6)
c, m, p	para- Θ_2^p -hard	(Proposition 7)

Table 1. Parameterized complexity results for $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$.

Parameters	parameterized complexity result	
n	in FPT	(Proposition 11)
c	in FPT	(Proposition 12)
h	in XP	(Proposition 13)
h, p	W[SAT]-hard	(Proposition 14)
p	para- Θ_2^p -hard	(Proposition 15)

Table 2. Parameterized complexity results for $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$.

Overview of Results We provide a parameterized complexity classification for the problem of computing an outcome of the Kemeny rule, for all possible combinations of the parameters that we consider—both (1) in the framework of formula-based judgment aggregation and (2) in the framework of constraint-based judgment aggregation.

For the framework of formula-based judgment aggregation, we give a tight classification for each possible case. In particular, we show the following. When parameterized by any set of parameters that includes c , n and m , the problem is fixed-parameter tractable (Proposition 3). Otherwise, when parameterized by any set of parameters that includes either n or both h and p , the problem is $\text{FPT}^{\text{NP}}[\text{few}]$ -complete (Propositions 1, 2, 8, 9 and 10). For all remaining cases, the problem is para- Θ_2^p -complete (Corollary 6 and Proposition 7).

For the framework of constraint-based judgment aggregation, we show the following results. When parameterized by any set of parameters that includes either c or n , the problem is fixed-parameter tractable (Propositions 11 and 12). Otherwise, when parameterized by any set of parameters that includes h , the problem is W[SAT]-hard and is in XP (Propositions 13 and 14). For all remaining cases, the problem is para- Θ_2^p -complete (Proposition 15). The results for the formula-based judgment aggregation framework are summarized in Table 1, and the results for the constraint-based framework are summarized in Table 2.

Interpretation of Results From a classical complexity point of view, the problem of computing outcomes of the Kemeny rule in judgment aggregation is equally difficult for both the formula-based and the constraint-based frameworks [17, 24], despite the notable differences between the frameworks. Our results show that the use of a parameterized complexity perspective more accurately displays the difference between the frameworks, in terms of their computational properties.

For instance, in the constraint-based framework, the logical relations between the issues are represented only using the integrity constraint, whereas in the formula-based framework, these logical relations are expressed by both the integrity constraint and the agenda formulas. As a result, in the constraint-based framework, taking either c or n as parameter suffices to get fixed-parameter tractability, whereas in the formula-based framework, one needs the combination of c , n and m as parameter for fixed-parameter tractability. In fact, in the formula-based framework, restricting both c and m to constant values (but leaving n unrestricted) does not decrease the parameterized complexity.

Another difference between the frameworks is that in the formula-based framework, checking whether an opinion is feasible is an NP-complete problem, whereas this is solvable in polynomial time in the constraint-based framework. Intuitively, this difference results in the fact that computing outcomes for the Kemeny rule is in XP when parameterized by h in the constraint-based framework, but in the formula-based framework, one needs both h and p as parameters to even get membership in $\text{FPT}^{\text{NP}}[\text{few}]$.

The formula-based framework has the advantage that it allows more succinct encodings than the constraint-based framework [16]. Our results show that this advantage comes with a drawback: when encoding judgment aggregation scenarios using the formula-based framework, one needs to be more careful to keep various parameter values low, in order to support more efficient algorithmic methods.

Roadmap In the remainder of Section 1, we discuss relevant related work. Then, in Section 2, we give a brief overview of the concepts and tools from the theory of (parameterized) complexity that we use in this paper. In Section 3, we introduce the two formal judgment aggregation frameworks, and we formally define the computational problem of computing an outcome for the Kemeny rule, as well as all the parameterized variants of this problem that we consider. We

provide the parameterized complexity results for all parameterized variants of the problem (for both judgment aggregation frameworks) in Section 4. Finally, we conclude in Section 5.

Related Work Parameterized complexity theory has been used to investigate the complexity of the winner determination problem in voting (which is analogous to the problem of computing the outcome of a judgment aggregation procedure) for several voting rules [5, 6, 33]. Parameterized complexity has also been used to study various other problems in the area of judgment aggregation, such as problems related to bribery [4, 19]. The complexity of computing outcomes for the Kemeny procedure in judgment aggregation (and other procedures) has been studied from a classical complexity point of view [17, 18, 24, 32]. It has also been studied what influence the choice of formal framework to model the setting of judgment aggregation has on the (classical) complexity of computing outcomes for various judgment aggregation procedures [16].

2 Parameterized Complexity

We begin by briefly introducing the relevant concepts and notation from propositional logic and (parameterized) complexity theory. We use the notation $[n]$, for any $n \in \mathbb{N}$, to denote the set $\{1, \dots, n\}$.

Propositional Logic Propositional formulas are constructed from propositional variables using the Boolean operators \wedge , \vee , \rightarrow , and \neg . A propositional formula is *doubly-negated* if it is of the form $\neg\neg\psi$. For every propositional formula φ , we let $\sim\varphi$ denote the *complement* of φ , i.e., $\sim\varphi = \neg\varphi$ if φ is not of the form $\neg\psi$, and $\sim\varphi = \psi$ if φ is of the form $\neg\psi$. For a propositional formula φ , the set $\text{Var}(\varphi)$ denotes the set of all variables occurring in φ . We use the standard notion of (*truth assignments*) $\alpha : \text{Var}(\varphi) \rightarrow \{0, 1\}$ for Boolean formulas and *truth* of a formula under such an assignment. We define the size $|\varphi|$ of a propositional formula φ as the total number of (occurrences of) Boolean operators and propositional variables in φ .

Classical Complexity We assume the reader to be familiar with the most common concepts from complexity theory, such as the complexity classes P and NP. These basic notions are explained in textbooks on the topic; see, e.g., [2]. In this paper, we will also refer to the complexity class Θ_2^P , that consists of all decision problems that can be solved by a polynomial-time algorithm that queries an NP oracle $O(\log n)$ times. The following problem is complete for the class Θ_2^P under polynomial-time reductions [7, 31, 44].

MAX-MODEL

Instance: A satisfiable propositional formula φ , and a variable $w \in \text{Var}(\varphi)$.

Question: Is there a model of φ that sets a maximal number of variables in $\text{Var}(\varphi)$ to true (among all models of φ) and that sets w to true?

Parameterized Complexity Next, we introduce the relevant concepts of parameterized complexity theory. For more details, we refer to textbooks on the topic [10, 13, 14, 23, 38]. An instance of a parameterized problem is a pair (x, k) where x is the main part of the instance, and k is the parameter. A parameterized problem is *fixed-parameter tractable* if instances (x, k) of the problem can be solved by a deterministic algorithm that runs in time $f(k)|x|^c$, where f is a computable function of k , and c is a constant (algorithms running within such time bounds are called *fpt-algorithms*). FPT denotes the

class of all fixed-parameter tractable problems. It often makes sense to consider parameterized problems with multiple parameters. When considering multiple parameters, we take their sum as a single parameter.

Parameterized complexity also offers a *completeness theory*, similar to the theory of NP-completeness, that provides a way to obtain evidence that a parameterized problem is not fixed-parameter tractable. Hardness for parameterized complexity classes is based on fpt-reductions, which are many-one reductions where the parameter of one problem maps into the parameter for the other. More specifically, a parameterized problem Q is fpt-reducible to another parameterized problem Q' if there is a mapping R that maps instances of Q to instances of Q' such that (i) $(I, k) \in Q$ if and only if $R(I, k) = (I', k') \in Q'$, (ii) $k' \leq g(k)$ for a computable function g , and (iii) R can be computed in time $f(k)|I|^c$ for a computable function f and a constant c . Central to the completeness theory are the classes of the Weft hierarchy, including the class W[SAT]. The parameterized complexity class W[SAT] can be characterized as the set of those parameterized problems that can be fpt-reduced to the problem MONOTONE-WSAT [1]. This problem is defined as follows.

MONOTONE-WSAT

Instance: A monotone propositional formula φ —i.e., φ contains no negations—and a positive integer k .

Parameter: k .

Question: Does there exist a truth assignment that sets exactly k variables in $\text{Var}(\varphi)$ true and that satisfies φ .

Moreover, the parameterized complexity class XP consists of all problems that can be solved in time $O(n^{f(k)})$, for some computable function f , where n is the input size and k is the parameter value.

The following parameterized complexity classes are analogues to classical complexity classes. Let K be a classical complexity class, e.g., Θ_2^P . The parameterized complexity class para-K is then defined as the class of all parameterized problems Q for which there exist a computable function f and a problem $Q' \in K$ such that for all instances (x, k) we have that $(x, k) \in Q$ if and only if $(x, f(k)) \in Q'$. Intuitively, the class para-K consists of all problems that are in K after a precomputation that only involves the parameter. A parameterized problem is para-K-hard if it is K-hard already for a constant value of the parameter [22].

The final parameterized complexity class that we consider is $\text{FPT}^{\text{NP}}[\text{few}]$, consisting of all parameterized problems that can be solved by an fpt-algorithm that queries an NP oracle at most $f(k)$ many times, where f is some computable function and where k denotes the parameter value [19, 20, 27]. Intuitively, this class consists of those problems that can be reduced to SAT by a Turing reduction that runs in fixed-parameter tractable time, and queries the oracle at most $f(k)$ times. The following parameterized variant of MAX-MODEL is $\text{FPT}^{\text{NP}}[\text{few}]$ -complete under fpt-reductions.

LOCAL-MAX-MODEL

Instance: A satisfiable propositional formula φ , a subset $X \subseteq \text{Var}(\varphi)$ of variables, and a variable $w \in X$.

Parameter: $|X|$.

Question: Is there a model of φ that sets a maximal number of variables in X to true (among all models of φ) and that sets w to true?

3 Judgment Aggregation

Next, we introduce the two formal judgment aggregation frameworks that we use in this paper: *formula-based judgment aggregation* (as

used by, e.g., [12, 17, 32]) and *constraint-based judgment aggregation* (as used by, e.g., [24]). For both frameworks, we will also define the computational problem $\text{OUTCOME}(\text{KEMENY})$ of computing an outcome of the Kemeny procedure, and we will formally define the parameters that we consider.

Formula-Based Judgment Aggregation An *agenda* is a finite, nonempty set Φ of formulas that does not contain any doubly-negated formulas and that is closed under complementation. Moreover, if $\Phi = \{\varphi_1, \dots, \varphi_n, \neg\varphi_1, \dots, \neg\varphi_n\}$ is an agenda, then we let $[\Phi] = \{\varphi_1, \dots, \varphi_n\}$ denote the *pre-agenda* associated to the agenda Φ . A *judgment set* J for an agenda Φ is a subset $J \subseteq \Phi$. We call a judgment set J *complete* if $\varphi \in J$ or $\neg\varphi \in J$ for all $\varphi \in \Phi$; and we call it *consistent* if there exists an assignment that makes all formulas in J true. Intuitively, the consistent and complete judgment sets are the opinions that individuals and the group can have.

We associate with each agenda Φ an integrity constraint Γ , that can be used to further restrict the set of feasible opinions. Such an integrity constraint consists of a single propositional formula. We say that a judgment set J is Γ -consistent if there exists a truth assignment that simultaneously makes all formulas in J and Γ true. Let $\mathcal{J}(\Phi, \Gamma)$ denote the set of all complete and Γ -consistent subsets of Φ . We say that finite sequences $\mathbf{J} \in \mathcal{J}(\Phi, \Gamma)^+$ of complete and Γ -consistent judgment sets are *profiles*, and where convenient we equate a profile $\mathbf{J} = (J_1, \dots, J_p)$ with the (multi)set $\{J_1, \dots, J_p\}$.

A *judgment aggregation procedure* (or *rule*) for the agenda Φ and the integrity constraint Γ , is a function F that takes as input a profile $\mathbf{J} \in \mathcal{J}(\Phi, \Gamma)^+$, and that produces a non-empty set of non-empty judgment sets. We call a judgment aggregation procedure F *resolute* if for any profile \mathbf{J} it returns a singleton, i.e., $|F(\mathbf{J})| = 1$; otherwise, we call F *irresolute*. An example of a resolute judgment aggregation procedure is the *strict majority rule* Majority, where $\text{Majority}(\mathbf{J}) = \{J^*\}$ and where $\varphi \in J^*$ if and only if φ occurs in the strict majority of judgment sets in \mathbf{J} , for all $\varphi \in \Phi$ (in case of a tie between φ and $\neg\varphi$, for $\varphi \in [\Phi]$, we arbitrarily let $\varphi \in J^*$). We call a judgment aggregation procedure F *complete* and Γ -consistent, if J is complete and Γ -consistent, respectively, for every $\mathbf{J} \in \mathcal{J}(\Phi, \Gamma)^+$ and every $J \in F(\mathbf{J})$. The procedure Majority is not consistent. Consider the agenda Φ with $[\Phi] = \{p, q, p \rightarrow q\}$, and the profile $\mathbf{J} = (J_1, J_2, J_3)$, where $J_1 = \{p, q, (p \rightarrow q)\}$, $J_2 = \{p, \neg q, \neg(p \rightarrow q)\}$, and $J_3 = \{\neg p, \neg q, (p \rightarrow q)\}$. The unique outcome $\{p, \neg q, (p \rightarrow q)\}$ in $\text{Majority}(\mathbf{J})$ is inconsistent.

The *Kemeny aggregation procedure* is based on a notion of distance. This distance is based on the Hamming distance $d(J, J') = |\{\varphi \in [\Phi] : \varphi \in (J \setminus J') \cup (J' \setminus J)\}|$ between two complete judgment sets J, J' . Intuitively, the Hamming distance $d(J, J')$ counts the number of issues on which two judgment sets disagree. Let J be a single Γ -consistent and complete judgment set, and let $(J_1, \dots, J_p) = \mathbf{J} \in \mathcal{J}(\Phi, \Gamma)^+$ be a profile. We define the distance between J and \mathbf{J} to be $d(J, \mathbf{J}) = \sum_{i \in [p]} d(J, J_i)$. Then, we let the outcome $\text{Kemeny}_{\Phi, \Gamma}(\mathbf{J})$ of the Kemeny rule be the set of those $J^* \in \mathcal{J}(\Phi, \Gamma)$ for which there is no $J \in \mathcal{J}(\Phi, \Gamma)$ such that $d(J, \mathbf{J}) < d(J^*, \mathbf{J})$. (If Φ and Γ are clear from the context, we often write $\text{Kemeny}(\mathbf{J})$ to denote $\text{Kemeny}_{\Phi, \Gamma}(\mathbf{J})$.) Intuitively, the Kemeny rule selects those complete and Γ -consistent judgment sets that minimize the cumulative Hamming distance to the judgment sets in the profile. The Kemeny rule is irresolute, complete and Γ -consistent.

We formalize the problem of computing an outcome of the Kemeny rule—in the formula-based judgment aggregation framework—with the following decision problem OUTCOME -

$(\text{KEMENY})^{\text{fb}}$. Any algorithm that solves $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ can be used to construct some $J^* \in \text{Kemeny}(\mathbf{J})$, with polynomial overhead, by iteratively calling the algorithm and adding formulas to the set L . Moreover, multiple outcomes J_1^*, J_2^*, \dots can be constructed by adding previously found outcomes as the sets L_i .

$\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$

Instance: An agenda Φ with an integrity constraint Γ , a profile $\mathbf{J} \in \mathcal{J}(\Phi, \Gamma)^+$ and subsets $L, L_1, \dots, L_u \subseteq \Phi$ of the agenda, with $u \geq 0$.

Question: Is there a judgment set $J^* \in \text{Kemeny}(\mathbf{J})$ such that $L \subseteq J^*$ and $L_i \not\subseteq J^*$ for each $i \in [u]$?

The parameters that we consider for the problem $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ are defined straightforwardly. For an instance $(\Phi, \Gamma, \mathbf{J}, L, L_1, \dots, L_u)$ of $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ with $\mathbf{J} = (J_1, \dots, J_p)$, we let $n = |[\Phi]|$, $m = \max\{|\varphi| : \varphi \in [\Phi]\}$, $c = |\Gamma|$, $p = |\mathbf{J}|$, and $h = \max\{d(J_i, J_{i'}) : 1 \leq i < i' \leq p\}$.

Constraint-Based Judgment Aggregation Let $\mathcal{I} = \{x_1, \dots, x_n\}$ be a finite set of *issues*. Intuitively, these issues are the topics about which the individuals want to combine their judgments. A truth assignment $\alpha : \mathcal{I} \rightarrow \{0, 1\}$ is called a *ballot*, and represents an opinion that individuals and the group can have. We will also denote ballots α by a binary vector $(b_1, \dots, b_n) \in \{0, 1\}^n$, where $b_i = \alpha(x_i)$ for each $i \in [n]$. Moreover, we say that $(p_1, \dots, p_n) \in \{0, 1, \star\}^n$ is a *partial ballot*, and that (p_1, \dots, p_n) agrees with a ballot (b_1, \dots, b_n) if $p_i = b_i$ whenever $p_i \neq \star$, for all $i \in [n]$. As in the case for formula-based judgment aggregation, we introduce an integrity constraint Γ , that can be used to restrict the set of feasible opinions (for both the individuals and the group). The integrity constraint Γ is a propositional formula on the variables x_1, \dots, x_n . We define the set $\mathcal{R}(\mathcal{I}, \Gamma)$ of *rational ballots* to be the ballots (for \mathcal{I}) that satisfy the integrity constraint Γ . Rational ballots in the constraint-based judgment aggregation framework correspond to complete and Γ -consistent judgment sets in the formula-based judgment aggregation framework. We say that finite sequences $\mathbf{r} \in \mathcal{R}(\mathcal{I}, \Gamma)^+$ of rational ballots are *profiles*, and where convenient we equate a profile $\mathbf{r} = (r_1, \dots, r_p)$ with the (multi)set $\{r_1, \dots, r_p\}$.

A *judgment aggregation procedure* (or *rule*), for the set \mathcal{I} of issues and the integrity constraint Γ , is a function F that takes as input a profile $\mathbf{r} \in \mathcal{R}(\mathcal{I}, \Gamma)^+$, and that produces a non-empty set of ballots. We call a judgment aggregation procedure F *rational* (or *consistent*), if r is rational for every $\mathbf{r} \in \mathcal{R}(\mathcal{I}, \Gamma)^+$ and every $r \in F(\mathbf{r})$.

As an example of a judgment aggregation procedure we consider the *strict majority rule* Majority, where $\text{Majority}(\mathbf{r}) = \{(b_1, \dots, b_n)\}$ and where each b_i agrees with the majority of the i -th bits in the ballots in \mathbf{r} (in case of a tie, we arbitrarily let $b_i = 1$). To see that Majority is not rational, consider the set $\mathcal{I} = \{x_1, x_2, x_3\}$ of issues, the integrity constraint $\Gamma = x_3 \leftrightarrow (x_1 \rightarrow x_2)$, and the profile $\mathbf{r} = (r_1, r_2, r_3)$, where $r_1 = (1, 1, 1)$, $r_2 = (1, 0, 0)$, and $r_3 = (0, 0, 1)$. The unique outcome $(1, 0, 1)$ in $\text{Majority}(\mathbf{r})$ is not rational.

The *Kemeny aggregation procedure* is defined for the constraint-based judgment aggregation framework as follows. Similarly to the case for formula-based judgment aggregation, the Kemeny rule is based on the Hamming distance $d(r, r') = |\{i \in [n] : b_i \neq b'_i\}|$, between two rational ballots $r = (b_1, \dots, b_n)$ and $r' = (b'_1, \dots, b'_n)$ for the set \mathcal{I} of issues and the integrity constraint Γ . Let r be a single

ballot, and let $(r_1, \dots, r_p) = \mathbf{r} \in \mathcal{R}(\mathcal{I}, \Gamma)^+$ be a profile. We define the distance between r and \mathbf{r} to be $d(r, \mathbf{r}) = \sum_{i \in [p]} d(r, r_i)$. Then, we let the outcome $\text{Kemeny}_{\mathcal{I}, \Gamma}(\mathbf{r})$ of the Kemeny rule be the set of those ballots $r^* \in \mathcal{R}(\mathcal{I}, \Gamma)$ for which there is no $r \in \mathcal{R}(\mathcal{I}, \Gamma)$ such that $d(r, \mathbf{r}) < d(r^*, \mathbf{r})$. (If \mathcal{I} and Γ are clear from the context, we often write $\text{Kemeny}(\mathbf{r})$ to denote $\text{Kemeny}_{\mathcal{I}, \Gamma}(\mathbf{r})$.) The Kemeny rule is irresolute and rational.

We formalize the problem of computing an outcome of the Kemeny rule—in the constraint-based judgment aggregation framework—with the following decision problem $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$. Similarly to algorithms for $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$, algorithms that solve $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ can be used to construct multiple outcomes.

$\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$

Instance: A set \mathcal{I} of issues with an integrity constraint Γ , a profile $\mathbf{r} \in \mathcal{R}(\mathcal{I}, \Gamma)^+$ and partial ballots l, l_1, \dots, l_u (for \mathcal{I}), with $u \geq 0$.

Question: Is there a ballot $r^* \in \text{Kemeny}(\mathbf{r})$ such that l agrees with r^* and each l_i does not agree with r^* ?

We define the parameters that we consider for $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ as follows. For an instance $(\mathcal{I}, \Gamma, \mathbf{r}, l, l_1, \dots, l_u)$ of $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ with $\mathbf{r} = (r_1, \dots, r_p)$, we let $n = |\mathcal{I}|$, $c = |\Gamma|$, $p = |\mathbf{r}|$, and $h = \max\{d(r_i, r_{i'}) : 1 \leq i < i' \leq p\}$. We remark that the parameter m does not make sense in the constraint-based framework, as issues are not represented by a logic formula. When needed, the parameter m for $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ is defined by letting $m = 1$.

4 Complexity Results

In this section, we develop the parameterized complexity results for the different parameterized variants of $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ and $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ that we consider.

4.1 Upper Bounds for the Formula-Based Framework

We begin with showing upper bounds for $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$. When parameterized either (i) by both h and p or (ii) by n , the problem is in $\text{FPT}^{\text{NP}}[\text{few}]$.

Proposition 1. $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ parameterized by h and p is in $\text{FPT}^{\text{NP}}[\text{few}]$.

Proof. The main idea behind this proof is that with these parameters, we can derive a suitable upper bound on the minimum distance of any complete and Γ -consistent judgment set to the profile \mathbf{J} , such that the usual binary search algorithm with access to an NP oracle only needs to make $O(\log h + \log p)$ many oracle queries.

We describe an algorithm A that solves $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ with the required number of oracle queries. Let $(\Phi, \Gamma, \mathbf{J}, L, L_1, \dots, L_u)$ be an instance. The algorithm needs to determine the minimum distance $d(J, \mathbf{J})$ for any complete and Γ -consistent judgment set J to the profile \mathbf{J} . Let d^* denote this minimum distance. An upper bound on d^* is given by $h(p-1)$. This upper bound can be derived as follows. Take an arbitrary $J \in \mathbf{J}$. Clearly $d(J, J) = 0$, and for every $J' \in \mathbf{J}$ with $J \neq J'$ we know that $d(J, J') \leq h$. Therefore, $d(J, \mathbf{J}) \leq h(p-1)$. Since $J \in \mathbf{J}$, we know that J is complete and Γ -consistent. Therefore, the minimum

distance of any complete and Γ -consistent judgment set to the profile \mathbf{J} is at most $h(p-1)$.

The algorithm A firstly computes d^* . Since $d^* \leq h(p-1)$, with binary search this can be done using at most $\lceil \log h(p-1) \rceil = O(\log h + \log p)$ many queries to an oracle—the oracle decides for a given value d_0 whether there exists a complete and Γ -consistent judgment set J with $d(J, \mathbf{J}) \leq d_0$. Then, with a single additional oracle query, the algorithm A determines whether there exists a complete and Γ -consistent judgment set J^* with $d(J^*, \mathbf{J}) = d^*$, $L \subseteq J^*$, and $L_j \not\subseteq J^*$ for each $j \in [u]$. \square

When parameterized by the number n of formulas in the pre-agenda, the number of possible judgment sets is bounded by a function of the parameter. This allows the problem to be solved in fixed-parameter tractable time, using a single query to an NP oracle for each judgment set to determine whether it is Γ -consistent.

Proposition 2. $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ parameterized by n is in $\text{FPT}^{\text{NP}}[\text{few}]$.

Proof. The main idea behind this proof is that the number of possible judgment sets is bounded by the parameter, that is, there are only 2^n many possible complete judgment sets. We describe an algorithm A that solves the problem in fixed-parameter tractable time by querying an NP oracle at most 2^n many times. Let $(\Phi, \Gamma, \mathbf{J}, L, L_1, \dots, L_u)$ be an instance. Firstly, the algorithm A enumerates all possible complete judgment sets $J_1, \dots, J_{2^n} \subseteq \Phi$. Then, for each such set J_i , the algorithm uses the NP oracle to determine whether J_i is Γ -consistent. Each judgment set J_i that is not Γ -consistent, is discarded. This can be done straightforwardly using 2^n many calls to the NP oracle—one for each set J_i . (The number of oracle calls that are needed can be improved to $O(n)$ by using binary search on the number of Γ -consistent sets J_i .)

Then, for each of the remaining (Γ -consistent) judgment sets J_i , the algorithm A computes the cumulative Hamming distance $d(J_i, \mathbf{J})$ to the profile \mathbf{J} . This can be done in polynomial time. Then, those J_i for which this distance is not minimal—that is, those J_i for which there exists some $J_{i'}$ such that $d(J_{i'}, \mathbf{J}) < d(J_i, \mathbf{J})$ —are discarded as well. The remaining judgment sets J_i then are exactly those complete and Γ -consistent judgment sets with a minimum distance to the profile \mathbf{J} .

Finally, the algorithm goes over each of these remaining sets J_i , and checks whether $L \subseteq J_i$ and $L_j \not\subseteq J_i$ for all $j \in [u]$. This can clearly be done in polynomial time. If this check succeeds for some J_i , the algorithm A accepts the input, and otherwise, the algorithm rejects the input. \square

When additionally parameterizing by c and m , Γ -consistency of the judgment sets can be decided in fixed-parameter tractable time, and thus the whole problem becomes fixed-parameter tractable.

Proposition 3. $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ parameterized by c , n and m is fixed-parameter tractable.

Proof. We describe an fpt-algorithm A that solves the problem. Let $(\Phi, \Gamma, \mathbf{J}, L, L_1, \dots, L_u)$ be an instance. The algorithm A works exactly in the same way as the algorithm in the proof of Proposition 2. The only difference is that in order to check whether a given judgment set J_i is Γ -consistent, it does not need to make an oracle query. Determining whether a given judgment set J_i is Γ -consistent can be done in a brute-force fashion (e.g., using truth tables) in time $2^{c+nm} \cdot |J_i|$, since there are at most $c + nm$ propositional variables involved. Therefore, the algorithm runs in fixed-parameter tractable time. \square

4.2 Lower Bounds for the Formula-Based Framework

Next, we turn to parameterized hardness results for the problem $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$. We begin with showing that the problem is $\text{para-}\Theta_2^{\text{p}}$ -hard even when parameterized by c , h and m . We will use the following lemma, whose straightforward proof we omit.

Lemma 4. *Let φ be a propositional formula on the variables x_1, \dots, x_n . In polynomial time we can construct a propositional formula φ' with $\text{Var}(\varphi') \supseteq \text{Var}(\varphi) \cup \{z_1, \dots, z_n\}$ such that for every truth assignment $\alpha : \text{Var}(\varphi) \rightarrow \{0, 1\}$ it holds that (1) $\varphi[\alpha]$ is true if and only if $\varphi'[\alpha]$ is satisfiable, and (2) if α sets exactly i variables to true, then $\varphi'[\alpha] \models z_i$.*

Proposition 5. $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ parameterized by c and h is $\text{para-}\Theta_2^{\text{p}}$ -hard.

Proof. We show that $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ is Θ_2^{p} -hard already for a constant value of the parameters, by giving a reduction from MAX-MODEL . Let (φ, w) be an instance of MAX-MODEL with $\text{Var}(\varphi) = \{x_1, \dots, x_n\}$ and $w = x_1$. Without loss of generality, we may assume that there is a model α of φ that sets at least two variables x_i to true. By Lemma 4, we can construct a suitable formula $\varphi' = c_1 \wedge \dots \wedge c_b$ with additional variables z_1, \dots, z_n that represent a lower bound on the number of variables among x_1, \dots, x_n that are true in models of φ .

We construct the agenda Φ by letting $[\Phi] = \{z_w, z_{\neg w}, z_1, \dots, z_n\} \cup \{y_{w,i}, y_{\neg w,i} : i \in [n+1]\} \cup \{y_{i,j} : i \in [n], j \in [i]\} \cup \{\chi, \chi'\}$, where $z_w, z_{\neg w}$ and all $y_{w,i}, y_{\neg w,i}, y_{i,j}$ are fresh variables. We let $Y = \{y_{w,i}, y_{\neg w,i} : i \in [n+1]\} \cup \{y_{i,j} : i \in [n], j \in [i]\}$. Moreover, we let χ be such that $\chi \equiv \neg((\bigvee Y \wedge \bigvee([\Phi] \setminus Y)) \vee ((z_w \leftrightarrow w \leftrightarrow \neg z_{\neg w}) \wedge \varphi'))$, and we define χ' such that $\chi' \equiv \chi$ (that is, we let χ' be a syntactic variant of χ).

Then, we construct the profile \mathbf{J} as follows. We let $\mathbf{J} = \{J_{w,i}, J_{\neg w,i} : i \in [n+1]\} \cup \{J_{i,j} : i \in [n], j \in [i]\}$. Each of the judgment sets in the profile includes exactly two formulas in $[\Phi]$. Consequently, the maximum Hamming distance between any two judgment sets in the profile is 4. For each $i \in [n+1]$, we let $\{y_{w,i}, z_w\} \subseteq J_{w,i}$ and $\{y_{\neg w,i}, z_{\neg w}\} \subseteq J_{\neg w,i}$. Moreover, for each $i \in [n]$ and each $j \in [i]$, we let $\{y_{i,j}, z_i\} \subseteq J_{i,j}$. It is straightforward to verify that each $J \in \mathbf{J}$ is consistent. Finally, we let $L = \{z_w\}$, $\Gamma = \top$, and $u = 0$.

In other words, all formulas in $[\Phi]$ are excluded in a majority of the judgment sets in the profile \mathbf{J} . However, some formulas in $[\Phi]$ are included in more judgment sets in the profile than others. The formulas z_w and $z_{\neg w}$ are both included in $n+1$ sets. Each formula z_i (for $i \in [n]$) is included in exactly i sets. All formulas in Y are included in exactly one set. Finally, the formulas χ and χ' are included in none of the sets. Intuitively, the formulas that are included in more judgment sets in the profile are cheaper to include in any candidate outcome J^* .

The complete judgment set that minimizes the cumulative Hamming distance to the profile \mathbf{J} is the set $J_0 = \{\neg\psi : \psi \in [\Phi]\}$ that includes no formulas in $[\Phi]$. However, this set is inconsistent, which is straightforward to verify using the definition of χ . It can be made consistent by adding two formulas ψ_1, ψ_2 from $[\Phi]$ (and removing their complements). The choice of ψ_1, ψ_2 that leads to a consistent judgment set with minimum distance to the profile is by letting $\psi_1 \in \{z_w, z_{\neg w}\}$ and letting $\psi_2 = z_\ell$, where ℓ is the maximum number of variables among x_1, \dots, x_n set to true in any

model of φ . Moreover, whenever $\psi_1 = z_w$, the resulting judgment set is consistent if and only if there is a model of φ that sets ℓ variables among x_1, \dots, x_n to true, including the variable w . From this, we directly know that $(\varphi, w) \in \text{MAX-MODEL}$ if and only if $(\Phi, \Gamma, \mathbf{J}, L) \in \text{OUTCOME}(\text{KEMENY})^{\text{fb}}$. This concludes our $\text{para-}\Theta_2^{\text{p}}$ -hardness proof. \square

This hardness result can straightforwardly be extended to the case where all formulas in the agenda are of constant size, by using the well-known Tseitin transformation [43], leading to the following corollary.

Corollary 6. $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ parameterized by c , h and m is $\text{para-}\Theta_2^{\text{p}}$ -hard.

Proof. We can modify the proof of Proposition 5 as follows. We replace the formula $\neg\chi$ (and its syntactic variant $\neg\chi'$) by a 3CNF formula that has the same effect. By using the standard Tseitin transformation [43], we can transform $\neg\chi$ into a 3CNF formula ψ such that for each truth assignment $\alpha : \text{Var}(\neg\chi) \rightarrow \{0, 1\}$ it holds that $\neg\chi[\alpha]$ is true if and only if $\psi[\alpha]$ is satisfiable. Moreover, we can do this in such a way that the variables in $\text{Var}(\psi) \setminus \text{Var}(\neg\chi)$ are fresh variables. Similarly, we transform $\neg\chi'$ into a 3CNF formula ψ' . Let $\psi = c_1 \wedge \dots \wedge c_b$ and $\psi' = c'_1 \wedge \dots \wedge c'_b$ (we can straightforwardly ensure that ψ and ψ' have the same number of clauses). Then, similarly to the proof of Proposition 5, we let $[\Phi] = \{z_w, z_{\neg w}, z_1, \dots, z_n\} \cup \{y_{w,i}, y_{\neg w,i} : i \in [n+1]\} \cup \{y_{i,j} : i \in [n], j \in [i]\} \cup \{c_i, c'_i : i \in [b]\}$. That is, instead of adding χ and χ' to the agenda, we add the clauses of ψ and ψ' as separate formulas to the agenda.

In the proof of Proposition 5, we had that $\neg\chi, \neg\chi' \in J$ for all judgment sets $J \in \mathbf{J}$. Instead, we now ensure that for all $J \in \mathbf{J}$, we have $c_i, c'_i \in J$ for all $i \in [b]$. From this, it follows that the set $\text{Kemeny}(\mathbf{J})$ of outcomes is in one-to-one correspondence with the set of outcomes in the proof of Proposition 5. Moreover, the maximum Hamming distance between any two judgment sets in the profile \mathbf{J} is 4. \square

The problem is also $\text{para-}\Theta_2^{\text{p}}$ -hard when parameterized by c , m and p .

Proposition 7. $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ parameterized by c , m and p is $\text{para-}\Theta_2^{\text{p}}$ -hard.

Proof. We firstly show $\text{para-}\Theta_2^{\text{p}}$ -hardness for the problem parameterized by c and p , by giving a reduction from MAX-MODEL that uses constant values of c and p . This reduction can be seen as a modification of the Θ_2^{p} -hardness proof for $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ given by Endriss and De Haan [18, Proposition 7 and Corollary 8].

Let (φ, w) be an instance of MAX-MODEL . We may assume without loss of generality that φ is satisfiable by some truth assignment that sets at least one variable in $\text{Var}(\varphi)$ to true. We construct an instance $(\Phi, \Gamma, \mathbf{J}, L)$ of $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ as follows. Take an integer b such that $b > \frac{3}{2}|\text{Var}(\varphi)|$, e.g., $b = 3|\text{Var}(\varphi)| + 1$. Let $[\Phi] = \text{Var}(\varphi) \cup \{z_{i,j} : i \in [b], j \in [3]\} \cup \{\varphi'_i : i \in [b]\}$, where each of the formulas φ'_i is a syntactic variant of the following formula φ' . We define $\varphi' = (\bigvee_{j \in [3]} \bigwedge_{i \in [b]} z_{i,j}) \vee \varphi$. Intuitively, the formula φ' is true either if (i) all variables $z_{i,j}$ are set to true for some $j \in [3]$, or if (ii) φ is satisfied. Then we let $\mathbf{J} = \{J_1, J_2, J_3\}$, where for each $j \in [3]$, we let J_j contain the formulas $z_{i,j}$ for all $i \in [b]$, all formulas in $\text{Var}(\varphi)$, all the formulas φ'_i , and no other formulas from $[\Phi]$. (For each $\varphi \in [\Phi]$, if $\varphi \notin J_j$, we let $\neg\varphi \in J_j$.)

Clearly, the judgment sets J_1 , J_2 and J_3 are all complete and consistent. Moreover, we let $\Gamma = \top$, and $L = \{w\}$. It is straightforward to verify that the parameters c and p have constant values.

We now argue that there is some $J^* \in \text{Kemeny}(\mathbf{J})$ with $L \subseteq J^*$ if and only if $(\varphi, w) \in \text{MAX-MODEL}$. To see this, we first observe that the only complete and consistent judgment sets J for which it holds that $d(J, \mathbf{J}) < d(J_j, \mathbf{J})$ (for any $j \in [3]$) must satisfy that $J \models \varphi$. Moreover, among those judgment sets J for which $J \models \varphi$, the judgment sets that minimize the distance to the profile \mathbf{J} satisfy that $z_{i,j} \notin J$ for all $i \in [b]$ and all $j \in [3]$, and $\varphi'_i \in J$ for all $i \in [b]$. Using these observations, we directly get that there is some $J^* \in \text{Kemeny}(\mathbf{J})$ with $L \subseteq J^*$ if and only if there is a model of φ that sets a maximal number of variables in $\text{Var}(\varphi)$ to true and that sets the variable w to true.

Then, to show that the problem is also $\text{para-}\Theta_2^{\text{P}}$ -hard when parameterized by c , m and p , we can modify the above reduction in a way that is entirely similar to the proof of Corollary 6, replacing the formulas φ'_i by the clauses of 3CNF formulas that have the same effect on the consistency of judgment sets as the formulas φ'_i . \square

For all parameterizations that do not include all of the parameters c , n and m , the problem $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ is $\text{FPT}^{\text{NP}}[\text{few}]$ -hard. We begin with the case where c can be unbounded; this proof can be extended straightforwardly to the other two cases.

Proposition 8. $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ parameterized by h , n , m and p is $\text{FPT}^{\text{NP}}[\text{few}]$ -hard.

Proof. We show $\text{FPT}^{\text{NP}}[\text{few}]$ -hardness by giving an fpt-reduction from LOCAL-MAX-MODEL. (This reduction from LOCAL-MAX-MODEL is very similar to the reduction from MAX-MODEL used in the proof of Proposition 7.) Let (φ, X, w) be an instance of LOCAL-MAX-MODEL, with $X = \{x_1, \dots, x_k\}$. We construct an instance $(\Phi, \Gamma, \mathbf{J}, L)$ as follows. Take an integer b such that $b > \frac{3}{2}|X|$, e.g., let $b = 3|X| + 1$. We let $[\Phi] = X \cup \{z_{i,j} : i \in [b], j \in [3]\}$. Moreover, we let $\Gamma = \varphi' = (\bigvee_{j \in [3]} \bigwedge_{i \in [b]} z_{i,j}) \vee \varphi$. Intuitively, the formula Γ is true either if (i) all variables $z_{i,j}$ are set to true for some $j \in [3]$, or if (ii) φ is satisfied. Then we let $\mathbf{J} = \{J_1, J_2, J_3\}$, where for each $j \in [3]$, we let J_j contain the formulas $z_{i,j}$ for all $i \in [b]$, and all formulas in X , and no other formulas in $[\Phi]$. (For each $\varphi \in [\Phi]$, if $\varphi \notin J_j$, we let $\neg\varphi \in J_j$.) Clearly, the judgment sets J_1 , J_2 and J_3 are all complete and Γ -consistent. Finally, we let $L = \{w\}$. It is easy to verify that $h = 2b = 6k + 2$ and $n = 3b + k = 10k + 3$, where $k = |X|$, and that m and p are constant. Therefore, all parameter values are bounded by a function of the original parameter k .

We now argue that there is some $J^* \in \text{Kemeny}(\mathbf{J})$ with $L \subseteq J^*$ if and only if $(\varphi, X, w) \in \text{LOCAL-MAX-MODEL}$. The argument for this conclusion is similar to the argument used in the proof of Proposition 7. We first observe that the only complete and consistent judgment sets J for which it holds that $d(J, \mathbf{J}) < d(J_j, \mathbf{J})$ (for any $j \in [3]$) must satisfy that $J \models \varphi$. Moreover, among those judgment sets J for which $J \models \varphi$, the judgment sets that minimize the distance to the profile \mathbf{J} satisfy that $z_{i,j} \notin J$ for all $i \in [b]$ and all $j \in [3]$. Using these observations, we directly get that there is some $J^* \in \text{Kemeny}(\mathbf{J})$ with $L \subseteq J^*$ if and only if there is a model of φ that sets a maximal number of variables in X to true and that sets the variable w to true. \square

Proposition 9. $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ parameterized by c , h , n and p is $\text{FPT}^{\text{NP}}[\text{few}]$ -hard.

Proof (sketch). We can modify the reduction from LOCAL-MAX-MODEL used in the proof of Proposition 8. Instead of using the formula φ' as the integrity constraint Γ , we let $\Gamma = \top$, and we enforce φ' to be true by adding b syntactic variants $\varphi'_1, \dots, \varphi'_b$ of it to the agenda and to the judgment sets. \square

Proposition 10. $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ parameterized by c , h , m and p is $\text{FPT}^{\text{NP}}[\text{few}]$ -hard.

Proof (sketch). We modify the reduction from LOCAL-MAX-MODEL given in the proof of Proposition 9. We use the same trick that we used in the proof of Corollary 6—that is, we use the standard Tseitin transformation [43] to transform each of the formulas φ'_i into a 3CNF formula φ''_i that forces φ'_i to be true. Then, we replace the formulas φ'_i by the clauses of φ''_i in the agenda and the judgment sets. \square

4.3 Upper Bounds for the Constraint-Based Framework

We now turn to showing upper bounds for $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$. When parameterized by the number n of issues, the number of possible ballots is bounded by a function of the parameter. This allows the problem to be solved in fixed-parameter tractable time.

Proposition 11. $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ parameterized by n is fixed-parameter tractable.

Proof. The main idea behind this proof is that the number of possible ballots is bounded by the parameter, that is, there are only 2^n many possible (rational) ballots. We describe an algorithm A that solves the problem in fixed-parameter tractable time. Let $(\mathcal{I}, \Gamma, \mathbf{r}, l, l_1, \dots, l_u)$ be an instance. Firstly, the algorithm A enumerates all possible ballots $r_1, \dots, r_{2^n} \in \{0, 1\}^n$. Then, for each such ballot r_i , the algorithm determines whether r_i is rational, by checking whether $\Gamma[r_i]$ is true. This can be done in polynomial time. Each irrational ballot is discarded.

Then, for each of the remaining (rational) ballots r_i , the algorithm A computes the cumulative Hamming distance $d(r_i, \mathbf{r})$ to the profile \mathbf{r} . This can also be done in polynomial time. Then, those r_i for which this distance is not minimal—that is, those r_i for which there exists some $r_{i'}$ such that $d(r_{i'}, \mathbf{r}) < d(r_i, \mathbf{r})$ —are discarded as well. The remaining ballots r_i then are exactly those rational ballots with a minimum distance to the profile \mathbf{r} .

Finally, the algorithm goes over each of these remaining ballots r_i , and checks whether l agrees with r_i and whether for all $j \in [u]$, l_j does not agree with r_i . If this check succeeds for some r_i , the algorithm A accepts the input, and otherwise, the algorithm rejects the input. \square

Since the size c of the integrity constraint is an upper bound on the number of issues that play a non-trivial role in the problem, this fixed-parameter tractability result easily extends to the parameter c .

Proposition 12. $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ parameterized by c is fixed-parameter tractable.

Proof. Since $|\Gamma| = c$, we know that the number of propositional variables in Γ is also bounded by the parameter c . Take an instance $(\mathcal{I}, \Gamma, \mathbf{r}, l, l_1, \dots, l_u)$. Then, let $\mathcal{I}' = \text{Var}(\Gamma) \subseteq \mathcal{I}$ be the subset of issues that are mentioned in the integrity constraint Γ . We know that any outcome $r^* \in \text{Kemeny}(\mathbf{r})$ agrees with the majority of ballots in \mathbf{r} on every issue in $\mathcal{I} \setminus \mathcal{I}'$ (in case of a tie, either choice

works). Therefore, all that remains is to determine whether there are suitable choices for the issues in \mathcal{I} (to obtain some $r^* \in \text{Kemeny}(\mathbf{r})$ that agrees with l and does not agree with l_j for all $j \in [u]$). By Proposition 11, we know that this is fixed-parameter tractable in $|\mathcal{I}'|$. Since $|\mathcal{I}'| \leq c$, we get fixed-parameter tractability also for $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ parameterized by c . \square

Bounding the maximum Hamming distance h between any two ballots in the profile gives us membership in XP.

Proposition 13. $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ parameterized by h is in XP.

Proof. Let $(\mathcal{I}, \Gamma, \mathbf{r}, l, l_1, \dots, l_u)$ be an instance, with $\mathbf{r} = (r_1, \dots, r_p)$. We describe an algorithm to solve the problem in time $O(p \cdot n^h \cdot n^c)$, for some constant c . The main idea behind this algorithm is the fact that each ballot whose Hamming distance to every ballot in the profile is more than h is irrelevant.

Take a ballot r such that $d(r, r_i) > h$ for each $i \in [p]$. We show that there exists a rational ballot r' with $d(r', \mathbf{r}) < d(r, \mathbf{r})$. Take any ballot in the profile, e.g., $r' = r_1$. Clearly, r' is rational. Since $d(r, r_i) > h$ for each $i \in [p]$, we know that $d(r, \mathbf{r}) > hp$. On the other hand, for r' we know that $d(r', r_i) \leq h$ for each $i \in [p]$ (and $d(r', r_1) = 0$), so $d(r', \mathbf{r}) \leq h(p-1)$. Therefore, $d(r', \mathbf{r}) < d(r, \mathbf{r})$.

We thus know that every rational ballot with minimum distance to the profile lies at Hamming distance at most h to some ballot r_i in the profile \mathbf{r} . The algorithm works as follows. It firstly enumerates all ballots with Hamming distance at most h to some $r_i \in \mathbf{r}$. This can be done in time $O(p \cdot n^h)$. Then, similarly to the algorithm in the proof of Proposition 11, it discards those ballots that are not rational, and subsequently discards those ballots that do not have minimum distance to the profile. Finally, it iterates over all remaining rational ballots with minimum distance to determine whether there is one among them that agrees with l and disagrees with each l_j . \square

4.4 Lower Bounds for the Constraint-Based Framework

Finally, we show parameterized hardness results for $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$. When parameterized by both h and p , the problem is W[SAT]-hard.

Proposition 14. $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ parameterized by h and p is W[SAT]-hard.

Proof. We give an fpt-reduction from the W[SAT]-complete problem MONOTONE-WSAT. Let (φ, k) be an instance of MONOTONE-WSAT. We construct an instance $(\mathcal{I}, \Gamma, \mathbf{r}, l)$ of $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ as follows. We let $\mathcal{I} = \text{Var}(\varphi) \cup \{z\} \cup \{y_{i,j} : i \in [3], j \in [3k+3]\}$. Moreover, we let $\Gamma = (z \wedge \varphi) \vee (\neg z \wedge \bigvee_{i \in [3]} (\bigwedge_{j \in [3k+3]} y_{i,j}))$. We define $\mathbf{r} = (r_1, r_2, r_3)$ as follows. For each r_i , we let $r_i(w) = 0$ for all $w \in \{z\} \cup \text{Var}(\varphi)$. Moreover, for each r_i and each $y_{\ell,j}$, we let $r_i(y_{\ell,j}) = 1$ if and only if $\ell = i$. It is readily verified that r_1, r_2 and r_3 are all rational. Finally, we let l be the partial assignment for which $l(z) = 1$, and that is undefined on all remaining variables. This completes our construction. Clearly, $p = 3$. Moreover, $h = 6k + 6$.

By construction of Γ , the only ballots that are rational—and that can have a smaller distance to the profile \mathbf{r} than the ballots r_1, r_2 and r_3 —are those ballots r^* that satisfy $(z \wedge \varphi)$. The ballots r_1, r_2 and r_3 have distance $4(3k+3) = 12k+12$ to the profile \mathbf{r} . Any ballot r^* that satisfies $(z \wedge \varphi)$ minimizes its distance to \mathbf{r} by setting

all variables $y_{i,j}$ to false. Any such ballot r^* has distance $3(3k+3) + 3(w+1) = 9k+3w+12$ to the profile \mathbf{r} , where w is the number of variables among $\text{Var}(\varphi)$ that it sets to true. Therefore, the distance of such a ballot r^* to the profile \mathbf{r} is smaller than (or equal to) the distance of r_1, r_2 and r_3 to \mathbf{r} if and only if $9k+3w+12 \leq 12k+12$, which is the case if and only if $w \leq k$. From this we can conclude that there is some $r^* \in \text{Kemeny}(\mathbf{r})$ that agrees with l if and only if $(\varphi, k) \in \text{MONOTONE-WSAT}$. \square

Finally, the proof of Proposition 7 can be modified to work also for the problem $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ parameterized by p , showing para- Θ_2^p -hardness for this case.

Proposition 15. $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ parameterized by p is para- Θ_2^p -hard.

Proof. We modify the Θ_2^p -hardness reduction used in the proof of Proposition 7 to work also for the case of $\text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ for a constant value of the parameter p . Instead of adding the formulas φ'_i to the agenda Φ , as done in the proof of Proposition 7, we let $\Gamma = \varphi'$. The remaining formulas in the agenda Φ were all propositional variables, and thus we can transform the instance $(\Phi, \Gamma, \mathbf{J}, L)$ that we constructed for $\text{OUTCOME}(\text{KEMENY})^{\text{fb}}$ into an instance $(\mathcal{I}, \Gamma, \mathbf{r}, l)$, where \mathbf{r} and l are constructed entirely analogously to \mathbf{J} and L . Clearly, $p = 3$. Moreover, by a similar argument to the one that is used in the proof of Proposition 7, we get that $(\mathcal{I}, \Gamma, \mathbf{r}, l) \in \text{OUTCOME}(\text{KEMENY})^{\text{cb}}$ if and only if $(\varphi, w) \in \text{MAX-MODEL}$. \square

5 Conclusion

We gave the first parameterized complexity results for the fundamental problem of computing outcomes of judgment aggregation procedures. We studied parameterized variants of this problem for the Kemeny rule, for all combinations of the parameters c, h, n, m and p . Moreover, we performed this parameterized complexity analysis for two formal frameworks for judgment aggregation: formula-based and constraint-based judgment aggregation.

Interestingly, for many combinations of parameters, the complexity of the problem differs between the two frameworks—which is in contrast with the fact that the problem has the same complexity in both frameworks when viewed from a classical complexity point of view. This reflects the ability of the framework of parameterized complexity to more accurately indicate what aspects of the problem input contribute to the complexity of the problem. The two judgment aggregation frameworks distribute the aspects of the problem differently over various parts of the problem input.

Future work includes extending the parameterized complexity investigation for computing outcomes of the Kemeny rule to different parameters. For instance, in particular for the constraint-based judgment aggregation framework, restricting the maximum degree of variables in the integrity constraint might lead to more positive parameterized complexity results. Other natural parameters that could be considered are width measures that capture the amount of structure in the logic formulas in the problem input. Moreover, it would be interesting to perform a similar parameterized complexity analysis for other judgment aggregation procedures.

ACKNOWLEDGEMENTS

This work has been supported by the Austrian Science Fund (FWF), project P26200 (Parameterized Compilation).

REFERENCES

- [1] Karl A. Abrahamson, Rodney G. Downey, and Michael R. Fellows, ‘Fixed-parameter tractability and completeness. IV. On completeness for W[P] and PSPACE analogues’, *Annals of Pure and Applied Logic*, **73**(3), 235–276, (1995).
- [2] Sanjeev Arora and Boaz Barak, *Computational Complexity – A Modern Approach*, Cambridge University Press, 2009.
- [3] John Bartholdi, Craig A Tovey, and Michael A Trick, ‘Voting schemes for which it can be difficult to tell who won the election’, *Social Choice and Welfare*, **6**(2), 157–165, (1989).
- [4] Dorothea Baumeister, Gábor Erdélyi, and Jörg Rothe, ‘How hard is it to bribe the judges? A study of the complexity of bribery in judgment aggregation’, in *Proceedings of the Second International Conference on Algorithmic Decision Theory (ADT 2011)*, Piscataway, NJ, USA, October 26–28, 2011, volume 6992 of *Lecture Notes in Computer Science*, pp. 1–15. Springer Verlag, (2011).
- [5] Nadja Betzler, Robert Brederick, Jiehua Chen, and Rolf Niedermeier, ‘Studies in computational aspects of voting – a parameterized complexity perspective’, in *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, eds., Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, volume 7370 of *Lecture Notes in Computer Science*, pp. 318–363. Springer Verlag, (2012).
- [6] Nadja Betzler, Michael R Fellows, Jiong Guo, Rolf Niedermeier, and Frances A Rosamond, ‘Fixed-parameter algorithms for kemeny rankings’, *Theoretical Computer Science*, **410**(45), 4554–4570, (2009).
- [7] Zhi-Zhong Chen and Seinosuke Toda, ‘The complexity of selecting maximal solutions’, *Information and Computation*, **119**, 231–239, (June 1995).
- [8] Yann Chevaleyre, Ulle Endriss, Jérôme Lang, and Nicolas Maudet, ‘A short introduction to computational social choice’, in *Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2007)*, pp. 51–69. Springer Verlag, (2007).
- [9] Vincent Conitzer and Toby Walsh, ‘Barriers to manipulation in voting’, in *Handbook of Computational Social Choice*, eds., Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel Procaccia, Cambridge University Press, Cambridge, (2015).
- [10] Marek Cygan, Fedor V Fomin, Lukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh, *Parameterized Algorithms*, Springer, 2015.
- [11] Franz Dietrich, ‘Scoring rules for judgment aggregation’, *Social Choice and Welfare*, **42**(4), 873–911, (2014).
- [12] Franz Dietrich and Christian List, ‘Judgment aggregation without full rationality’, *Social Choice and Welfare*, **31**(1), 15–39, (2008).
- [13] Rodney G. Downey and Michael R. Fellows, *Parameterized Complexity*, Monographs in Computer Science, Springer Verlag, New York, 1999.
- [14] Rodney G. Downey and Michael R. Fellows, *Fundamentals of Parameterized Complexity*, Texts in Computer Science, Springer Verlag, 2013.
- [15] Ulle Endriss, ‘Judgment aggregation’, in *Handbook of Computational Social Choice*, eds., Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel Procaccia, Cambridge University Press, Cambridge, (2016).
- [16] Ulle Endriss, Umberto Grandi, Ronald de Haan, and Jérôme Lang, ‘Succinctness of languages for judgment aggregation’, in *Proceedings of the Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2016)*, Cape Town, South Africa, April 25–29, 2016. AAAI Press, (2016).
- [17] Ulle Endriss, Umberto Grandi, and Daniele Porello, ‘Complexity of judgment aggregation’, *J. Artif. Intell. Res.*, **45**, 481–514, (2012).
- [18] Ulle Endriss and Ronald de Haan, ‘Complexity of the winner determination problem in judgment aggregation: Kemeny, Slater, Tideman, Young’, in *Proceedings of AAMAS 2015, the 14th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS/ACM, (2015).
- [19] Ulle Endriss, Ronald de Haan, and Stefan Szeider, ‘Parameterized complexity results for agenda safety in judgment aggregation’, in *Proceedings of the 5th International Workshop on Computational Social Choice (COMSOC-2014)*. Carnegie Mellon University, (June 2014).
- [20] Ulle Endriss, Ronald de Haan, and Stefan Szeider, ‘Parameterized complexity results for agenda safety in judgment aggregation’, in *Proceedings of AAMAS 2015, the 14th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS/ACM, (2015).
- [21] Piotr Faliszewski, Edith Hemaspaandra, and Lane A Hemaspaandra, ‘Using complexity to protect elections’, *Communications of the ACM*, **53**(11), 74–82, (2010).
- [22] Jörg Flum and Martin Grohe, ‘Describing parameterized complexity classes’, *Information and Computation*, **187**(2), 291–319, (2003).
- [23] Jörg Flum and Martin Grohe, *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*, Springer Verlag, Berlin, 2006.
- [24] Umberto Grandi, *Binary Aggregation with Integrity Constraints*, Ph.D. dissertation, University of Amsterdam, 2012.
- [25] Umberto Grandi and Ulle Endriss, ‘Lifting integrity constraints in binary aggregation’, *Artificial Intelligence*, **199–200**, 45–66, (2013).
- [26] Davide Grossi and Gabriella Pigozzi, *Judgment Aggregation: A Primer*, Morgan & Claypool Publishers, 2014.
- [27] Ronald de Haan and Stefan Szeider, ‘Fixed-parameter tractable reductions to SAT’, in *Proceedings of the 17th International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2014) Vienna, Austria, July 14–17, 2014*, eds., Uwe Egly and Carsten Sinz, volume 8561 of *Lecture Notes in Computer Science*, pp. 85–102. Springer Verlag, (2014).
- [28] Ronald de Haan and Stefan Szeider, ‘The parameterized complexity of reasoning problems beyond NP’, in *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2014)*, Vienna, Austria, July 20–24, 2014, eds., Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter. AAAI Press, (2014).
- [29] Edith Hemaspaandra, Holger Spakowski, and Jörg Vogel, ‘The complexity of Kemeny elections’, *Theoretical Computer Science*, **349**(3), 382–391, (2005).
- [30] Jon Kleinberg and Éva Tardos, *Algorithm design*, Pearson Education India, 2006.
- [31] Mark W. Krentel, ‘The complexity of optimization problems’, *J. of Computer and System Sciences*, **36**(3), 490–509, (1988).
- [32] Jérôme Lang and Marija Slavkovic, ‘How hard is it to compute majority-preserving judgment aggregation rules?’, in *21st European Conference on Artificial Intelligence (ECAI 2014)*. IOS Press, (2014).
- [33] Claudia Lindner and Jörg Rothe, ‘Fixed-parameter tractability and parameterized complexity, applied to problems from computational social choice’, *Mathematical Programming Glossary*, (2008).
- [34] Christian List and Philip Pettit, ‘Aggregating sets of judgments: An impossibility result’, *Economics and Philosophy*, **18**(1), 89–110, (2002).
- [35] Christian List and Clemens Puppe, ‘Judgment aggregation: A survey’, in *Handbook of Rational and Social Choice*, Oxford University Press, (2009).
- [36] Michael K. Miller and Daniel Osherson, ‘Methods for distance-based judgment aggregation’, *Social Choice and Welfare*, **32**(4), 575–601, (2009).
- [37] Klaus Nehring, Marcus Pivato, and Clemens Puppe. Condorcet admissibility: indeterminacy and path-dependence under majority voting on interconnected decisions. MPRA, 2011. <http://mpra.ub.uni-muenchen.de/32434/>.
- [38] Rolf Niedermeier, *Invitation to Fixed-Parameter Algorithms*, Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, Oxford, 2006.
- [39] Venkatesh Raman and Saket Saurabh, ‘Improved fixed parameter tractable algorithms for two “edge” problems: MAXCUT and MAXDAG’, *Information Processing Letters*, **104**(2), 65–72, (2007).
- [40] Jörg Rothe, *Economics and Computation*, Springer, 2016.
- [41] Jörg Rothe, Holger Spakowski, and Jörg Vogel, ‘Exact complexity of the winner problem for young elections’, *Theory Comput. Syst.*, **36**(4), 375–386, (2003).
- [42] *Automation of reasoning. Classical Papers on Computer Science 1967–1970*, eds., Jörg Siekmann and Graham Wrightson, volume 2, Springer Verlag, 1983.
- [43] G. S. Tseitin, ‘Complexity of a derivation in the propositional calculus’, *Zap. Nauchn. Sem. Leningrad Otd. Mat. Inst. Akad. Nauk SSSR*, **8**, 23–41, (1968). English translation reprinted in [42].
- [44] Klaus W. Wagner, ‘Bounded query classes’, *SIAM J. Comput.*, **19**(5), 833–846, (1990).

AGM-Style Revision of Beliefs and Intentions

Marc van Zee and Dragan Doder
Computer Science and Communication
University of Luxembourg

marcvanzee@gmail.com, dragan.doder@uni.lu

Abstract. We introduce a logic for temporal beliefs and intentions based on Shoham’s database perspective and we formalize his coherence conditions on beliefs and intentions. In order to do this we separate strong beliefs from weak beliefs. Strong beliefs are independent from intentions, while weak beliefs are obtained by adding intentions to strong beliefs and everything that follows from that. We provide AGM-style postulates for the revision of strong beliefs and intentions: strong belief revision may trigger intention revision, but intention revision may only trigger revision of weak beliefs. After revision, the strong beliefs are coherent with the intentions. We show in a representation theorem that a revision operator satisfying our postulates can be represented by a pre-order on interpretations of the beliefs, together with a selection function for the intentions.

1 Introduction

Recently there has been an increase in articles studying the dynamics of intentions in logic [7, 10, 15, 26, 14, 9]. Most of those papers take as a starting point the logical frameworks derived from Cohen and Levesque [6], which in turn formalize Bratman’s [4] planning theory of intention. In this paper, we take a different starting point, and study the revision of intentions from a *database perspective* [23]. The database perspective consists of a planner, a belief database and an intention database. Shoham [24] describes it as “(...) a generalization of the AGM scheme for belief revision, (...). In the AGM framework, the intelligent database is responsible for storing the planner’s beliefs and ensuring their consistency. In the enriched framework, there are two databases, one for beliefs and one for intentions, which are responsible for maintaining not only their individual consistency but also their mutual consistency.” (p.48) Shoham further developed these ideas with Jacob Banks, one of his PhD students, and behavioral economist Dan Ariely in the intelligent calendar application Timeful, which attracted over \$6.8 million in funding and was acquired by Google in 2015¹, who aim to integrate it into their Calendar applications. As Shoham [24] says himself: “The point of the story is there is a direct link between the original journal paper and the ultimate success of the company.” (p.47) Thus, it seems clear that his philosophical proposal has led to some success on the practical side. In this paper, we investigate whether his proposal can lead to interesting theoretical insights as well. More specifically, the aim of this paper is to develop a suitable formal theory for the belief and intention databases in the database perspective, to formalize the coherence conditions that Shoham puts on the databases, and to study belief and intention revision for this theory. Following Shoham’s proposal, our

methodology is to generalize AGM revision [1] for temporal beliefs and intentions and to prove a representation theorem.

In the area of intention revision and reconsideration, Grant *et al.* [9] combine intention revision with AGM-like postulates. There have also been a number of contributions applying AGM-style revision to action logics [22, 11, 20, 21, 3]. However, these proposals only characterize revision using a set of postulates, without proving representation theorems. There are also approaches that focus on the semantical level by postulating revision on a Kripke model [2].

In this paper, we first review two recent formalisations based on Shoham’s database perspective, namely the IPS (Icard-Pacuit-Shoham) framework [10] and our own previous work PAL (Parameterized-time Action Logic) [30], and we discuss the shortcomings of these logics. We extend PAL in order to formalize Shoham’s database perspective. An overview of our approach is displayed in Figure 1. We separate strong beliefs from weak beliefs. Strong beliefs are beliefs that are independent of the intentions of the agent, while weak beliefs are those beliefs that are obtained by adding the consequences of all intended actions to the strong beliefs, and everything that follows from that. A planner (outside of our agent) may add beliefs and intentions. The belief database consists of strong beliefs, and can only be updated by a strong belief formula. Revision of strong beliefs affects weak beliefs but, according to Shoham’s database perspective, they may only remove intentions. Intention revision may in turn trigger revision of the weak beliefs. The main result is to characterize this revision process correctly through postulates and to prove a representation theorem.

The structure of this paper is as follows. Section 2 is preliminary and introduces Shoham’s database perspective, the IPS framework, and PAL. In Section 3 we formalize strong and weak beliefs, and we formalize Shoham’s coherence condition in Section 4. In Section 5 we study revision of beliefs and intentions, and in Section 6 we discuss related work.

2 Preliminaries: The Database Perspective

We start by introducing our running example that we will use frequently throughout the paper.

Example 1 (Running Example) *An agent located in Luxembourg is considering to attend the IJCAI conference in New York City, NY (USA) in July 2016 and the ECAI conference in The Hague (the Netherlands) in August 2016. Although it would like to attend both events, there is insufficient budget available for traveling. The agent thus believes that it is possible to attend IJCAI at time 0 (July 2016) and that it is possible to attend ECAI at time 1 (August 2016), but*

¹ <http://venturebeat.com/2015/05/04/google-acquires-scheduling-app-timeful-and-plans-to-integrate-it-into-google-apps/>

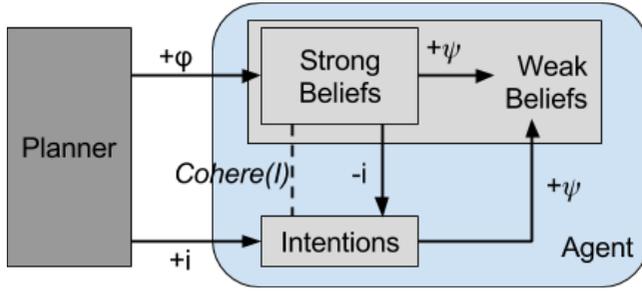


Figure 1: Our formalization of the database perspective. Strong beliefs are denoted by ϕ , weak beliefs by ψ , and intentions by i . An arrow indicates that a component can add (+) or remove (-) a formula from another component. The dashed line represents a coherence condition on strong beliefs and intentions. Weak beliefs are obtained from strong beliefs by adding the consequences of all intentions. The planner is considered a black box and can update strong beliefs and intentions. Strong beliefs may update intentions, but intentions may only update weak beliefs.

also believes that it is impossible to attend both conferences. If the agent decides to attend IJCAI, then it would like to combine this with a visit to a colleague in New York at time 2 (September 2016).

2.1 Shoham's Database Perspective

Shoham's database perspective [23] contains a planner (e.g., a STRIPS-like planner) that is itself engaged in some form of practical reasoning. In the course of planning, it may add actions to be taken at various times in the future to an intention database and add observations to a belief database. The intentions are *future-directed intentions* of the form (a, t) , meaning that action a will be executed at time t .² The beliefs are also time-indexed, and are of the form p_t , meaning that p is true at time t . Shoham treats the planner as a "black box": It provides the databases with input but its internal workings are unknown. Shoham proposes informal revision procedures for beliefs and intentions based on the following coherence conditions:

1. Beliefs must be internally consistent.
2. Intentions must be internally consistent.
 - (a) At most one action can be intended for a given time moment.
 - (b) If two intended actions immediately follow one another, the earlier cannot have postconditions that are inconsistent with the preconditions of the latter.
3. Intentions must be consistent with beliefs.
 - (a) If you intend to take an action you cannot believe that its preconditions do not hold.³
 - (b) If you intend to take an action, you believe that its postconditions hold.

Note that requirement 3a and 3b describe an asymmetry between pre- and postconditions: The postconditions are believed to be true after an intended action, but the preconditions may not. Therefore, we

² The notion of intention here is rather restrictive and important characteristics of intentions are missing. See the conclusion for a discussion.

³ Shoham notes here that "it is important to distinguish between the time of belief, and the time to which the belief refers. When an intention to act at time t_2 is added at time t_1 (with $t_1 < t_2$), then at time t_1 it is believed that right after the action is taken at t_2 its postconditions will hold." [23, p.7]

might think of the requirements as one of "optimistic" beliefs. According to Shoham [23]: "It is a good fit with how planners operate. Adopting an optimistic stance, they feel free to add intended actions so long as they are consistent with current beliefs." (p. 7)

Shoham then "sketches" informal revision procedures, in which belief revision may trigger intention revision and visa versa, potentially leading to a long cascade of changes. Facts that are believed because they are postconditions of currently held intentions are annotated as such, because "if the intention is withdrawn then the belief in the postcondition can be eliminated as well." (p. 8)

2.2 Icard et al. (IPS)

Icard et al. [10] develop a "formal semantical model to capture action, belief and intention, based on the 'database perspective'" (p.1). They assume a set of atomic sentences $\text{Prop} = \{p, q, r, \dots\}$ and deterministic primitive actions $\text{Act} = \{a, b, c, \dots\}$. Entries in the belief database are represented by a language generated from:

$$\phi := p_t \mid \text{pre}(a)_t \mid \text{post}(a)_t \mid \text{Do}(a)_t \mid \Box\phi \mid \phi \wedge \phi \mid \neg\phi$$

with $p \in \text{Prop}, a \in \text{Act}$, and $t \in \mathbb{Z}$. p_t means that p is true at time t , $\text{Do}(a)_t$ means that the agent does action a at time t , and $\text{pre}(a)_t$ and $\text{post}(a)_t$ represent respectively the precondition and postcondition of action a at time t .

Icard et al. use a semantics of *appropriate paths*. They define $P = \mathcal{P}(\text{Prop} \cup \{\text{pre}(a), \text{post}(a) : a \in \text{Act}\})$, and a path $\pi : \mathbb{Z} \rightarrow (P \times \text{Act})$ as a mapping from a time point to a set of proposition-like formulas true at that time (denoted $\pi(t)_1$) and the next action a on the path (denoted $\pi(t)_2$). They define an equivalence relation $\pi \sim_t \pi'$, which means that π and π' represent the same situation up to t . Using this, they propose a notion of appropriateness:

Definition 1 (Appropriate Set of Paths) A set of paths Π is appropriate iff for all $\pi \in \Pi$:

- If $\pi(t)_2 = a$, then $\text{post}(a) \in \pi(t+1)_1$,
- If $\text{pre}(a) \in \pi(t)_1$, then there exists $\pi' \sim_t \pi$ s.t. $\pi'(t)_2 = a$.

The truth definition \models_{Π} is defined relative to an appropriate set of paths Π , and the modality is defined as follows:

$$\pi, t \models_{\Pi} \Box\phi, \text{ iff for all } \pi' \in \Pi, \text{ if } \pi \sim_t \pi' \text{ then } \pi', t \models \phi.$$

A model for a formula is an appropriate set of paths. They introduce an intention database $I = \{(a, t), \dots\}$ as a set of action-time pairs (a, t) and put the following coherence condition on their logic:

$$\text{Cohere}^*(I) = \diamond \bigwedge_{(a,t) \in I} \text{pre}(a)_t.$$

This captures the intuition that an agent considers it possible to carry out all intended actions. They state that a pair (Π, I) is coherent if and only if there exists a path in Π in which $\text{Cohere}^*(I)$ is true. IPS distinguishes *intention-contingent*, or weak, beliefs from *non-contingent*, or strong, beliefs. Contingent beliefs B^I are obtained from a belief-intention database (B, I) as follows:

$$B^I = \text{Cl}(B \cup \{\text{Do}(a)_t : (a, t) \in I\}).$$

In order to switch from belief bases to an appropriate set of paths, Icard et al. introduce the functions ρ and β : "Given a set of formulas B , we can consider the set of paths on which all formulas of B hold at time 0, denoted $\rho(B)$. Conversely, given a set of paths Π , we let

$\beta(\Pi)$ be defined as the set of formulas valid at 0 in all paths in Π .⁷ (p.3)

The first issue with IPS is that the definition of non-contingent beliefs is problematic for coherence. We demonstrate this using our running example.

Example 2 (Non-contingent beliefs in IPS) *Suppose that the agent believes it has to go to the ECAI conference at time 1 because the event will be held in The Hague, which is very close to Luxembourg (where the agent is located) and thus is cheaper for traveling. However, the agent has the intention to go to IJCAI. We can formalize this in the IPS framework as a belief-intention base (B, I) with $\{post(ECAI)_2, \neg\Diamond(do(IJCAI)_0 \wedge post(ECAI)_2)\} \subseteq B$ and $(IJCAI, 0) \in I$. The coherence condition holds because $Cohere^*(I) = \Diamond(pre(IJCAI)_0)$ is consistent with B . However, the contingent beliefs B^I are inconsistent since $\neg\Diamond(do(IJCAI)_0 \wedge post(ECAI)_2) \wedge do(IJCAI)_0$ implies $\neg post(ECAI)_2$, but this is inconsistent with the initial belief $post(ECAI)_2$.*

The example above shows that contingent beliefs may become inconsistent if the agent has intentions that are conflicting with what it believes non-contingently. While the agent believes that it will not go to IJCAI, it still has the intention to do so. This is coherent according to $Cohere^*(I)$, since the agent doesn't believe that it is impossible to go to IJCAI. However, adding the consequences of the intention to go to IJCAI results in a conflict with the fact that the agent believes it will attend ECAI. The source of the problem is that non-contingent beliefs in the IPS framework such as $post(ECAI)_2$ are actually not non-contingent, because they are contingent on the actions that occur in the path (namely to attend ECAI in time 1). We solve this problem in Section 3 by requiring that strong beliefs are always prefixed by a modal operator. In this way, the beliefs are about possibility and necessity, but they are not beliefs about a specific future.

The second issue with IPS is that the definition of ρ is circular, and as a result it does not seem to be possible to apply it to all formulas of their logic. The IPS definition of ρ is $\rho(B) = \{\pi \mid \pi \models_{\Pi} B\}$. That is, the set of paths for a belief base B is those paths in which all formulas of B are true, given an appropriate set of paths Π . However, the function ρ should construct the appropriate set of paths. Therefore, the definition of ρ is circular. It seems that the function ρ only works for belief bases containing no modalities. We omit details for space constraints, but the construction of the canonical model in the proof of their representation theorem uses the function ρ to switch from a belief base to a set of paths (see Proof Sketch in the Appendix of Icard *et al.* [10]). Therefore, the representation theorem does not hold for all formulas of the logic, since it is not possible to apply the function ρ to all beliefs.

Summarizing, we recognize two shortcomings of the IPS framework as a formal basis for the database perspective: the definition of contingent beliefs is problematic, and the representation theorem does not hold for belief bases containing modalities.

2.3 Parameterized-time Action Logic

In our previous work, we develop Parameterized-time Action Logic (PAL) [29, 30, 31] as an alternative to IPS. PAL differs syntactically from IPS in that the \Box -modality is indexed by a time-point, and semantically in that it uses a standard branching time logic semantics.

Definition 2 (PAL Language [31]) *Let $Act = \{a, b, c, \dots\}$ be a finite set of deterministic primitive actions, and $Prop = \{p, q, r, \dots\} \cup$*

$\{pre(a), post(a) \mid a \in Act\}$ be a finite set of propositions.⁴ The sets $Prop$ and Act are disjoint. The language \mathcal{L}_{PAL} is inductively defined by the following BNF grammar:

$$\varphi ::= \chi_t \mid do(a)_t \mid \Box_t \varphi \mid \varphi \wedge \varphi \mid \neg \varphi$$

with $\chi \in Prop, a \in Act$, and $t \in \mathbb{N}$. We abbreviate $\neg\Box_t\neg$ with \Diamond_t , and define $\perp \equiv p_0 \wedge \neg p_0$ and $\top \equiv \neg \perp$.

PAL uses a CTL*-like [19] tree semantics consisting of a tree $T = (S, R, v, act)$ where S is a set of states, R is an accessibility relation that is serial, linearly ordered in the past and connected, $v : S \rightarrow 2^{Prop}$ is a valuation function from states to sets of propositions, and $act : R \rightarrow Act$ is a function from accessibility relations to actions, such that actions are deterministic, i.e. if $act((s, s')) = act((s, s''))$, then $s' = s''$.

Given a tree $T = (S, R, v, act)$, a path $\pi = (s_0, s_1, \dots)$ in T is a sequence of states such that $(s_t, s_{t+1}) \in R$. The formula π_t refers to the t 'th state of the path π , so $v(\pi_t)$ and $act((\pi_t, \pi_{t+1}))$ refer respectively to the propositions true and the next action on path π at time t . For readability, $act((\pi_t, \pi_{t+1}))$ is abbreviated with $act(\pi_t)$.

Similarly to IPS, we define an equivalence relation on paths: two paths π and π' are equivalent up to time t , denoted $\pi \sim_t \pi'$, if and only if they contain the same states up to and including time t .

Definition 3 (Model [31]) *A model is a pair (T, π) where $T = (S, R, v, act)$ is a tree and π is a path in T , and for all $\pi \in T$ the following conditions hold:*

1. *If $act(\pi_t) = a$, then $post(a) \in v(\pi_{t+1})$.*
2. *If $pre(a) \in v(\pi_t)$, then there is some π' in T with $\pi \sim_t \pi'$ and $act(\pi'_t) = a$.*

Definition 4 (Truth definitions) *Let (T, π) be a model with $T = (S, R, v, act)$:*

- $T, \pi \models \chi_t$ iff $\chi \in v(\pi_t)$ with $\chi \in Prop$*
- $T, \pi \models do(a)_t$ iff $act(\pi_t) = a$*
- $T, \pi \models \neg\varphi$ iff $T, \pi \not\models \varphi$*
- $T, \pi \models \varphi \wedge \varphi'$ iff $T, \pi \models \varphi$ and $T, \pi \models \varphi'$*
- $T, \pi \models \Box_t \varphi$ iff for all π' in T : if $\pi' \sim_t \pi$, then $T, \pi' \models \varphi$*

We axiomatize PAL and show that it is sound and strongly complete, i.e. $T \vdash \varphi$ iff $T \models \varphi$.

Furthermore, we characterize AGM belief revision in this logic by bounding all inputs and output of the revision process up to some time t . Using these constraints, we are able to represent a belief set B as a propositional formula ψ such that $B = \{\varphi \mid \psi \vdash \varphi\}$ and we prove the Katsuno and Mendelzon (KM) [12] representation theorem.

We now formalize the beliefs of our agent in the running example in PAL.

Example 3 (PAL model) *A possible partial PAL model (T, π_2) of the beliefs of our conference planning agent is shown in Figure 2, where the thick path represents the actual path. In the actual path, the agent believes it attends IJCAI at time 0 and visit a colleague at time 1. It also considers it possible to do nothing at time 0 and attend ECAI at time 1 in an alternative path. However, it does not consider it to be possible to attend both conferences.⁵ Some formulas that are true in Figure 2 are:*

$$T, \pi_3 \models \neg do(visit)_1$$

⁴ Throughout this paper we denote atomic propositions with χ .

⁵ Note that $pre(nop) \equiv post(nop) \equiv \top$. They have been omitted from the figure for readability.

$$\begin{aligned}
T, \pi_2 &\models \text{post}(\text{visit})_2 \rightarrow \neg \text{do}(\text{ECAI})_1 \\
T, \pi_1 &\models \diamond_0(\text{do}(\text{IJCAI})_0 \wedge \neg \text{do}(\text{visit})_1) \\
T, \pi_2 &\models \diamond_0 \square_1 \text{do}(\text{ECAI})_1 \\
T, \pi_1 &\models \neg \diamond_0(\text{do}(\text{IJCAI})_0 \wedge \text{do}(\text{ECAI})_1)
\end{aligned}$$

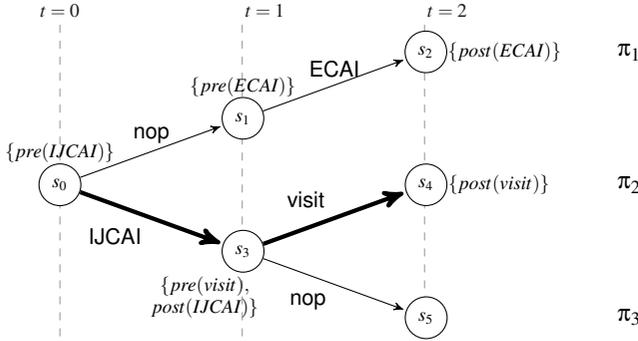


Figure 2: Example PAL Model (T, π_2) from $t = 0$ to $t = 2$.

In our previous work, we only consider the revision of PAL formulas. However, in the current approach we distinguish between strong and weak beliefs, and define the belief database as a set of strong beliefs (Figure 1). This means that our previous approach cannot be applied directly. We cannot apply it to strong beliefs, because it is not ensured that after revision we end up with strong beliefs again. We can also not apply it to the beliefs that result as consequence of intentions (weak beliefs), because revision of these beliefs should not affect strong beliefs. We explain the details of strong and weak beliefs in more detail in the next section.

3 Formalizing Strong and Weak Beliefs

Suppose the conference planning agent of our running example adopts the intention to attend IJCAI at time 0. When adopting this intentions, the agent will form new beliefs based on the success of this intentions, e.g., that it will be in New York at time 1. These further beliefs can be used in the course of further planning, for instance it may adopt the intention to visit a colleague in New York at time 1. If the agent then learns that its flight from Luxembourg to New York is canceled, it should drop the intention to attend IJCAI. Yet, by dropping this intention that was based on the now-dropped belief, other beliefs, including the belief that it will be in New York at time 1, should also be dropped, which may in turn force the intention to visit a colleague to be dropped. And so on. Thus, belief revision may trigger intention revision, which again can trigger belief revision, etc. However, intention revision should not change the “basic” beliefs of an agent. For instance, if the agent adopts the intention to attend IJCAI at time 0, and it does not annotate the beliefs based on this intention in any way, then the planner may believe that it is no longer necessary to attend IJCAI in order to be in New York at time 1, and thus decide to retract the intention. This is the so-called Little Nell paradox, and has been discussed extensively in the literature [16, 5, 27]. Shoham proposed as well to annotate postconditions of intentions in coherence condition 3b in order to separate them from ordinary beliefs (see Section 2.1).

In order to deal with these nuances, we separate strong and weak beliefs (this is the terminology used by van der Hoek *et al.* [27] as

well). The idea behind strong beliefs is that they represent the agent’s ideas about what is inevitable, no matter how it would act in the world. They thus represent the agent’s view on the current situation and the future within which it can plan its action. Formally, strong beliefs at some time t are formulas that start either with \diamond_t or \square_t .

Definition 5 (Strong Beliefs) *The set of all strong beliefs \mathbb{B}_t in time t for \mathcal{L}_{PAL} is inductively defined by the following BNF grammar:*

$$\varphi ::= \square_t \psi \mid \varphi \wedge \varphi \mid \neg \varphi,$$

where $\psi \in \mathcal{L}_{PAL}$ and $t \in \mathbb{N}$. A strong belief set B_t is a set of strong beliefs closed under consequence, i.e. $B_t = \text{Cn}(B')$ where $B' \subseteq \mathbb{B}_t$. In the remainder of this paper, we assume $t = 0$ and we simply write \mathbb{B} and B to abbreviate \mathbb{B}_0 and B_0 .

The weak beliefs $WB(B, I)$ are obtained from strong beliefs B by adding beliefs that are contingent on the intentions I . In other words, weak beliefs are all the strong beliefs and moreover the consequences of all intentions, and everything that follows from this.

Definition 6 (Weak Beliefs) *Given a pair (B, I) , the weak beliefs are defined as:*

$$WB(B, I) = \text{Cn}(B \cup \{\text{do}(a)_t \mid (a, t) \in I\}).$$

Let us formalize these notions in our running example.

Example 4 (Strong and weak beliefs) *Let (B, I) be such that the beliefs up to time 2 are represented by the model in Figure 2⁶, and let $I = \{(IJCAI, 0), (\text{visit}, 1)\}$. Some examples of strong beliefs are $\diamond_0 \text{do}(\text{IJCAI})_0$ (“it is possible at time 0 to attend IJCAI at time 0”), $\diamond_0 \text{do}(\text{ECAI})_1$ (“it is possible at time 0 to attend ECAI at time 1”), and $\neg \diamond_0(\text{post}(\text{ECAI})_2 \wedge \text{post}(\text{IJCAI})_1)$ (“It is not possible that the postconditions of attending ECAI and IJCAI are both true, respectively at time 2 and 1”). Some examples of weak beliefs are $\text{do}(\text{IJCAI})_1$, $\text{post}(\text{ijcai})_2$, and $\text{post}(\text{visit})_2 \rightarrow \text{do}(\text{IJCAI})_0$.*

The reader may already have noted that, semantically, strong beliefs are independent of the specific path on which they are true. For instance, returning to Example 3, the belief $\neg \diamond_0(\text{do}(\text{IJCAI})_0 \wedge \text{do}(\text{ECAI})_1)$ is true in path π_1 , but it is also true in all other paths of the model. This seems to indicate that the models of a set of strong beliefs are rather sets of trees instead of sets of tree-path pairs. It captures the intuition that strong beliefs are not dependent on a specific future, but are about possibility and necessity. In Section 5, we will make this property more precise and use it in order to characterize revision of strong beliefs.

4 Formalizing Shoham’s Coherence Conditions

In this section we formalize Shoham’s coherence condition in our framework. We first demonstrate that $\text{Cohere}^*(I)$ of IPS (Section 2.2) is too permissive because it allows models in which intentions are not jointly executable.

Example 5 (Coherence in IPS) *Suppose the agent of the running example has two intentions: $I = \{(IJCAI, 0), (\text{ECAI}, 1)\}$. Intuitively,*

⁶ In other words, all models are the same up to $t = 2$. This single model property simplifies the exposition of our framework and we assume this throughout the paper.

the agent's intentions do not cohere with its beliefs, because it believes it cannot execute them both due to insufficient budget. However, according to $\text{Cohere}^*(I)$ the agent is coherent because the preconditions of all intentions hold on some path (namely the actual path of Figure 2). This is because while the precondition to attend IJCAI is true at time 0, the agent only executes this action in a path different from the path in which it attends ECAI as well.

Thus, $\text{Cohere}^*(I)$ does not fulfill Shoham's coherence condition 2b (Section 2.1). Although the preconditions of the intended actions are true on a path, the intentions are not jointly executable because the postcondition of the first action is inconsistent with the precondition of the second. More specifically, the problem is that it is not possible to define the precondition of a set of actions in terms of preconditions of individual actions, because it cannot be ensured that all the intentions are fulfilled on the same path as well. Therefore, in order to formalize a coherence condition in PAL, we extend the language with preconditions of finite action sequences, which ensures that after executing the first action, the precondition for the remaining actions are still true. We modify the language, the definition of a model, the axiomatization, and we show that the new axiomatization is sound and strongly complete. We call the new logic PAL-P (Parameterized-time Action Logic with extended Preconditions).

Definition 7 (PAL-P Language) The language \mathcal{L} is obtained from \mathcal{L}_{PAL} by adding $\{\text{pre}(a, b, \dots)_t \mid \{a, b, \dots\} \subseteq \text{Act}, t \in \mathbb{N}\}$ to the set of propositions.

We also extend the definition of a model accordingly.

Definition 8 (PAL-P Model) A model is a pair (T, π) with $T = (S, R, v, \text{act})$ such that for all $\pi \in T$ the following holds:

1. If $\text{act}(\pi_t) = a$, then $\text{post}(a) \in v(\pi_{t+1})$,
2. If $\text{pre}(a) \in v(\pi_t)$, then there is some π' in T with $\pi \sim_t \pi'$ and $\text{act}(\pi'_t) = a$,
3. If $\text{pre}(\dots, a, b)_t \in v(\pi_t)$, then $\text{pre}(\dots, a)_t \in v(\pi_t)$,
4. If $\text{pre}(a, b, \dots)_t \in v(\pi_t)$, then there is some π' in T with $\pi \sim_t \pi'$, $\text{act}(\pi'_t) = a$, and $\text{pre}(b, \dots)_{t+1} \in v(\pi'_{t+1})$.

We refer to models of PAL-P with m_1, m_2, \dots , we refer to sets of models with M_1, M_2, \dots , and we refer to the set of all models with \mathbb{M} .

Definition 9 The logic PAL-P consists of the all the axiom schemas and rules of PAL [31] (Def. 7), and the following two:

$$\text{pre}(\dots, a, b)_t \rightarrow \text{pre}(\dots, a)_t \quad (A11)$$

$$(\text{pre}(a, b, \dots)_t \wedge \text{do}(a)_t) \rightarrow \text{pre}(b, \dots)_{t+1} \quad (A12)$$

The relation \vdash is defined in the usual way with the restriction that necessitation can be applied to theorems only.

Theorem 1 (Completeness Theorem) The logic PAL-P is sound and strongly complete, i.e. $T \vdash \varphi$ iff $T \models \varphi$.

Note that it is not directly possible in PAL-P to express preconditions for actions that do not occur directly after each other. In order to do so, we simply make a disjunction over all possible action combinations in the time points in between the actions. Thus, if for instance $\text{Act} = \{a, b\}$ and $I = \{(a, 1), (b, 3)\}$, then $\text{Cohere}(I) = \diamond_0 \bigvee_{x \in \text{Act}} \text{pre}(a, x, b)_1 = \diamond_0 (\text{pre}(a, a, b)_1 \vee \text{pre}(a, b, b)_1)$.⁷

⁷ Our construction of preconditions over action sequences may lead to a coherence condition involving a big disjunction. This is a drawback in terms of computational complexity. Alternatively, one may explicitly denote the time of each precondition, e.g. $\text{pre}(a, b)_{(t_1, t_2)}$. We chose the former since it is conceptually closer to the original syntax, but the latter can be implemented straightforwardly.

Definition 10 (Coherence) Given an intention database $I = \{(b_{t_1}, t_1), \dots, (b_{t_n}, t_n)\}$ with $t_1 < \dots < t_n$, let

$$\text{Cohere}(I) = \diamond_0 \bigvee_{\substack{a_k \in \text{Act}: k \notin \{t_1, \dots, t_n\} \\ a_k = b_k: k \in \{t_1, \dots, t_n\}}} \text{pre}(a_{t_1}, a_{t_1+1}, \dots, a_{t_n})_{t_1}.$$

For a given set of models M , we say that (M, I) is coherent iff there exists some $m \in M$ with $m \models \text{Cohere}(I)$. For a given agent $A = (B, I)$, we say that the A is coherent iff B is consistent with $\text{Cohere}(I)$, i.e., $B \not\vdash \neg \text{Cohere}(I)$.

Naturally, if a set of intentions is coherent with a set of strong beliefs, then its subset is coherent as well.

Lemma 1 if $I' \subseteq I$, then $\text{Cohere}(I) \vdash \text{Cohere}(I')$.

The following proposition was originally proposed by Icard *et al.*, and holds in our framework.

Proposition 1 If (B, I) is coherent, then $WB(B, I)$ is consistent.

Let us demonstrate the difference between $\text{Cohere}^*(I)$ of IPS and our $\text{Cohere}^*(I)$ using the running example. It turns out the above proposition does not hold for IPS.

Example 6 (Comparing the coherence conditions) Recall from Example 5 that the model of Figure 2 is coherent with intentions $I = \{(IJCAI, 0), (ECAI, 1)\}$ according to $\text{Cohere}^*(I)$ of the IPS framework. However, the weak beliefs of this agent are inconsistent because the agent believes it is impossible to execute both intentions. Thus, Proposition 1 does not hold here.

On the other hand, the model is not coherent according to $\text{Cohere}(I)$, because the agent does not have the possibility to jointly execute both intentions (i.e. the preconditions for both actions together is false). Thus, none of the models satisfy $\diamond_0 \text{pre}(IJCAI, ECAI)_0$, even though they satisfy $\diamond_0 \text{pre}(IJCAI)_0 \wedge \diamond_0 \text{pre}(ECAI)_1$. Thus, we see that in this case Proposition 1 holds.

We now discuss Shoham's coherence conditions. Condition 1 is a direct consequence of Proposition 1. Condition 2a follows from our notion of coherence: if $a \neq b$, $\{(a, t), (b, t)\} \subseteq I$, and B is some set of strong beliefs, then $WB(B, I) \vdash \text{do}(a)_t \wedge \text{do}(b)_t$. On the other hand, $\text{do}(a)_t \rightarrow \neg \text{do}(b)_t$ is an of PAL-P (Axiom A7, Def. 7 [30]), so the weak beliefs are inconsistent. By Proposition 1, B is not coherent with I . We show that condition 2b holds in the following proposition.

Proposition 2 (Coherence Condition 2b) If $\{(a, t), (b, t+1)\} \subseteq I$ and B is a set of strong beliefs such that (B, I) coherent, then $\{\text{pre}(b)_{t+1}, \text{post}(a)_{t+1}\}$ is consistent with B .

Condition 3a follows from our coherence condition: If (B, I) is coherent, i.e. $B \not\vdash \neg \text{Cohere}(I)$, then it follows by Lemma 1 that for all $(a, t) \in I$, $B \not\vdash \neg \diamond_0 \text{pre}(a)_t$ (since $\text{Cohere}(\{(a, t)\}) = \diamond_0 \text{pre}(a)_t$). Finally, condition 3b is formalized using our notion of weak beliefs and axiom A9 of PAL-P: $\text{do}(a)_t \rightarrow \text{post}(a)_t$ (Def. 7 [30]).

5 Belief and Intention Revision

In this section we formalize the revision procedures on an agent. That is, we formalize all the arrows from Figure 1. Recall that we distinguish between the revision of strong beliefs, the revision of intentions, and the revision of weak beliefs, which is a consequence of both. The revision of strong beliefs may trigger intention revision, while intention revision only triggers the revision of weak beliefs.

5.1 Revision Postulates

Following KM, we fix a way of representing a belief set B consisting of strong beliefs by a propositional formula ψ such that $B = \{\varphi \mid \psi \vdash \varphi\}$. Since intentions and beliefs that have been added by a planner are naturally bounded up to some time point t , we define a bounded revision function and we restrict the syntax and semantics of PAL-P up to a specific time point. As a consequence, it is then possible to obtain the single formula ψ for a set of strong beliefs B (Corollary 1). The difficulty of obtaining this result is that when revising a belief database that is bounded up to some time t with a strong belief, we have to ensure that the resulting belief database is also bounded up to t , and that it remains a strong belief. We first define some notation that we use in the rest of this paper.

Definition 11 An agent is a pair (ψ, I) consisting of a belief formula ψ , and an intention base I . \mathbb{A} denotes the set of all agents, \mathbb{B} denotes the set of all strong beliefs, \mathbb{I} denotes the set of all intentions, and \mathbb{D} denotes the set of all intention databases. We denote $\mathbb{A}, \mathbb{B}, \mathbb{I}$, and \mathbb{D} bounded up to t with respectively $\mathbb{A}^t, \mathbb{B}^t, \mathbb{I}^t$, and \mathbb{D}^t . However, if the restriction is clear from context, we may omit the superscript notation. We define ε as a special “empty” intention.

We now define a bounded revision function $*_t$ revising an agent (ψ, I) with a tuple (φ, i) consisting of a strong belief φ and an intention i , denoted $(\psi, I) *_t (\varphi, i)$, where t is the maximal time point occurring in ψ, I, φ , and i .

Definition 12 (Agent Revision Function) An Agent revision function $*_t : \mathbb{A} \times (\mathbb{B} \times \mathbb{I}) \rightarrow \mathbb{A}$ maps an agent, a strong belief formula, and an intention— all bounded up to t — to an agent bounded up to t such that if,

$$(\psi, I) *_t (\varphi, i) = (\psi', I'),$$

$$(\psi_2, I_2) *_t (\varphi_2, i_2) = (\psi'_2, I'_2),$$

then following postulates hold:

(P1) ψ' implies φ .

(P2) If $\psi \wedge \varphi$ is satisfiable, then $\psi' \equiv \psi \wedge \varphi$.

(P3) If φ is satisfiable, then ψ' is also satisfiable.

(P4) If $\psi \equiv \psi_2$ and $\varphi \equiv \varphi_2$ then $\psi' \equiv \psi'_2$.

(P5) If $\psi \equiv \psi_2$ and $\varphi_2 \equiv \varphi \wedge \varphi'$ then $\psi' \wedge \varphi'$ implies ψ'_2 .

(P6) If $\psi \equiv \psi_2$, $\varphi_2 \equiv \varphi \wedge \varphi'$, and $\psi' \wedge \varphi'$ is satisfiable, then ψ'_2 implies $\psi' \wedge \varphi'$.

(P7) (ψ', I') is coherent.

(P8) If $(\psi', \{i\})$ is coherent, then $i \in I'$.

(P9) If $(\psi', I \cup \{i\})$ is coherent, then $I \cup \{i\} \subseteq I'$.

(P10) $I' \subseteq I \cup \{i\}$.

(P11) If $I = I_2$, $i = i_2$, and $\psi' \equiv \psi'_2$, then $I' = I'_2$.

(P12) For all I'' with $I' \subset I'' \subseteq I \cup \{i\}$: (ψ', I'') is not coherent.

Postulates (P1)-(P6) are simply the KM postulates in our setting, which are equivalent to the AGM postulates [12]. They also state that the revision of strong beliefs does not depend on the intentions. Postulates (P7)-(P10) also appear in IPS. Postulate (P7) states that the outcome of a revision should be coherent. Postulate (P8) states that the new intention i take precedence over all other current intentions; if possible, it should be added, even if all current intentions have to be discarded. Postulate (P9) and (P10) together state that if it is possible to simply add the intention, then this is the only change that is made. Postulate (P11) states that if we revise with the same i but with a different belief, and we end up with the same belief in both cases, then we also end up with the same intentions. Finally, (P12) states that we do not discard intentions unnecessarily. This last postulate

is comparable to the *parsimony requirement* introduced by Grant *et al.* [9].

We now discuss our revision function in some more detail, starting with a simple example.

Example 7 (Adding an intention) Suppose we have an agent $A = (\psi, I)$ such that all models of the strong beliefs B are the same as the partial model in Figure 2 up to $t = 2$, and suppose that $I = \{(IJCAI, 0), (\text{visit}, 1)\}$. That is, that agent has the intention to attend IJCAI at time 0 and then visit a colleague at time 1. Suppose now that the agent changes its intention to attend ECAI at time 1. Formally: $(\psi, I) *_2 (\top, i) = (\psi, I')$ with $i = (ECAI, 1)$. Since $(\psi, \{i\})$ is coherent but (ψ, I') is not, from (P8) and (P9) we obtain $i \in I'$. Furthermore, from (P10) we have that $I' \subseteq \{(IJCAI, 0), (\text{visit}, 1), (ECAI, 1)\}$. Combining this gives $I' = \{(ECAI, 1)\}$ as the only coherent outcome. Thus, the agent no longer intends to attend IJCAI and to visit the colleague. Note that, although the strong beliefs didn't change after revising with the new intention, the weak beliefs did change. For example, $\text{post}(IJCAI)_1 \in \text{WB}(\psi, I) \setminus \text{WB}(\psi, I')$ and $\text{post}(ECAI)_2 \in \text{WB}(\psi, I') \setminus \text{WB}(\psi, I)$.

The revision function $*_t$ takes a tuple (φ, i) as input, and the postulates (P1)-(P7) ensure that revision of strong beliefs occurs prior to the revision of intentions. Therefore, it may seem plausible that revising with (φ, i) is the same as first revising with (φ, ε) and then with (\top, i) . In other words, the following postulate seems to follow:

$$\begin{aligned} &\text{If } (\psi, I) *_t (\varphi, i) = (\psi', I') \\ &\text{and } ((\psi, I) *_t (\varphi, \varepsilon)) *_t (\top, i) = (\psi'', I''), \quad (\text{P13}^*) \\ &\text{then } \psi' \equiv \psi'' \text{ and } I' = I''. \end{aligned}$$

However, this property does not follow from (P1)-(P12), and we show in the following example that adding the postulate would in fact conflict with the maximality postulate (P12).

Example 8 (Joint vs separate revision) Suppose some agent $A = (\psi, I)$ with beliefs up to $t = 2$ corresponding to the model on the left of Figure 3. The agent has the possible actions to go to the dentist (d) or to stay at work (w), and then go eating (e) or go to the movies (m). Before revision, the agent has intentions $I = \{(d, 0), (e, 1)\}$ (left image of Figure 3, intentions shown as bold lines). It then revises with the belief that it cannot go eating after going to the dentist (φ) and as a result with the intention to go to the movie at time 1 ($i = (m, 1)$). The resulting strong beliefs after revising with φ are shown on the right of Figure 3.

Let us first analyze joint revision. That is, $(\psi, I) *_2 (\varphi, i) = (\psi', I')$. Both $(\psi', \{(d, 0), (m, 1)\})$ and $(\psi', \{(m, 1)\})$ are coherent, so by the maximality postulate (P12), $I' = \{(d, 0), (m, 1)\}$. Hence, the agent intends to go to the dentist and go to the movie.

For separate revision, let $(\psi, I) *_2 (\varphi, \varepsilon) = (\psi', \bar{I})$ and $(\psi', \bar{I}) *_2 (\top, i) = (\psi', \bar{I}')$ (note that ψ' is the same as for joint revision, by (P4)). Now, since $(\psi', \{(d, 0)\})$ and $(\psi', \{(e, 1)\})$ are both coherent, we either have $\bar{I} = \{(d, 0)\}$ or $\bar{I} = \{(e, 1)\}$. Suppose that $\bar{I} = \{(e, 1)\}$. In that case, since $(\psi', \{(e, 1), (m, 1)\})$ is incoherent, we obtain $\bar{I}' = \{(m, 1)\}$ by the postulates (P8) and (P10). Thus, (P13^{*}) doesn't hold.

In separate revision, the agent has to choose whether to go eating or to go to the dentist after revising beliefs. When it chooses to go eating, it has to drop this intention again when deciding to go to the movies, since these two intentions conflict. In joint revision, this is not the case since the agent can compare going to the movies with both possibilities and choose the biggest set of intentions.

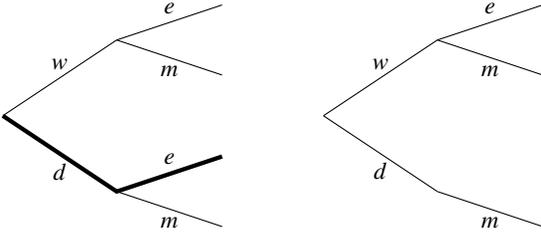


Figure 3: Left: Partial model of strong beliefs ψ of agent (ψ, I) with $I = \{(d, 0), (e, 1)\}$ (bold lines). Right: Revised strong beliefs of agent after learning its no possible to eat (e) after the dentist (d).

The following proposition states that revising with nothing doesn't change a coherent agent. That is, the new beliefs are equivalent to the one prior to revision and the intention database is unchanged.

Proposition 3 *Suppose an agent (ψ, I) is coherent, and $(\psi, I) *_t (\top, \varepsilon) = (\psi', I')$. Then $\psi \equiv \psi'$ and $I = I'$.*

5.2 Representation Theorem

We next characterize all revision schemes satisfying (P1)-(P12) in terms of minimal change with respect to an ordering among interpretations and a selection function accommodating new intentions while restoring coherence. We bound models of strong beliefs up to t , which means that all the paths in the model are “cut off” at t . This ensures finitely many non-equivalent formulas for some belief set B . A t -bounded model $m^t = (T^t, \pi^t)$ is a model containing a tree T in which all paths, including π , have length t . Strong beliefs are about possibility and necessity, and they are independent of a specific path. Therefore, if a single path in a tree is a model of a strong belief, then all paths in this tree are models of this strong belief. Formally, a set of models of a strong belief M_{SB} satisfies the following condition:

If $(T, \pi) \in M_{SB}$, then $(T, \pi') \in M_{SB}$ for all $\pi' \in T$.

A set of t -bounded models of a strong belief M_{SB}^t contains only t restricted models of a strong belief. We write \mathbb{M}_{SB}^t to denote the set of all sets of t -bounded models of strong beliefs. We now show that we can represent a set of models of strong beliefs by a single formula.

Lemma 2 *Let $Ext(M_{SB}^t)$ be the set of all possible extensions of a set of bounded model of strong beliefs M_{SB}^t to models, i.e. $Ext(M_{SB}^t) = \{m \in \mathbb{M} \mid m^t \in M_{SB}^t\}$. Given a set of t -bounded models of strong beliefs M_{SB}^t , there exists a strong belief formula $form(M_{SB}^t)$ such that $Mod(form(M_{SB}^t)) = Ext(M_{SB}^t)$.*

Corollary 1 *Given a t -bounded strong belief set B , there exists a formula ψ such that $B = \{\varphi \mid \psi \vdash \varphi\}$.*

Proof Sketch. For a given belief set B , we can show that there exists a set of t -bounded models of a strong belief M_{SB}^t s.t. $Ext(M_{SB}^t) = Mod(B)$. If $\psi = form(M_{SB}^t)$, then $Mod(\psi) = Mod(B)$, and by the completeness theorem, $B = Cl(\psi)$. \square

Given an intention database I , we define a selection function γ_I^t that tries to accommodate a new intention based on strong beliefs. The selection function specifies preferences on which intention an agent would like to keep in the presence of the new beliefs.

Definition 13 (Selection Function) *Given an intention database I , a selection function $\gamma_I^t : \mathbb{M}_{SB} \times \mathbb{I} \rightarrow \mathbb{D}$ maps a set of models of a strong belief and an intention to an updated intention database—all bounded up to t —such that if $\gamma_I^t(M^t, \{i\}) = I'$, then:*

1. (M^t, I') is coherent.
2. If $(M^t, \{i\})$ is coherent, then $i \in I'$.
3. If $(M^t, I \cup \{i\})$ is coherent, then $I \cup \{i\} \subseteq I'$.
4. $I' \subseteq I \cup \{i\}$.
5. For all I'' with $I' \subset I'' \subseteq I \cup \{i\}$: (M^t, I'') is not coherent.

The five conditions on the selection function are in direct correspondence with postulates (P7)-(P10), (P12) of the agent revision function $*_t$. Note that postulate (P11) doesn't have a corresponding condition in the definition above but is represented by the fact that the selection function takes the revised beliefs as input. That is, intention revision occurs after belief revision.

The following proposition states that a selection function does not change intentions unnecessarily. That is, if an intention is already in the intention database, or if it's empty, the intention database remains unchanged.

Proposition 4 *Given some coherent pair (M, I) , if $i = \varepsilon$ or $i \in I$, then $\gamma_I^t(M, i) = I$.*

KM define a faithful assignment from a belief formula to a pre-order over models. Since we are also considering intentions, we extend this definition such that it also maps intentions databases to selection functions.

Definition 14 (Faithful assignment) *A faithful assignment is a function that assigns to each strong belief formula $\psi \in \mathbb{B}^t$ a total pre-order \leq_ψ^t over \mathbb{M} and to each intention database $I \in \mathbb{D}^t$ a selection function γ_I^t and satisfies the following conditions:*

1. If $m_1, m_2 \in Mod(\psi)$, then $m_1 \leq_\psi^t m_2$ and $m_2 \leq_\psi^t m_1$.
2. If $m_1 \in Mod(\psi)$ and $m_2 \notin Mod(\psi)$, then $m_1 < m_2$.
3. If $\psi \equiv \phi$, then $\leq_\psi^t = \leq_\phi^t$.
4. If $T^t = T_2^t$, then $(T, \pi) \leq_\psi^t (T_2, \pi_2)$ and $(T_2, \pi_2) \leq_\psi^t (T, \pi)$.

Conditions 1 to 3 on the faithful assignment are the same as those of KM. Condition 4 ensures that we do not distinguish between models in the total pre-order \leq_ψ^t whose trees are the same up to time t . This is essentially what is represented in the revision function by bounding the all input of the revision function $*_t$ up to t . Moreover, \leq_ψ^t does not distinguish between models obtained by selecting two different paths from the same tree. This corresponds to the fact that we are using strong belief formulas in the revision, which do not distinguish between different paths in the same tree as well.

Theorem 2 (Representation Theorem) *An agent revision operator $*_t$ satisfies postulates (P1)-(P12) iff there exists a faithful assignment that maps each ψ to a total pre-order \leq_ψ^t and each I to a selection function γ_I^t such that if $(\psi, I) *_t (\varphi, i) = (\psi', I')$, then:*

1. $Mod(\psi') = \min(Mod(\varphi), \leq_\psi^t)$
2. $I' = \gamma_{I'}^t(Mod(\psi'), i)$

Finally, it turns out to be straightforward to formulate the DP postulates for iterated revision in our framework for the strong beliefs and to prove their representation theorem. Due to space constraints we have omitted these results.

6 Related Work

Grant *et al.* [9] develop AGM-style postulates for belief, intention, and goal revision. They provide a detailed analysis and propose different reconsideration strategies, but restrict themselves to a syntactic analysis. Much effort in combining AGM revision with action logics (e.g., the Event Calculus [17], Temporal Action Logics [13], extensions to the Fluent Calculus [25], and extensions to the Situation Calculus (see [18, Ch.2] for an overview)) concentrates on extending these action theories to incorporate *sensing* or *knowledge-producing actions*. Shapiro *et al.* [22] extend the Situation Calculus to reason about beliefs rather than knowledge by introducing a modality B and shows that both the AGM postulates and the DP postulates are satisfied in this framework. A similar approach concerning the Fluent Calculus has been formalized by Jin and Thielscher [11], and is further developed by Scherl [20] and Scherl and Levesque [21] by taking into account the frame problem as well. However, none of these approaches prove representation theorems linking revision to a total pre-order on models. Baral and Zhang [2] model belief updates on the basis of semantics of modal logic S5 and show that their knowledge update operator satisfies all the KM postulates. Bonanno [3] combines temporal logic with AGM belief revision by extending a temporal logic with a belief operator and an information operator. Both these approaches do not take action or time into account and do not prove representation theorems. The concept of strong beliefs has been discussed extensively in the literature, for instance in the story of *Little Nell* [16] or a paradox found in *knowledge-based programs* [8] (see van der Hoek *et al.* [27] for a detailed discussion).

7 Conclusion

We develop a logical theory for reasoning about temporal beliefs and intentions based on Shoham's database perspective. We propose postulates for revision of strong beliefs and intentions, and prove a representation theorem relating the postulates to our formal model.

For future work, we aim to make the role of the planner more explicit. Currently, our agent only receives updates from the planner, but allowing the agent to do planning tasks itself would allow it to, for instance, replace intentions instead of merely discarding them. This paves the road to develop a richer notion of intentions. If the databases take over part of the planning, then well-known problems such as the frame problem become more stringent: Once a fact is established (for example, as a postcondition of an intention), it persists until it explicitly contradicts postconditions established by future intentions. Existing action logics (e.g., the Event Calculus or the Fluent Calculus) have dealt with these problems in detail, so comparing and possibly enriching them with our formalism seems both useful and relevant future work. Finally, it is our long-term goal to apply Shoham's database perspective to decision making in large-scale enterprises [28, 29, 32], in a similar way Timeful applied it to decision making for individuals.

Acknowledgments We thank Leon van der Torre, Eric Pacuit, and Thomas Icard for useful comments. Marc van Zee and Dragan Doder are both funded by the National Research Fund (FNR), Luxembourg, respectively by the RationalArchitecture and the PRIMAT project.

Proof Sketches

Proposition 5 (Coherence Condition 2b) *If $\{(a,t), (b,t+1)\} \subseteq I$ and B is a set of strong beliefs such that (B,I) coherent, then $\{pre(b)_{t+1}, post(a)_{t+1}\}$ is consistent with B .*

Proof Sketch. Let $I = \{(a,t), (b,t+1)\}$ and let B be a set of strong beliefs whose set of models is M . We need to show that consistency of $\{pre(b)_{t+1}, post(a)_{t+1}\}$ with B follows from our coherence condition. Note that the coherence formula is $Cohere(I) = \diamond_0 pre(a,b)_t$. By the axiom (A12) (Definition 4) and the Deduction theorem we have $pre(a,b)_t \wedge do(a)_t \vdash pre(b)_{t+1}$. Using the axiom (A9): $do(a)_t \rightarrow post(a)_{t+1}$ from PAL (Def. 7 [30]) and Deduction theorem we obtain $do(a)_t \vdash post(a)_{t+1}$. Consequently,

$$pre(a,b)_t \wedge do(a)_t \vdash pre(b)_{t+1} \wedge post(a)_{t+1}.$$

Thus, in order to prove that $\{pre(b)_{t+1}, post(a)_{t+1}\}$ is consistent with B , it is sufficient to show that $pre(a,b)_t \wedge do(a)_t$ is consistent with B . If (B,I) is coherent, then there is a model $m = (T, \pi) \in M$ such that $m \models Cohere(I)$, so there is $\pi' \in T$ such that $(T, \pi') \models pre(a,b)_t$. By Definition 3.2, then there exist $\pi'' \in T$ such that $\pi' \sim_t \pi''$ and $act(\pi'') = a$. Then $(T, \pi'') \models pre(a,b)_t \wedge do(a)_t$. Since M is a set of models of strong beliefs, we obtain $(T, \pi'') \in M$, i.e., (T, π'') is also a model of B . Then B is consistent with $pre(a,b)_t \wedge do(a)_t$, by Completeness theorem. \square

Proposition 6 *If an agent $A = (B,I)$ is coherent, then $WB(B,I)$ is consistent.*

Proof Sketch. Using axioms A8, A11, and A12, one can show that $\{pre(a_0, \dots, a_m)_t\} \vdash \diamond_t (do(a_0)_t \wedge \diamond_{t+1} (do(a_1)_{t+1} \wedge \diamond_{t+2} (\dots)))$. By taking the contrapositive of A3, $pre(a_0, \dots, a_m)_t$ implies $\diamond_t \bigwedge_{k=0}^m do(a_k)_{t+k}$. Therefore, $Cohere(I)$ (Def. 10) implies

$$\diamond_0 \bigvee_{\substack{a_k \in Act: k \notin \{t_1, \dots, t_k\} \\ a_k = b_k: k \in \{t_1, \dots, t_n\}}} \diamond_{t_1} (do(a_{t_1})_{t_1} \wedge do(a_{t_1+1})_{t_1+1} \wedge \dots \wedge do(a_{t_n})_{t_n})$$

Consequently, $Cohere(I)$ implies $\diamond_0 \diamond_{t_1} \bigwedge_{k=1}^n do(b_{t_k})_{t_k}$, and by (A3) this implies $\diamond_0 \bigwedge_{(a,t) \in I} do(a)_t$. Therefore, if (B,I) is coherent, then the set $B \cup \{\diamond_0 \bigwedge_{(a,t) \in I} do(a)_t\}$ is consistent. By the fact that B is a strong belief set, $B \cup \{\bigwedge_{(a,t) \in I} do(a)_t\}$ is consistent, i.e. $WB(B,I)$ is consistent. \square

Theorem 3 (Representation Theorem) *An agent revision operator $*_t$ satisfies postulates (P1)-(P12) iff there exists a faithful assignment that maps each ψ to a total pre-order \leq_ψ^t and each I to a selection function γ_I^t such that if $(\psi, I) *_t (\varphi, i) = (\psi', I')$, then:*

1. $Mod(\psi') = \min(Mod(\varphi), \leq_\psi^t)$
2. $I' = \gamma_I^t(Mod(\psi'), i)$

Proof Sketch. We only sketch the proof of " \Rightarrow ": Suppose that some agent revision operator $*_t$ satisfies postulates (P1)-(P12). Given models m_1 and m_2 , let $(\psi, \emptyset) *_t (form(m_1^t) \vee form(m_2^t), \varepsilon) = (\psi', \emptyset)$. We define \leq_ψ^t by $m_1 \leq_\psi^t m_2$ iff $m_1 \models \psi$ or $m_1 \models \psi'$. We also define γ_I^t by $\gamma_I^t(M^t_{SB}, i) = I'$, where $(form(M^t_{SB}), I) *_t (\top, i) = (\psi_2, I')$ (note that $\psi_2 \equiv form(M^t_{SB})$).

Let us prove condition 4 of Definition 9. For $m_1 = (T, \pi)$ and $m_2 = (T_2, \pi_2)$, let ψ' be as above. Since $\psi, \psi' \in \mathbb{B}^t$ and $T^t = T_2^t$, we have $m_1 \models \psi$ iff $m_2 \models \psi$ and $m_1 \models \psi'$ iff $m_2 \models \psi'$, so $m_1 \leq_\psi^t m_2$ and $m_2 \leq_\psi^t m_1$.

Following KM, one can show that conditions 1 to 3 from Definition 9 hold, and furthermore that $Mod(\psi') = \min(Mod(\varphi), \leq_\psi^t)$. We now prove $I' = \gamma_I^t(Mod(\psi'), i)$. By our definition of γ_I^t we have that $(\psi', I) *_t (\top, i) = (\psi_2, \gamma_I^t(Mod(\psi'), i))$ (recall that $\psi' \equiv \psi_2$). Since $(\psi, I) *_t (\varphi, i) = (\psi', I')$, by (P11) we obtain that $I' = \gamma_I^t(Mod(\psi'), i)$. Using postulate (P7)-(P10) and (P12) we can prove that γ_I^t is a selection function. \square

REFERENCES

- [1] Carlos E. Alchourron, Peter Gärdenfors, and David Makinson, 'On the logic of theory change: Partial meet contraction and revision functions', *Journal of Symbolic Logic*, **50**(2), 510–530, (06 1985).
- [2] Chitta Baral and Yan Zhang, 'Knowledge updates: Semantics and complexity issues', *Artificial Intelligence*, **164**(1), 209–243, (2005).
- [3] Giacomo Bonanno, 'Axiomatic characterization of the AGM theory of belief revision in a temporal logic', *Artificial Intelligence*, **171**(2), 144–160, (2007).
- [4] Michael E. Bratman, *Intention, plans, and practical reason*, Harvard University Press, Cambridge, MA, 1987.
- [5] Philip R Cohen and Hector J Levesque, 'Intention is choice with commitment', *Artificial Intelligence*, **42**(2-3), 213–261, (1990).
- [6] Philip R. Cohen and Hector J. Levesque, 'Teamwork', *Noûs*, **25**(4), 487–512, (1991).
- [7] Hans Ditmarsch, Tiago Lima, and Emiliano Lorini, *Intention Change via Local Assignments*, 136–151, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [8] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi, *Reasoning about Knowledge*, MIT Press, 1995.
- [9] John Grant, Sarit Kraus, Donald Perlis, and Michael Wooldridge, 'Postulates for revising BDI structures', *Synthese*, **175**(1), 39–62, (2010).
- [10] Thomas Icard, Eric Pacuit, and Yoav Shoham, 'Joint revision of belief and intention', *Proc. of the 12th International Conference on Knowledge Representation*, 572–574, (2010).
- [11] Yi Jin and Michael Thielscher, 'Representing beliefs in the fluent calculus.', in *ECAI*, pp. 823–827. IOS Press, (2004).
- [12] Hirofumi Katsuno and Alberto O. Mendelzon, 'Propositional knowledge base revision and minimal change', *Artificial Intelligence*, **52**(3), 263–294, (dec 1991).
- [13] Jonas Kvarnström, *TALplanner and other extensions to Temporal Action Logic*, Ph.D. dissertation, Linköpings universitet, 2005.
- [14] Emiliano Lorini, Mehdi Dastani, Hans P. van Ditmarsch, Andreas Herzig, and John-Jules Ch. Meyer, 'Intentions and assignments.', in *LORI*, volume 5834 of *Lecture Notes in Computer Science*, pp. 198–211. Springer, (2009).
- [15] Emiliano Lorini and Andreas Herzig, 'A logic of intention and attempt', *Synthese*, **163**(1), 45–77, (2008).
- [16] Drew McDermott, 'A temporal logic for reasoning about processes and plans', *Cognitive science*, **6**(2), 101–155, (1982).
- [17] Erik T Mueller, *Commonsense reasoning*, Morgan Kaufmann, 2010.
- [18] Theodore Patkos, *A formal theory for reasoning about action, knowledge and time*, Ph.D. dissertation, University of Crete-Heraklion, 2010.
- [19] M. Reynolds, 'An axiomatization of full computation tree logic', *Journal of Symbolic Logic*, **66**(3), 1011–1057, (2002).
- [20] Richard B Scherl, 'Action, belief change and the frame problem: A fluent calculus approach', in *Proceedings of the Sixth workshop on Non-monotonic Reasoning, Action, and Change at IJCAI*, (2005).
- [21] Richard B Scherl and Hector J Levesque, 'Knowledge, action, and the frame problem', *Artificial Intelligence*, **144**(1), 1–39, (2003).
- [22] Steven Shapiro, Maurice Pagnucco, Yves Lesprance, and Hector J. Levesque, 'Iterated belief change in the situation calculus.', *Artificial Intelligence*, **175**(1), 165–192, (2011).
- [23] Yoav Shoham, 'Logical theories of intention and the database perspective', *Journal of Philosophical Logic*, **38**(6), 633–647, (2009).
- [24] Yoav Shoham, 'Why knowledge representation matters', *Commun. ACM*, **59**(1), 47–49, (January 2016).
- [25] Michael Thielscher, 'The concurrent, continuous fluent calculus', *Studia Logica*, **67**(3), 315–331, (2001).
- [26] Wiebe van der Hoek, Wojciech Jamroga, and Michael Wooldridge, 'Towards a theory of intention revision', *Synthese*, **155**(2), 265–290, (2007).
- [27] Wiebe Van der Hoek and Michael Wooldridge, 'Towards a logic of rational agency', *Logic Journal of IGPL*, **11**(2), 135–159, (2003).
- [28] Dirk van der Linden and Marc van Zee, 'Insights from a Study on Decision Making in Enterprise Architecture.', in *PoEM (Short Papers)*, volume 1497 of *CEUR Workshop Proceedings*, pp. 21–30, (2015).
- [29] Marc van Zee, 'Rational Architecture = Architecture from a Recommender Perspective', in *Proceedings of the International Joint Conference on Artificial Intelligence*, (2015).
- [30] Marc van Zee, Mehdi Dastani, Dragan Doder, and Leendert van der Torre, 'Consistency Conditions for Beliefs and Intentions', in *Twelfth International Symposium on Logical Formalizations of Commonsense Reasoning*, (2015).
- [31] Marc van Zee, Dragan Doder, Mehdi Dastani, and Leendert van der Torre, 'AGM Revision of Beliefs about Action and Time', in *Proceedings of the International Joint Conference on Artificial Intelligence*, (2015).
- [32] Marc Van Zee, Georgios Plataniotis, Dirk van der Linden, and Diana Marosin, 'Formalizing enterprise architecture decision models using integrity constraints', in *2014 IEEE 16th Conference on Business Informatics*, volume 1, pp. 143–150. IEEE, (2014).

Facility Location Games with Optional Preference

Hongning Yuan¹ and Kai Wang² and Ken C.K. Fong³ and Yong Zhang⁴ and Minming Li⁵

Abstract. In this paper, we propose the optional preference model for the facility location game with two heterogeneous facilities on a line. Agents in this new model are allowed to have optional preference, which gives more flexibility for agents to report. Aiming at minimizing maximum cost or sum cost of agents, we propose different deterministic strategy-proof mechanisms without monetary transfers. Depending on which facility the agent with optional preference cares for, we consider two variants of the optional preference model: Min (caring for the closer one) and Max (caring for the further one). For the Min variant, we propose a 2-approximation mechanism for the maximum cost objective, as well as a lower bound of $4/3$, and a $(n/2+1)$ -approximation mechanism for the sum cost objective, as well as a lower bound of 2. For Max variant, we propose an optimal mechanism for the maximum cost objective and a 2-approximation mechanism for the sum cost objective.

1 INTRODUCTION

In this paper, we study facility location games with optional preference. The classic facility location game models the scenario where the government plans to build a public facility on a street (modeled as a line segment) where some self-interested agents who tend to minimize their own costs are situated. The agents are required to report their locations as private information, which will then be mapped to a single facility location by a mechanism. The purpose of the mechanism is to optimize certain objectives like minimizing the sum cost or maximum cost. Previous works on facility location games can mainly be classified into three categories.

1. Only one facility is built and agents may like or dislike the facility.
2. Two or more homogeneous facilities are built and agents care for the closer one.
3. Two heterogeneous facilities are built and agents' utilities are the summation of their utilities towards these two facilities.

Preference is a basic and key element in the facility location problem as we can see from the above three categories. For the two heterogeneous facility location game, there are three kinds of preferences, namely three kinds of agents: agents who like facility 1 (F_1), agents who like facility 2 (F_2) and agents who like both facility 1 and 2 (F_1, F_2). In the existing literature, an agent who likes both facilities needs to access both of them at the same time and therefore his cost will be the sum of the distances from these two facilities [15]. In this paper, we generalize the cost function to be an arbitrary function of the agent's distances to the two facilities. A special case

of this generalization is the optional preference model we focus on in this paper, where the agent's cost depends on the distance from the closer/further facility. The optional preference model for the two heterogeneous facility location game is also a natural extension from the two homogeneous facility location game.

We consider two variants of optional preference: Min and Max, in which the agents with preference for both facilities only care for the closer one (Min) or the further one (Max) respectively. Both of these two variants can find their applications in many real life scenarios. For the Min variant, consider a local government building bus stops for two bus routes on a street. Residents on this street whose destinations for work are covered by both bus routes can go to either of the stops, and to reduce their walking distance they would definitely pick the closer one. For the rest who must take one of the bus routes, they need to go to the respective ones. For the Max variant, consider a factory requiring two different raw materials from two sites to start manufacturing. The factory has multiple trucks which can be sent out simultaneously. Therefore, assuming that trucks have the same speed, the time the factory needs to wait for depends on its distance to the further site. We allow the two facilities to be put in the same location which is well justified in the bus stop scenario since the government has the choice to build one bus stop for two bus routes and also in the factory scenario since the storage sites of the raw materials can be the same place.

In the scenarios mentioned above, we assume all agents know the mechanisms that the government will adopt. An agent may have a chance to reduce his cost by misreporting. Given that locations are usually detectable by the government, we assume that agents can only lie about their preferences. A mechanism is strategy-proof if it can guarantee that an agent cannot reduce his cost by misreporting. In addition, we need to evaluate the mechanisms in terms of optimization of sum cost (the sum of costs of all agents) or the maximum cost (the maximum cost of all agents). The evaluation is mainly conducted by the approximation ratio for the sum cost/maximum cost of a mechanism, which is the worst ratio between the sum cost/maximum cost of the mechanism output and the optimal sum cost/maximum cost among all possible profiles.

1.1 Our contribution

We initiated the study of the optional preference model under two objectives: minimizing maximum cost or sum cost. We proposed strategy-proof mechanisms and also derived lower bounds for this new model.

- For the Min variant, to minimize the maximum cost we proposed a strategy-proof 2-approximation mechanism and showed that it is impossible to achieve an approximation ratio better than $4/3$. To minimize sum cost, we gave a $(n/2+1)$ -approximation strategy-proof mechanism where n is the number of agents and managed

¹ City University of Hong Kong, email: hongnyuan2-c@my.cityu.edu.hk

² City University of Hong Kong, email: kai.wang@my.cityu.edu.hk

³ City University of Hong Kong, email: ken.fong@my.cityu.edu.hk

⁴ Shenzhen Institutes of Advanced Technology, email: zhangyong@siat.ac.cn

⁵ City University of Hong Kong, email: minming.li@cityu.edu.hk

to prove a lower bound of 2 in the form of a limit by showing that the approximation ratio infinitely approaches 2 with the increase of the number of agents.

- For the Max variant, we proposed an optimal strategy-proof mechanism to minimize the maximum cost. Meanwhile, we also proposed a 2-approximation strategy-proof mechanism to minimize the sum cost.

1.2 Related work

Mechanism design for the facility location game was firstly studied by Procaccia and Tennenholtz [13] where results for single-peaked preference by Moulin [12] were adopted. They also extended the facility location game to the scenario with two homogeneous facilities or with one agent possessing multiple locations. Lu et al. [11] [10] improved the bounds for both the two homogeneous facilities scenario and one agent possessing multiple locations case. Fotakis and Tzamos [8] explored the facility location game with k facilities and showed that the strategy-proofness can be achieved by adding winner-imposing constraints. Filos-Ratsikas et al. [7] extended the single-peaked preference to double-peaked preference where every agent has two ideal places for the facility on his two sides. Serafino and Ventre [14, 15] initiated the study on two heterogeneous facility location game where the cost of the agent is the summation of the distances to both facilities. Other extensions of the classic facility location game can be found at [1, 9, 16, 4, 6, 18].

Meanwhile, Cheng et al. [2] initiated the mechanism design for obnoxious facility location games, where agents have the preference to stay as far away as possible from the facility, with both deterministic and randomized group strategy-proof mechanisms. Later they further extended the model into trees and circles [3]. Ye et al. [17] considered the problem with the objective of maximizing sum of squares of distances and sum of distances. They gave lower bounds and proposed both deterministic and randomized mechanisms.

Combining the above two directions together, dual preference model was studied in [19, 5], where some agents want to be close to the facility while others want to be far away from the facility. Zou et al. [19] also studied another model where two facilities need to be built within a certain distance and agents want to be close to one of the facilities but far away from the other facility.

2 PRELIMINARIES

Let $N = \{1, 2, \dots, n\}$ be the set of agents located on a line from left to right in sequence and $\mathcal{F} = \{F_1, F_2\}$ be the set of facilities to build. Each agent has a location $x_i \in \mathbb{R}$ and a preference $p_i \subseteq \mathcal{F}$, where x_i is public information and p_i is private information, which can be reported as single preference $\{F_1\}$, $\{F_2\}$ or optional preference $\{F_1, F_2\}$. We denote X_k as the set of locations of all agents with preference $\{F_k\}$: $X_k = \{x_i \mid p_i = \{F_k\}, i \in N\}$, and $X_{1,2}$ as the set of locations of all agents with optional preference: $X_{1,2} = \{x_i \mid p_i = \{F_1, F_2\}, i \in N\}$. We also denote L as the full length of the line segment with agents located on, i.e., $x_1 = 0$ and $x_n = L$. A location profile \mathbf{x} is a collection of location of the agents $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ where $x_1 \leq x_2 \leq \dots \leq x_n$. A preference profile \mathbf{p} is a collection of preference reported by all agents $\mathbf{p} = \{p_1, p_2, \dots, p_n\}$. A mechanism M is a function that takes a profile \mathbf{p} as input and returns the building locations of the facilities as output. A solution \mathbf{s} is a pair of building locations of the two facilities $\mathbf{s} = (y_1, y_2)$, or $\mathbf{s}(M) = (y_1, y_2)$ with a given mechanism M , with

y_1 being the building location for F_1 and y_2 for F_2 . Given the locations of the facilities, we will use $d(i, F_k)$, or $d_{\mathbf{s}}(i, F_k)$ ($k \in \{1, 2\}$) with a given solution \mathbf{s} , to denote the distance from agent i to F_k . The cost of agent i , denoted as $cost_i$, or $cost_i(\mathbf{s})$ with a given solution \mathbf{s} , is a function of the distances between him and his preferred facilities, which may vary under different settings. For the Min variant of the optional preference model, we have the cost of agent i as $cost_i = \min_{F_k \in p_i} d(i, F_k)$. In other words, the cost of agents with optional preference depends on the closer one of the two facilities. While for the Max variant of the optional preference model, we have the cost of agent i as $cost_i = \max_{F_k \in p_i} d(i, F_k)$. In other words, the cost of agents with optional preference depends on the further one of the two facilities.

Since we focus on the optional preference model, the cost of any agent eventually depends on only one facility. If an agent's cost eventually depends on the distance between him and facility F_k , we say he is F_k -typed. We want to design strategy-proof mechanisms so that no agent can benefit from lying about his preference.

Definition 1 A mechanism M is strategy-proof if for any agent i , we have $cost_i(\mathbf{s}) \leq cost_i(\mathbf{s}')$, where \mathbf{s} is the solution output by M when agent i reports truthfully and \mathbf{s}' is the solution output by M when agent i lies.

In this paper, we consider two objectives for minimizing the cost of an agent set: minimizing the maximum cost, or minimizing the sum cost. Given an agent set, the *sum cost* is the sum of costs from all agents $sc = \sum_{i \in N} cost_i$, or $sc(\mathbf{s}) = \sum cost_i(\mathbf{s})$ with a given solution \mathbf{s} . The *maximum cost* is the maximum value of the costs from all agents $mc = \max_{i \in N} cost_i$, or $mc(\mathbf{s}) = \max_{i \in N} cost_i(\mathbf{s})$ with a given solution \mathbf{s} . Our primary goal is to design strategy-proof mechanisms to allocate the facilities. However sometimes the solution output by the mechanism may not be optimal. Given an objective, we say a mechanism is α -approximation or has an approximation ratio α if for any given profile, let \mathbf{s} be the solution returned by the mechanism and \mathbf{s}^* be the solution with optimal cost, we have $sc(\mathbf{s}) \leq \alpha \cdot sc(\mathbf{s}^*)$ for sum cost, or $mc(\mathbf{s}) \leq \alpha \cdot mc(\mathbf{s}^*)$ for maximum cost.

3 OPTIONAL PREFERENCE (MIN)

In this section, we study the Min variant of the optional preference model, i.e., $cost_i = \min_{F_k \in p_i} d(i, F_k)$.

3.1 Maximum Cost

We first analyze the objective of minimizing the maximum cost.

Upper bound

We present a 2-approximation strategy-proof mechanism.

Mechanism 1 Given a profile, let solution $\mathbf{s}_1 = (0, L)$ and solution $\mathbf{s}_2 = (L, 0)$. If $mc(\mathbf{s}_1) \leq mc(\mathbf{s}_2)$, output \mathbf{s}_1 , otherwise output \mathbf{s}_2 .

Theorem 1 Mechanism 1 is strategy-proof.

Proof. From the mechanism, agents with preference $\{F_1, F_2\}$ do not have the incentive to lie as the two possible outputs are the same in their perspectives.

In the following, we focus on any agent i of single preference. Since his location is fixed and the facilities locations in the two outputs are fixed too, his cost can only be either x_i or $L - x_i$. One of them will be $cost_i(\mathbf{s}_1)$, and the other will be $cost_i(\mathbf{s}_2)$.

If solution s_1 is returned, we can see that agent i only has an incentive to change the output when $cost_i(s_1) > L/2 > cost_i(s_2)$, which means he now has the larger one of the two possible costs and he wants to change to the smaller one. In this case, however, since agent i already has the larger one of the two possible costs, he cannot make the mechanism think that he has a larger cost in solution s_1 no matter what preference he lies to have. Since agent i is the only lying agent, the value of $mc(s_1)$ will not increase. For the same reason, since his cost in solution s_2 is the smaller one, he cannot make the mechanism think that he has a smaller cost in solution s_2 . Since he is the only lying agent, the value of $mc(s_2)$ will not decrease. Therefore we will still have $mc(s_1) \leq mc(s_2)$, which means agent i cannot change the output by lying.

The analysis is similar when solution s_2 is returned.

Therefore, Mechanism 1 is strategyproof. \square

Theorem 2 Mechanism 1 is 2-approximation.

Proof. When there are two agents, it is easy to check that Mechanism 1 is 2-approximation.

When there are more than two agents, according to the preference of agent 1 and agent n , there are four cases:

Case 1. $p_1 = p_n = \{F_1\}$ or $p_1 = p_n = \{F_2\}$. In this case, it is clear that the maximum cost returned by the mechanism is L which comes from agent 1 or agent n , and the maximum cost in the optimal solution is $L/2$ which comes from agent 1 and agent n . Therefore the approximation ratio is $\frac{L}{L/2} = 2$ in this case.

Case 2. $p_1 = \{F_1\}$ and $p_n = \{F_2\}$. In this case, according to the mechanism solution $s_1 = (0, L)$ is output. We will analyze the case when the maximum cost comes from an F_1 -typed agent in solution s_1 . The analysis when it comes from an F_2 -typed agent will be similar by symmetry. Let agent m be this agent, we can see that the maximum cost now will be $x_m - 0 = x_m$ in solution s_1 . Meanwhile, if $p_m = \{F_1, F_2\}$, we have $x_m \leq L/2$ otherwise he will be F_2 -typed in solution s_1 . Therefore, in the optimal solution, if he is F_1 -typed, we can see that the optimal maximum cost will be at least $(x_m - x_1)/2 = x_m/2$. If he is F_2 -typed, then the optimal maximum cost will be at least $(L - x_m)/2 \geq x_m/2$. As a result, the approximation ratio will be at most $\frac{x_m}{x_m/2} = 2$ in this case.

Case 3. $p_1 = \{F_2\}$ and $p_n = \{F_1\}$. The analysis is similar to Case 2 by symmetry.

Case 4. $\{F_1, F_2\} \in \{p_1, p_n\}$. In this case, if agent 1 and agent n are of the same type, then the analysis will be similar to Case 1. If they are of different types, then the analysis will be similar to Case 2 or Case 3.

Therefore, Mechanism 1 is 2-approximation. \square

Lower bound

We show that the optimal solution is not strategy-proof and derive the lower bound of the approximation ratio.

Consider two instances with the same location profile $\mathbf{x} = \{0, 2, 4, 6\}$ depicted in Figure 1(a) and Figure 1(b), respectively. The agents in Figure 1(a) have the preferences $p_1 = p_2 = p_4 = \{F_1, F_2\}$ and $p_3 = \{F_2\}$; while the agents in Figure 1(b) have the preference $p_1 = p_3 = p_4 = \{F_1, F_2\}$ and $p_2 = \{F_2\}$. To minimize the maximum distance between each agent and his preferred facility, the optimal allocation for the cases in Figure 1(a) and in Figure 1(b) are $s_1 = (1, 5)$ and $s_2 = (5, 1)$, respectively. Both have the optimal cost 1. Note that they are the only optimal solutions.

Consider another instance with the location profile $\mathbf{x} = \{0, 2, 4, 6\}$ depicted in Figure 1(c). The agents in Figure 1(c) have

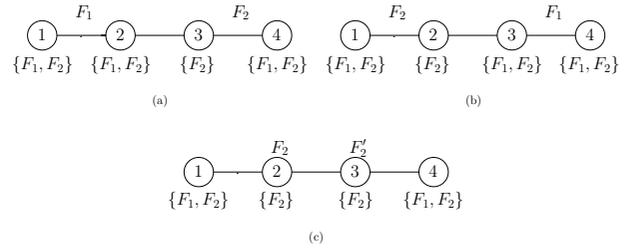


Figure 1. Instances used for proving the lower bound of 4/3.

the preference $p_1 = p_4 = \{F_1, F_2\}$ and $p_2 = p_3 = \{F_2\}$. For this instance, there are two types of optimal solutions $s'_1 = (y_1, 2)$ where $4 \leq y_1 \leq 6$ and $s'_2 = (y_1, 4)$ where $0 \leq y_1 \leq 2$. Both types have the optimal cost 2.

However, neither s'_1 nor s'_2 can be the output of a strategy-proof mechanism. Given the input as shown in Figure 1(c), if the mechanism's output is s'_1 , agent 2 from Figure 1(a) would be able to gain by lying about his preference to be $p'_2 = \{F_2\}$; otherwise, if the mechanism's output is s'_2 , agent 3 from Figure 1(b) would be able to gain by lying about his preference to be $p'_3 = \{F_2\}$.

Theorem 3 There exists no α approximation deterministic strategy-proof mechanism for the facility location problem with $\alpha < 4/3$.

Proof. Assume there is a deterministic strategy-proof mechanism M with the approximation ratio α . Since the configuration in Figure 1(c) is symmetric, without loss of generality, we may assume that mechanism M 's output for this instance is $(y_1(c), y_2(c))$ such that $y_1(c) \geq y_2(c)$. In other words, F_2 is allocated to the left of F_1 . Consider M 's output $(y_1(a), y_2(a))$ for the instance in Figure 1(a).

1. If $y = \min\{y_1(a), y_2(a)\} \leq 1$, to guarantee the strategy-proofness, agent 2 cannot gain by misreporting his profile from $\{F_1, F_2\}$ to be $\{F_2\}$. That means for instance Figure 1(c), the mechanism cannot allocate any facility between $(y, 4 - y)$. From the previous assumption, in Figure 1(c), the mechanism allocates F_2 to the left. In this case, the maximum cost is at least $4 - y$ while the optimal value is 2. Thus, the approximation ratio is at least $(4 - y)/2 \geq 3/2$.
2. Otherwise, $y = \min\{y_1(a), y_2(a)\} > 1$. Similarly to the previous analysis, to guarantee the strategy-proofness, the mechanism cannot allocate any facility between $(y, 4 - y)$. In this case, the maximum cost is at least $4 - y$ and the approximation ratio is at least $(4 - y)/2$. Note that the approximation ratio for the instance in Figure 1(a) is y . Combining these two cases, the approximation ratio of the mechanism is at least $\max\{(4 - y)/2, y\} \geq 4/3$.

From the above analysis, we can see that the approximation ratio for any strategy-proof mechanism is at least 4/3. \square

3.2 Sum cost

We now focus on the problem under the objective of minimizing the sum cost.

Upper bound

We first present a $(n/2+1)$ -approximation strategy-proof mechanism.

Mechanism 2 Given a profile \mathbf{p} , let profile \mathbf{q} be the profile when we treat every agent in profile \mathbf{p} as an agent with preference $\{F_1, F_2\}$.

We find the optimal solution \mathbf{s}_q^* of profile \mathbf{q} and divide the agents into two sets \mathcal{L} and \mathcal{R} from the middle, i.e., $\mathcal{L} = \{i | x_i \leq (y_1(\mathbf{s}_q^*) + y_2(\mathbf{s}_q^*))/2\}$, $\mathcal{R} = \{i | x_i > (y_1(\mathbf{s}_q^*) + y_2(\mathbf{s}_q^*))/2\}$. Let agent l and agent r be the median agent of set \mathcal{L} and set \mathcal{R} . Then for profile \mathbf{p} , we have solution $\mathbf{s}_1 = (x_l, x_r)$ and $\mathbf{s}_2 = (x_r, x_l)$. If $sc(\mathbf{s}_1) \leq sc(\mathbf{s}_2)$, output \mathbf{s}_1 , otherwise output \mathbf{s}_2 .

Theorem 4 Mechanism 2 is strategy-proof.

Proof. Given the mechanism, first we note that no agent with preference $\{F_1, F_2\}$ has the incentive to lie as the two possible outputs are the same in their perspectives. For agents with preference $\{F_1\}$ or $\{F_2\}$, we analyze the case for an agent in set \mathcal{L} with the output of Mechanism 2 being \mathbf{s}_1 and the results for other cases will be the same by symmetry. If this agent has preference $\{F_1\}$, he has no incentive to lie as his cost only increases in \mathbf{s}_2 . If this agent has preference $\{F_2\}$, lying to have preference $\{F_1\}$ makes $sc(\mathbf{s}_1)$ decrease and $sc(\mathbf{s}_2)$ increase, while lying to have preference $\{F_1, F_2\}$ makes $sc(\mathbf{s}_1)$ decrease and $sc(\mathbf{s}_2)$ remain the same. Note that to output \mathbf{s}_1 we already have $sc(\mathbf{s}_1) \leq sc(\mathbf{s}_2)$, which means the agent cannot change the output of Mechanism 2 by lying.

Therefore, no agent has the incentive to lie, and Mechanism 2 is strategy-proof. \square

Theorem 5 Mechanism 2 is $(n/2+1)$ -approximation, where n is the number of agents.

Proof. Let \mathbf{s} be the solution output by Mechanism 2 and \mathbf{s}' be the other solution in Mechanism 2. Let sc_p and sc_q be the sum cost for profile \mathbf{p} and profile \mathbf{q} respectively, we have $sc_p(\mathbf{s}) \geq sc_q(\mathbf{s})$ and:

$$sc_p(\mathbf{s}) - sc_q(\mathbf{s}) = \sum_{k \in \{1,2\}} \sum_{p_i = \{F_k\}} [d_s(i, F_k) - \min(d_s(i, F_1), d_s(i, F_2))]$$

Let \mathbf{s}_p^* be the optimal solution for profile \mathbf{p} . We want to show the following inequality:

$$sc_p(\mathbf{s}) + sc_p(\mathbf{s}') < (n+2)sc_p(\mathbf{s}_p^*) \quad (1)$$

According to Mechanism 2, we can see that solution \mathbf{s} and \mathbf{s}' have the same cost as solution \mathbf{s}_q^* for profile \mathbf{q} . Since \mathbf{s}_q^* is the optimal solution of profile \mathbf{q} , we have:

$$sc_q(\mathbf{s}) = sc_q(\mathbf{s}') = sc_q(\mathbf{s}_q^*) \leq sc_q(\mathbf{s}_p^*) \leq sc_p(\mathbf{s}_p^*)$$

Moreover, let $\beta_i = \sum_{k \in \{1,2\}} d_s(i, F_k) - 2 \min(d_s(i, F_1), d_s(i, F_2))$ we have: $sc_p(\mathbf{s}) + sc_p(\mathbf{s}') = 2sc_q(\mathbf{s}) + \sum_{k \in \{1,2\}} \sum_{p_i = \{F_k\}} \beta_i$.

Let $d = x_r - x_l$ be the distance between agent l and r . For agent i , if $x_l < x_i < x_r$, we have $\beta_i \leq d$, otherwise $\beta_i = d$.

Since $sc_q(\mathbf{s}) \leq sc_p(\mathbf{s}_p^*)$ and $\beta_i \leq d$, we have

$$\begin{aligned} sc_p(\mathbf{s}) + sc_p(\mathbf{s}') &< (n+2)sc_p(\mathbf{s}_p^*) \\ &\Leftarrow 2sc_q(\mathbf{s}) + \sum_{p_i = \{F_k\}, k \in \{1,2\}} \beta_i < nsc_p(\mathbf{s}_p^*) + 2sc_p(\mathbf{s}_p^*) \\ &\Leftarrow \sum_{p_i = \{F_k\}, k \in \{1,2\}} d < nsc_p(\mathbf{s}_p^*) \end{aligned}$$

Note that $\sum_{p_i = \{F_k\}, k \in \{1,2\}} d \leq nd$, we have the inequality (1) proved if $sc_p(\mathbf{s}_p^*) \geq d$. Therefore we will focus on the case when $sc_p(\mathbf{s}_p^*) < d$.

Given the two sets of agents \mathcal{L} and \mathcal{R} , we denote $\mathcal{L}_k = \{i | p_i = \{F_k\}\} \cap \mathcal{L}$ and $\mathcal{R}_k = \{i | p_i = \{F_k\}\} \cap \mathcal{R}$, $k \in \{1, 2\}$. We also

denote $\mathcal{L}^- = \{i | x_i \leq x_l\}$ and $\mathcal{R}^+ = \{i | x_i \geq x_r\}$. It is clear that $|\mathcal{L}^-| \geq 1$ and $|\mathcal{R}^+| \geq 1$.

We note that agent l and agent r must be of different types, otherwise we have $sc_p(\mathbf{s}_p^*) \geq d$. Without loss of generality, for profile \mathbf{p} , we assume l is F_1 -typed and r is F_2 -typed in \mathbf{s}_p^* . Then all agents in \mathcal{L}^- must be F_1 -typed and all agents in \mathcal{R}^+ must be F_2 -typed, otherwise we have $sc_p(\mathbf{s}_p^*) \geq d$. According to the number of agents in \mathcal{R}_1 and \mathcal{L}_2 , we have the following 4 cases:

Case 1: $|\mathcal{R}_1| = 0$ and $|\mathcal{L}_2| = 0$.

In this case, we have all agents in set \mathcal{L} being F_1 -typed and all agents in \mathcal{R} being F_2 -typed. We can see that solution \mathbf{s} is already the optimal solution.

Case 2: $|\mathcal{R}_1| > 0$ and $|\mathcal{L}_2| = 0$.

In this case, for any agent $u \in \mathcal{R}_1$, since agent l and u are both F_1 -typed, we have $sc_p(\mathbf{s}_p^*) \geq |x_u - x_l|$, i.e., $cost_u(\mathbf{s}) \leq sc_p(\mathbf{s}_p^*)$. Meanwhile, we note that $\mathcal{R}_1 \cap \mathcal{R}^+ = \emptyset$, otherwise we have $sc_p(\mathbf{s}_p^*) \geq d$. As a result, we have $|\mathcal{R}_1| \leq n/2$. Therefore we have the following inequality:

$$sc_p(\mathbf{s}) \leq sc_q(\mathbf{s}) + \sum_{u \in \mathcal{R}_1} cost_u(\mathbf{s}) \leq sc_q(\mathbf{s}) + (n/2)sc_p(\mathbf{s}_p^*)$$

Note that $sc_q(\mathbf{s}) \leq sc_p(\mathbf{s}_p^*)$. Therefore, we have $sc_p(\mathbf{s}) \leq (n/2 + 1)sc_p(\mathbf{s}_p^*)$.

Case 3: $|\mathcal{R}_1| = 0$ and $|\mathcal{L}_2| > 0$.

For this case, we have the same result as Case 2 by symmetry.

Case 4: $|\mathcal{R}_1| > 0$ and $|\mathcal{L}_2| > 0$.

In this case, consider agent l and an arbitrary agent in \mathcal{R}_1 , it is clear that the sum of their cost is at least $d/2$, same for agent r and an arbitrary agent in \mathcal{L}_1 . Therefore we have $sc_p(\mathbf{s}) \geq d/2 + d/2 = d$, which contradicts our setting.

Therefore, we have $sc_p(\mathbf{s}) \leq (n/2 + 1)sc_p(\mathbf{s}_p^*)$ and the theorem is proved. \square

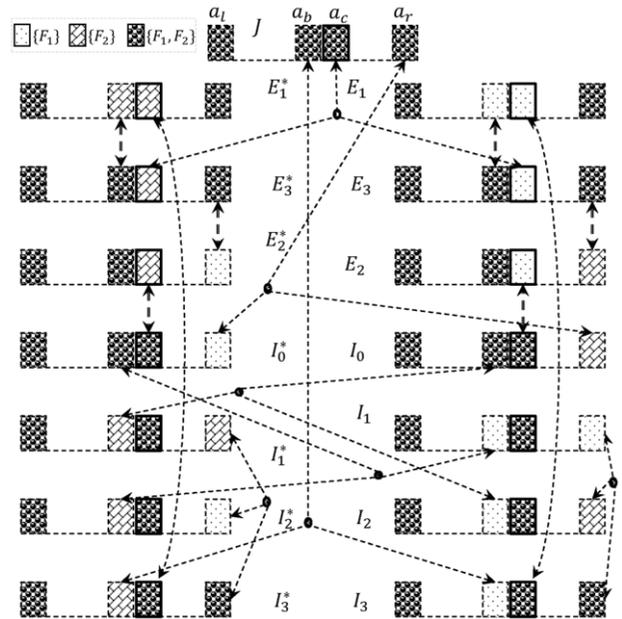


Figure 2. Instance used for proving the lower bound of 2.

Lower bound

In this section, we show that to minimize sum cost, the limit of approximation ratio approaches 2 when the number of agents n approaches infinity.

Theorem 6 Given $n + 2, n \geq 3$ agents, for any $\tau \rightarrow 0$, there exists no α -approximation deterministic strategy-proof mechanism for the facility location problem with $\alpha < 2 - 1/(n + 1) - \tau/n^2$.

To prove Theorem 6, we will first construct a set of profiles and introduce some lemmas. Let $x_l = -n^2, x_b = -\tau, x_c = 0, x_r = n^2 - 1$ be four locations. We locate one agent at each location and refer to these four agents as l, b, c, r respectively. As we have $n + 2$ agents in total, the remaining $n - 2$ agents will be located at x_c .

Starting from profile J where all agents have preference $\{F_1, F_2\}$, we construct the following set of profiles by changing the preferences of agent l, b, c and r . Note that only the preference of these four agents are changed, and the preference of remaining $n - 2$ agents at x_c will always be $\{F_1, F_2\}$ in these profiles:

$$I_0 = J[p_r \rightarrow \{F_2\}]$$

$$I_3 = J[p_b \rightarrow \{F_1\}], I_2 = I_3[p_r \rightarrow \{F_2\}], I_1 = I_3[p_r \rightarrow \{F_1\}]$$

$$E_3 = J[p_c \rightarrow \{F_1\}], E_2 = E_3[p_r \rightarrow \{F_2\}], E_1 = E_3[p_b \rightarrow \{F_1\}]$$

where I_0 is the profile when agent r in J has preference $\{F_2\}$ instead of $\{F_1, F_2\}$. Similar for other profiles. Meanwhile, in the following part, we will denote the lying of an agent in a similar way.

$I_a \xrightarrow{p_i \rightarrow \{F_k\}} I_b$ means agent i in profile I_a lies to have preference for F_k to change the profile to I_b . As agents are selfish, we will say this lying is prevented if $cost_i(I_b) \geq cost_i(I_a)$, otherwise we will say this lying is not prevented.

On the other hand, by exchanging the preference of agents with preference $\{F_1\}$ and agents with preference $\{F_2\}$ in profile I , we get the corresponding symmetric profile I^* , which is depicted in Figure 2. The analysis and results for these profiles will be similar by symmetry.

Let $\sigma = 2 - 1/(n + 1) - \tau/n^2$. For the sake of contradiction, we assume that there exists an α -approximation deterministic strategy-proof mechanism M with $\alpha < \sigma$. We derive the available ranges to put the facilities for the above profiles except profile J under this assumption with lemmas. Then we prove Theorem 6 by showing that given the available ranges derived there is no way to maintain the assumption for profile J no matter where to put the facilities.

One can verify that the optimal sum cost for profile $J, I_0, I_1, I_3, E_1, E_3$ is $opt_1 = x_r + \tau$ and the optimal sum cost for profile I_2, E_2 is $opt_2 = -x_l + \tau$. As a result, the following lemma easily follows:

Lemma 7 For any profile $I, I \in \{J, I_0, I_1, I_2, I_3, E_1, E_2, E_3\}$, no solution can have sum cost more than $x_r - x_l - \tau$.

Proof. Any feasible solution with sum cost at least $x_r - x_l - \tau$ makes $\alpha \geq \sigma$, since $\sigma \cdot \max(opt_1, opt_2) = \sigma(n^2 + \tau) < x_r - x_l - \tau$. \square

Let $y_1(I)$ and $y_2(I)$ be the building locations of F_1 and F_2 for profile I . Based on Lemma 7, we have the following lemma.

Lemma 8 For any profile $I, I \in \{I_1, I_2, I_3, E_1, E_2, E_3\}$, we have $d(c, y_1(I)) < d(c, y_2(I))$.

Proof. Given the profile structure, note that there is at least one agent with preference $\{F_1\}$ located at x_b or x_c and $n - 2$ agents with preference $\{F_1, F_2\}$ located at x_c . If $d(c, y_1(I)) > d(c, y_2(I))$ or

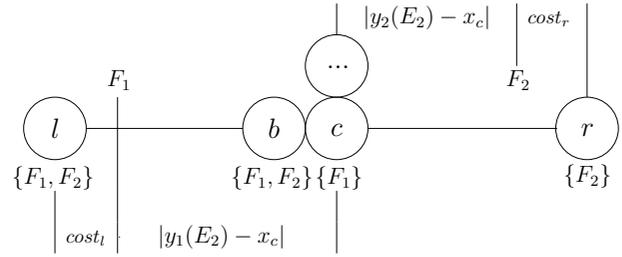


Figure 3. The instance of profile E_2 with $d(c, y_1(E_2)) > d(c, y_2(E_2))$.

$d(c, y_1(I)) = d(c, y_2(I))$, the $n - 2$ agents located at x_c will be F_2 -typed or can be regarded as F_2 -typed. Therefore, for profile E_2 depicted in Figure 3, we can see the total cost for all agents located at x_c is already at least $|y_1(I) - x_c| + |y_2(I) - x_c|$. Combined with the cost for agent l and agent r , we have the sum cost sc at least $x_r - x_l$, same for profile E_1 and E_3 . For profile I_1, I_2, I_3 , we have the sum cost sc at least $x_r - x_l - \tau$ for similar reasons. Therefore in order to maintain $\alpha < \sigma$, we must have $d(c, y_1(I)) < d(c, y_2(I))$. \square

Let us now define $h_l = -n, h_r = n^2/(n + 1) - 1, h'_r = n$, which will be part of the expressions of available ranges in the following part. We have $h_l - x_l \geq (\sigma - 1) \cdot opt_1$ and $x_r - h_r \geq (\sigma - 1) \cdot opt_2$.

Lemma 9 For any profile $I, I \in \{I_2, E_2\}$, we have $y_1(I) < y_2(I)$ and $y_2(I) \in (h_r, 2x_r - h_r)$.

Proof. Note that in I_2 and E_2 , l has preference $\{F_1, F_2\}$ and r has preference $\{F_2\}$. Therefore, the total cost of agent l and agent r is already $x_r - x_l$ if $y_1(I) \geq y_2(I)$. Combined with Lemma 8, all agents except agent r will be F_1 -typed and their total cost is at least opt_2 . Therefore, to keep $\alpha < \sigma$, the cost of agent r cannot reach $(\sigma - 1) \cdot opt_2$, i.e., $|y_2(I) - x_r| < (\sigma - 1) \cdot opt_2$. Since $(\sigma - 1) \cdot opt_2 \leq x_r - h_r$, we have $y_2(I) \in (h_r, 2x_r - h_r)$. \square

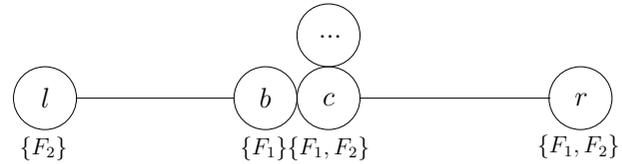


Figure 4. The instance of profile I'_3 .

Lemma 10 For any profile $I, I \in \{I_1, I_3, E_1, E_3\}$, we have $y_1(I) \in (h_r, h'_r)$ and $y_2(I) \in (2x_l - h_l, h_l)$.

Proof. We will prove the lemma for profile I_3 as the analysis for the other profiles is similar. Let profile $I'_3 = I_3[p_l \rightarrow \{F_2\}]$ depicted in Figure 4. As it is symmetric in preference with profile E_2 , we can see Lemma 9 is also applicable to I'_3 in a symmetric way, i.e., $y_2(I'_3) \in (2x_l - h_l, h_l)$. Therefore, in order to prevent lying $I_3 \xrightarrow{p_l \rightarrow \{F_2\}} I'_3$, one facility must be located in $(2x_l - h_l, h_l)$. By Lemma 9, to prevent lying $I_3 \xrightarrow{p_r \rightarrow \{F_2\}} I_2$, one facility must be located in $(h_r, 2x_r - h_r)$.

Moreover, if facility F_1 is located in $(2x_l - h_l, h_l)$, since F_1 is closer to c by Lemma 8, the sum cost is already at least $|h_l - x_l| +$

⁶ Note that the ranges may be beyond x_l and x_r .

$|h_l - x_b| + (n - 1)|h_l - x_c| = (x_c - x_l - \tau) + (n - 1)(x_c - h_l) > \sigma \cdot opt_1$. Therefore F_1 must be located in $(h_r, 2x_r - h_r)$. Meanwhile, we also have $y_1(I_3) < h'_r$, otherwise the sum cost is already at least $|h'_r - x_b| + n|h'_r - x_c| + |h'_r - x_r| = (x_r - x_c + \tau) + n(h'_r - x_c) > \sigma \cdot opt_1$.

Therefore, we have $y_1(I) \in (h_r, h'_r)$ and $y_2(I) \in (2x_l - h_l, h_l)$. \square

Based on Lemma 10, we will make some extension to Lemma 9.

Lemma 11 For any profile I , $I \in \{I_2, E_2\}$, we have $y_1(I) \in (-h_r, h_r - 2\tau)$.

Proof. We will prove the lemma for profile I_2 as the analysis for profile E_2 is similar. According to Lemma 10, in order to prevent lying $I_3 \xrightarrow{p_r \rightarrow \{F_2\}} I_2$, the cost of agent r must be at least $x_r - h'_r$. As a result, if $y_1(I_2) \notin (-h_r, h_r - 2\tau)$, the sum cost will be at least $(x_c - x_l - \tau) + (n - 1)h_r + (x_r - h'_r) > \sigma \cdot opt_2$, which contradicts our assumption.

Therefore, we have $y_1(I) \in (-h_r, h_r - 2\tau)$. \square

Lemma 12 For profile I_0 , one facility must be located in $(-h_r, h_r - 2\tau)$

Proof. By Lemma 11, lying $I_0 \xrightarrow{p_b \rightarrow \{F_1\}} I_2$ is not prevented if no facility is located in $(-h_r, h_r - 2\tau)$. \square

With the above lemmas, we will focus on profile J and show the proof for Theorem 6 below.

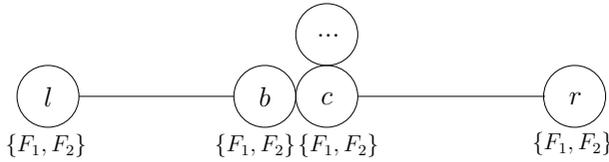


Figure 5. The instance of profile J .

Proof. Without loss of generality, we assume $y_1(J) \leq y_2(J)$ since the preference of each agent is $\{F_1, F_2\}$ and switching the positions of F_1 and F_2 does not affect the sum cost. First we note that by Lemma 10, the cost of agent b is at least $h_r + \tau$ in I_3 . Therefore in order to prevent lying $I_3 \xrightarrow{p_b \rightarrow \{F_1, F_2\}} J$ and $I_3^* \xrightarrow{p_b \rightarrow \{F_1, F_2\}} J$, neither F_1 nor F_2 can be located within $(x_b - (h_r + \tau), x_b + (h_r + \tau))$, which is $(-h_r - 2\tau, h_r)$ for profile J . According to the relative locations of the facilities and agent c , we have the following 4 cases:

Case 1) For J , if both F_1 and F_2 are on the left side of x_c , the cost of agent r is at least $x_r + h_r + 2\tau$. Note that for I_0 we have one facility located in $(-h_r, h_r - 2\tau)$ by Lemma 12. As a result, $J \xrightarrow{p_r \rightarrow \{F_2\}} I_0$ can not be prevented. Therefore it is not strategy-proof which contradicts our assumption.

Case 2) For J , if both F_1 and F_2 are on the right side of x_c , the sum cost is at least $|h_r - x_l| + |h_r - x_b| + (n - 1)|h_r - x_c| = (x_c - x_l + \tau) + (n + 1)h_r > \sigma \cdot opt_1$, which makes $\alpha > \sigma$ and contradicts our assumption.

Case 3) For J , if F_1 and F_2 are on different sides of x_c and $d(c, y_1(J)) \leq d(c, y_2(J))$, combined with Lemma 8 and 9, in order to prevent lying between profiles E_1, E_3 and I_3 , we have $y_1(E_3) = y_1(E_1) = y_1(I_3)$. Meanwhile, to prevent lying $I_3 \xrightarrow{p_b \rightarrow \{F_1, F_2\}} J$, there must be $x_b - y_1(J) \geq y_1(I_3) - x_b$. On the other hand, to prevent lying between E_3 and J , there must be $x_c - y_1(J) = y_1(E_3) - x_c$. However,

these two equations lead to $x_b \geq x_c$, which contradicts our setting.

Case 4) For J , if F_1 and F_2 are on different sides of x_c and $d(c, y_1(J)) > d(c, y_2(J))$, there must be $y_2(J) \in [h_r, h'_r)$. Otherwise sum cost is at least $n(h'_r - x_c) + \tau + (x_r - h'_r) > \sigma \cdot opt_1$. Meanwhile, for profile I_0 , we must have $d(r, y_2(I_0)) < d(r, y_1(I_0))$. Otherwise sum cost is at least $x_r - x_l - \tau$ no matter where to put the facilities. Therefore, in order to prevent $J \xrightarrow{p_r \rightarrow \{F_2\}} I_0$ and $I_0 \xrightarrow{p_r \rightarrow \{F_1, F_2\}} J$, there must be $|x_r - y_2(I_0)| = x_r - y_2(J)$. Hence for profile I_0 , facility F_1 must be placed in $(-h_r, h_r - 2\tau)$ by Lemma 12 since F_2 cannot be placed in $(-h_r, h_r - 2\tau)$. As a result, the sum cost is at least $(x_c - x_l + \tau) + (x_r - h'_r) > \sigma \cdot opt_1$, which makes $\alpha > \sigma$ and contradicts our assumption.

Therefore, mechanism M does not exist and the claim is proved. \square

4 OPTIONAL PREFERENCE (MAX)

In this section, we focus on the Max variant of the optional preference model, i.e., $cost_i = \max_{F_k \in p_i} d(i, F_k)$.

4.1 Maximum Cost

For maximum cost objective, we propose a deterministic strategy-proof mechanism which is optimal at the same time.

Mechanism 3 Given a profile, output $y_1 = y_2 = L/2$ if $\{F_1, F_2\} \in \{p_1, p_n\}$ or $p_1 = p_n$. Otherwise if $p_1 = \{F_1\}, p_n = \{F_2\}$, output $y_1 = \frac{1}{2} \max(X_1 \cup X_{1,2}), y_2 = L - \frac{1}{2}(L - \min(X_2 \cup X_{1,2}))$; if $p_1 = \{F_2\}, p_n = \{F_1\}$, output $y_1 = L - \frac{1}{2}(L - \min(X_1 \cup X_{1,2})), y_2 = \frac{1}{2} \max(X_2 \cup X_{1,2})$.

Theorem 13 Mechanism 3 is strategy-proof.

Proof. Let y'_1 and y'_2 denote the corresponding output when an agent lies. Given the mechanism, there are three cases:

Case 1. $\{F_1, F_2\} \in \{p_1, p_n\}$ or $p_1 = p_n$. In this case, only agent 1 and agent n can influence the output by lying. Let us consider agent 1 to be the lying agent as the result for agent n would be the same. If agent 1 has preference $\{F_1, F_2\}$, lying to have preference $\{F_1\}$ will make $y'_2 = L - \frac{1}{2}(L - \min(X_2 \cup X_{1,2})) > L/2 = y_2$ and his cost will increase. Lying to have preference F_2 would have the same result. If agent 1 has preference for F_1 , lying to have preference $\{F_1, F_2\}$ does not change the output. Lying to have preference $\{F_2\}$ will make $y'_1 = L - \frac{1}{2}(L - \min(X_1 \cup X_{1,2})) > L/2 = y_1$ and his cost will increase. If agent 1 has preference $\{F_2\}$, we would have the same result. Therefore agent 1 has no incentive to lie, which means no agent has the incentive to lie.

Case 2. $p_1 = \{F_1\}, p_n = \{F_2\}$. In this case, only agent 1, agent n , the agent at $\max(X_1 \cup X_{1,2})$ and the agent at $\min(X_2 \cup X_{1,2})$ can influence the output by lying. Let us consider agent 1 or the agent at $\max(X_1 \cup X_{1,2})$ be the lying agent. The result for the other two agents would be the same by symmetry. For agent 1, if he lies to have preference $\{F_1, F_2\}$ or $\{F_2\}$, we will have $y'_1 = L/2 > \frac{1}{2} \max(X_1 \cup X_{1,2}) = y_1$ and his cost will increase. For the agent at $\max(X_1 \cup X_{1,2})$, he could only change the output by lying to have preference $\{F_2\}$, in which case we will have a new value of $\max(X_1 \cup X_{1,2})$ (denoted as $\max(X_1 \cup X_{1,2})'$),