JOINT LEARNING AND OPTIMIZATION OF UNKNOWN COMBINATORIAL UTILITY FUNCTIONS

Paolo Campigotto, Andrea Passerini, Roberto Battiti

March 2013

Technical Report # DISI-13-021

Joint learning and optimization of unknown combinatorial utility functions

Paolo Campigotto[a], Andrea Passerini[a], Roberto Battiti[a]

[a]*DISI - Dipartimento di Ingegneria e Scienza dell'Informazione,*
Università degli Studi di Trento,
Via Sommarive 14, I-38123 Povo, TN, Italy.

Abstract

This work considers the problem of automatically discovering the solution preferred by a decision maker (DM). Her preferences are formalized as a combinatorial utility function, but they are not fully defined at the beginning and need to be learnt during the search for the satisficing solution. The initial information is limited to a set of catalog features from which the decisional variables of the DM are to be selected.

An interactive optimization procedure is introduced, which iteratively learns an approximation of the utility function modeling the quality of candidate solutions and uses it to generate novel candidates for the following refinement. The source of learning signals is the decision maker, who is fine-tuning her preferences based on the learning process triggered by the presentation of tentative solutions.

The proposed approach focuses on combinatorial utility functions consisting of a weighted sum of conjunctions of predicates in a certain theory of interest. The learning stage exploits the sparsity-inducing property of 1-norm regularization to learn a combinatorial function from the power set of all possible conjunctions of the predicates up to a certain degree. The optimization stage consists of maximizing the learnt combinatorial utility function to generate novel candidate solutions. The maximization is cast into an Optimization Modulo Theory problem, a recent formalism allowing to efficiently handle both discrete and continuous-valued decisional features. Experiments on realistic problems demonstrate the effectiveness of the method in focusing towards the optimal solution and its ability to recover from suboptimal initial choices.

Keywords:
Preference elicitation, machine learning, combinatorial optimization, Satisfiability, Satisfiability Modulo Theory, Optimization Modulo Theory.

1. Introduction

In real world optimization tasks, a significant portion of the problem-solving effort is usually devoted to specifying in a computable manner the function to be optimized. This modeling work consists of modifying and refining the problem definition on the basis of information elicited from the decision maker (DM). Typically, asking a user to quantify her real objectives *a priori*, without seeing any optimization results, is extremely difficult. *Interactive* decision making approaches handle this initial lack of complete knowledge by keeping the user in the loop of the optimization process. They use the information from the DM during the optimization task to guide the search towards the solution preferred by the user.

A paradigmatic case of incomplete problem definition is provided by multi-objective optimization problems, which consider the simultaneous optimal attainment of a set of conflicting objectives. While the DM usually can define the set of desirable objectives, she cannot define their relative importance, the trade-offs and the proper combination of them into an overall *utility function*. Several interactive multi-objective algorithms have been proposed to

Email addresses: `campigotto@disi.unitn.it` (Paolo Campigotto), `passerini@disi.unitn.it` (Andrea Passerini), `battiti@disi.unitn.it` (Roberto Battiti)

learn the utility function modeling the preferences of the DM, see e.g. [1] for a recent review and [2] for a recent example. They iteratively alternate preference elicitation (decision stage) and solution generation (optimization phase). At each iteration, the DM evaluates the proposed candidate solutions. The preference information obtained is used to refine a model of the DM preferences; new solutions are generated based on the learnt model. By adopting this approach, the DM preferences drive the search process and only a subset of the Pareto optimal solutions needs to be generated and evaluated.

In general, formalizing the user preferences into a mathematical model is not trivial: a model should capture the qualitative notion of preference and represent it as a quantitative function. Let us assume that the candidate solutions of the problem are described by a set of n features $x_1, \ldots, x_n$. The simplest and most used utility model is the *additive* function, where the preference of the DM for the candidate solution $\mathbf{x}$ is given by the sum of sub-utility functions:

$$U(\mathbf{x}) = \sum_{k=1}^{n} u_k(x_k)$$

with each sub-utility function u_k is defined on a single feature x_k. Additive utility models are appropriate under the assumption of preferential independence among the set of features [3]. Preferential independence exists when the DM preference for the values of a feature x_i does not depend on the fixed values of other features x_j, $j \neq i$. Consider, for example, a costumer of a real estate company articulating her preferences by using only two attributes: the number of bedrooms (x_1) and the distance from the city center (x_2). Her preference over the values of x_1 is the same regardless of the values of x_2, and vice versa: she *always* prefers houses with two bedrooms over houses with one bedroom and she *always* favors the location nearest to the city center.

Additive models fail to capture complex DM preferences including non-linear relationships among the features of the candidate solutions. For example, a customer willing to have a gym in the neighborhood in the case of candidate houses near the city center. When considering houses in the suburbs, the presence of free parking makes houses without a garage attractive (to keep price low) and, similarly, the proximity of green areas providing the opportunity for outdoor sport activities decreases her interest for a gym in the neighborhood. *Generalized additive independence* (GAI) [3] models overcome the limitation of simple additive models by encoding utilities as a sum of sub-utility "overlapping" functions:

$$U(\mathbf{x}) = \sum_{k=1}^{p} u_k(X_k)$$

where $X_k, k = 1 \ldots p$, are subsets of the n features that may be *non*-disjoint.

The papers [4, 5, 6] present a framework for the elicitation of GAI utilities based on the minimax regret decision criterion. The minimax regret criterion guarantees worst-case bounds on the quality of the decision made under uncertain knowledge of the DM preferences. In particular, the minimax regret optimal decision minimizes the worst case loss with respect to the possible realizations of the unknown DM utility function.

Recent work in the field of constraint programming [7] formalizes the user preferences in terms of *soft constraints*. In soft constraints, a generalization of hard constraints, each assignment to the variables of one constraint is associated with a preference value. A candidate solution to a set of soft constraints assigns a value to the variables of each constraint. The desirability of a candidate solution is computed by combining the preference values of its assignments to the variables of the different constraints. The work in [7] introduces an elicitation strategy for soft constraint problems with missing preferences, to find the solution preferred by the decision maker by asking the final user to reveal as few preferences as possible.

In this paper, soft constraints are cast into *first-order logic formulas*, with each formula being the conjunction of predicates in a certain theory of interest. The theory fixes the interpretation of the symbols used in the predicates (e.g., the theory of arithmetic for dealing with integer or real numbers). The DM preferences are represented by combinatorial utility functions which are weighted combinations of the first-order logic formulas. For example, consider the case of flight selection. The predicate $x_1 + x_2 \leq 5$ *hours* defines the preference for a travel duration, calculated as flight duration (x_1) plus transfer time to the departure airport (x_2), smaller than five hours. The predicate $x_3 < 2$ states the desirability for a flight with number of stopovers (x_3) smaller than two. The weighted combination of the couple of predicates articulates the DM preferences about the flight to book. Similarly to the GAI models, this representation of the DM preferences can express the combined effects of multiple non-linearly related decisional features. Our preference elicitation algorithm formulates the combinatorial utility functions as *Optimization Modulo Theory* instances where the objective function is unknown and has to be interactively learnt. *Optimization Modulo Theory* [8, 9] is an extension of *Satisfiability Modulo Theory* [10] (SMT), a powerful formalism to check the satisfiability of formulas in a decidable first-order theory. SMT has received increasing attention in recent years, thanks to a number of successful applications in areas like verification systems, planning and model checking. *Optimization Modulo Theory* extends SMT by considering optimization (MAX-SMT) problems. Rather than checking for the existence of a satisfying assignment as in SMT, the target is a satisfying assignment that minimizes a given cost function. The adoption of the Optimization Modulo Theory formalism enables a generic approach which efficiently handles both discrete and continuous-valued decisional features.

Our algorithm for the joint learning and optimization of the DM preferences consists of an iterative procedure alternating a search phase with a model refinement phase. At each step, the current approximation of the utility function is used to guide the search for optimal configurations; preference information is required for a subset of the recovered candidates, and the utility model is refined according to the feedback received. A set of randomly generated configurations is employed to initialize the utility model at the first iteration.

Unlike previous work on preference elicitation for constraint problems [7], our method does not assume to know in advance the decisional features of the user and their detailed combination. In the paper [7] soft constraint topology and structure is assumed to be known and the incomplete information consists of missing local preference values only. The initial amount of knowledge required by our approach is limited to a set of "catalog" features from which the decisional variables of the DM are selected. The limited initial knowledge translates in a *sparse* unknown utility function: only a fraction of the catalog features represent the decisional items of the DM and only a subset of the possible terms constructed from them defines the DM utility function. Furthermore, our method can handle uncertain, inconsistent and contradictory preference information from the final user, which characterizes many human decision processes. The robustness to noisy feedback from the DM distinguishes our approach also from the regret-based preference elicitation methods [4, 5, 6], where the bounds on the quality of the recommended solutions and the guarantee about the convergence to provably-optimal configurations are valid under the unrealistic assumption of noise-free and certain answers from the DM.

This paper introduces the first approach combining learning, interactive optimization and SMT. A preliminary version of our technique was presented in [11]. This manuscript considerably extends the preliminary version by a deeper comparison with the preference elicitation literature, a more detailed description of the Satisfiability Modulo Theory formalism and a wider experimental evaluation including also an additional benchmark designed in the spirit of real-world applications. Furthermore, the quantitative judgments asked to the DM in the preliminary version are replaced by less cognitive demanding queries, consisting of qualitative comparisons of candidate solutions.

The organization of the paper is as follows. Section 2 introduces our preference elicitation algorithm, while Section 3 analyzes its properties and its parameters setting. Section 4 introduces SMT and Optimization Modulo Theory and explains how this formalism is used by our algorithm. Related work is discussed in Section 5. Experimental results in Section 6 on both synthetic and realistic problems demonstrate the effectiveness of our approach in focusing towards the optimal solutions, its robustness and its ability to recover from suboptimal initial choices. A discussion including potential research directions concludes the paper.

2. Overview of our approach

For ease of explanation, we first introduce the simplest formulation of our preference elicitation method, which considers Boolean decisional features only. The generalization to discrete and continuous-valued features is presented in Section 4. In the simplest formulation of our algorithm, candidate configurations are n dimensional vectors $\mathbf{x}$ consisting of Boolean *catalog* features. *A priori* knowledge of the problem is limited to the set of catalog features. The unknown combinatorial utility function expressing the DM preferences is the weighted combination of Boolean terms generated from the catalog features (weighted MAX-SAT instance), and it has to be jointly and interactively learned during the optimization process. Furthermore, the optimal utility function is complex enough to prevent exhaustive enumeration of possible solutions. The only assumption we make on the utility function is its sparsity, both in the number of features (from the whole set of catalog ones) and in the number of terms constructed from them. The assumption is grounded on the bounded rationality of a human DM, which can simultaneously handle only a limited number of features to make decisions, and drives some specific design choices of our optimization algorithm.

```
procedure interactive_sparse_optimization
    input: set of the catalog variables
    output: learnt utility function and configuration optimizing it
    /* Initialization phase */
    Select $s$ configurations uniformly at random;
    Ask the DM for the ranking of the $s$ configurations;
    initialize training set $D$ by the ranking of the $s$ configurations;
    /* Refinement phase */
    while (termination_criterion)
        /* Utility function learning phase */
        Based on $D$, select terms and relative weights for current
        weighted MAX-SAT formulation (Eq. 2);
        /* Optimization phase */
        Get $s/2$ new configurations by optimizing current weighted MAX-SAT
        formulation;
        Ask the DM for the ranking of the new $s/2$ configurations and add it to $D$;
    return configuration optimizing the learnt weighted MAX-SAT formulation
```

Figure 1: Pseudocode for the interactive optimization algorithm. The parameter s defines the number of examples to be compared by the DM at the different iterations. The training set D contains the partial rankings of candidate solutions generated at the initialization phase and at the different iterations of the algorithm.

See Sections 3.5 and 7 for a discussion on how to generalize the approach to arbitrarily complex and dense utility functions, as could be found in optimizing machine rather than human decision processes.

Our method consists of an iterative procedure alternating a utility function learning phase with a search phase. At each step, the current approximation of the utility function, represented as a weighted MAX-SAT instance, is used to guide the search for optimal configurations. A subset of candidate configurations is obtained by solving the weighted MAX-SAT instance (*search phase*). Preference information is required for these candidates, and the utility model is refined according to the feedback received (*learning phase*). In particular, the DM expresses her preferences by ranking the candidate solutions generated at each iteration of our method. Thus the preference information collected at the n-*th* iteration consists of the partial rankings generated at the iteration $1, 2, \ldots, n-1$. A set of randomly generated configurations to be ranked is employed to initialize the utility model. The pseudocode of our algorithm is in Fig. 1. The initialization, learning and search phases of our approach are detailed below, while its properties and the parameters setting are discussed in Sec. 3.

2.1. Initializing the algorithm

A set of random configurations is generated to approximate the DM utility function at the first iteration. Each Boolean feature is assigned a truth value independently and uniformly at random. The evaluation of diverse examples stimulates the preference expression, especially when the user is still uncertain about her final preference [12]. In particular, the diversity of the examples helps the user to reveal the hidden preferences: in many cases the decision maker is not aware of all preferences until she sees them violated. For example, a user does not usually think about the preference for an intermediate airport until one solution suggests an airplane change in a place she dislikes [12].

2.2. Learning an approximation of the utility function

The refinement of the utility model consists of learning the weights of the terms, discarding the terms with zero weight. It includes both the selection of the relevant features from the catalog set and the learning of their detailed combination from the space of all possible conjunction up to certain degree d. To identify sparse solutions, charaterized by few terms with non-zero weights, we adopt 1-norm regularization, which provides an embedded feature selection capability [13]. Features selection is crucial to maximize the learning accuracy with data sets characterized by redundant and irrelevant features [14]. The "*Least absolute shrinkage and selection operator*" (Lasso) [13] is a popular method for regression tasks which uses 1-norm regularization to achieve a sparse solution. Let $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1 \ldots m}$ be the set of m training examples, where $\mathbf{x}_i$ contains the feature values of the i-*th* example and y_i represents its score value. The loss+penalty formulation of Lasso regression is defined by the following optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}} \sum_{i=1}^{m}(y_i - \mathbf{w}^T \cdot \Phi(\mathbf{x}_i))^2 + \lambda\|\mathbf{w}\|_1 \tag{1}$$

where $\mathbf{w}$ is the weight vector and the mapping function Φ projects the input vector to a higher dimensional space. The square loss function $\sum_{i=1}^{m}(y_i - \mathbf{w}^T \cdot \Phi(\mathbf{x}_i))^2$ measures the empirical risk as the sum of the squared training errors. The penalty function consists of 1-norm regularization, favouring automated feature selection. The regularization parameter λ trade-offs the penalty and loss terms.

In our context, the mapping function Φ projects sample vectors to the space of all possible conjunctions up to d Boolean variables. The score value y_i represents the quantitative evaluation of the decision maker for the solution $\mathbf{x}_i$. The preference scores are taken from a predefined ordered set expressing the desirability levels for the candidate solutions. This was the approach adopted in our original formulation [11]. However, comparing solutions is a much more affordable task for the DM than assigning preference scores. We thus cast the task of learning the utility function into a ranking problem.

Given a set of m candidate solutions $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ and an order relation $\succ$ such that $\mathbf{x}_i \succ \mathbf{x}_j$ if $\mathbf{x}_i$ should be preferred to $\mathbf{x}_j$, a *ranking* $(y_1, \ldots, y_m)$ of the m solutions can be specified as a permutation of the first m natural numbers such that $y_i < y_j$ if $\mathbf{x}_i \succ \mathbf{x}_j$. Learning to rank consists of learning a ranking function r from a dataset $\mathcal{D} = \{(\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_{m_i}^{(i)}), (y_1^{(i)}, \ldots, y_{m_i}^{(i)})\}_{i=1}^{s}$, made of s sets with their desired rankings. The ranking function r associates each candidate instance with a real number, with the aim that $r(\mathbf{x}_i) > r(\mathbf{x}_j) \iff \mathbf{x}_i \succ \mathbf{x}_j$. In this way, it provides an ordering that agrees with the observed training examples. Given the ranking dataset $\mathcal{D}$, the 1-norm regularization formulation can be adapted to learn the ranking function r by imposing constraints on the correct pairwise ordering of solutions within a set: $r(\mathbf{x}_h^{(i)}) > r(\mathbf{x}_k^{(i)}) \iff y_h^{(i)} < y_k^{(i)}$. In the case of support vector machines, $r(\mathbf{x}) = \mathbf{w}^T \cdot \Phi(\mathbf{x})$ and $r(\mathbf{x}_i) > r(\mathbf{x}_j) \iff \mathbf{w}^T \cdot (\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)) > 0$. The resulting loss+penalty formulation for 1-norm support vector machine can be written as:

$$\min_{\mathbf{w}\in\mathbb{R}} \sum_{i=1}^{s} \sum_{h,k,y_h^{(i)}<y_k^{(i)}} [1 - \mathbf{w}^T \cdot (\Phi(\mathbf{x}_h^{(i)}) - \Phi(\mathbf{x}_k^{(i)}))]_+ + \lambda\|\mathbf{w}\|_1 \tag{2}$$

where subscript "+" indicates the positive part. The first term is the so-called empirical Hinge Loss, adapted for ranking tasks. It assigns a linear penalty to inconsistencies in the ranking, i.e., cases where a less preferred solution is ranked higher than a more preferred one, or correct rankings where the score difference is smaller than the support vector machine margin, i.e., $0 < \mathbf{w}^T \cdot (\Phi(\mathbf{x}_h^{(i)}) - \Phi(\mathbf{x}_k^{(i)})) < 1$. The formulation in Eq. (2) handles ties and partial rankings, as constraints are only included whenever two examples should be ranked differently.

The learnt function $\hat{f}(\mathbf{x}) = \mathbf{w}^T \cdot \Phi(\mathbf{x})$, where $\mathbf{w}$ is the solution of the ranking problem, will be used as the novel approximation of the utility function f of the DM.

2.3. *Optimizing the learnt utility function*

The learnt utility function $\hat{f}$ expressing the approximation of the DM preferences is represented as a weighted MAX-SAT instance. The set of novel candidate solutions to be ranked by the DM is obtained by applying a complete solver over the weighted MAX-SAT instance. The size of the weighted MAX-SAT instance is indeed bounded by the limited cognitive capabilities of the human DM. The adoption of local search techniques for large scale problems will be discussed in Sec. 7. The MAX-SAT solver returns an optimizer for the input problem, i.e., the configuration $\mathbf{x}^*$ maximizing the weighted sum of the terms representing $\hat{f}$. However, the currently learnt utility $\hat{f}$ is only an approximation of the true unknown DM utility. In order to refine this approximation, a *diversified set* of *informative* training examples is needed. In our algorithm, the creation of informative training examples (*active learning*) is driven by the following principles:

1. the generation of top-quality configurations, consistent with the learnt DM preferences;
2. the generation of diversified configurations, i.e., alternative possibly suboptimal configurations with respect to the learnt utility $\hat{f}$;

3. the search for the DM features which were not recovered by the current approximation $\hat{f}$, i.e., features not appearing in any of the terms in $\hat{f}$.

The rationale for the first principle is the modeling of the relevant areas of the fitness surface (i.e., the shape of the utility function locally guiding the search to the correct direction) and the search for solutions satisficing the DM. As a matter of fact, a preference elicitation system that asks to rank low quality configurations will be likely considered useless or annoying by the user [15]. The second principle favours the exploration of the relationships among the features recovered by the current preference model $\hat{f}$. Finally, as the learnt formulation of $\hat{f}$ may miss some of the DM decisional features, their search is promoted by the third principle. Let us note that the need for a set of good, given the current known preferences, and diverse configurations to be evaluated by the user is pointed out, e.g., in [12].

Based on the above principles, our active learning strategy works as follows. In order to obtain multiple solutions in addition to $\mathbf{x}^*$, the MAX-SAT formulation of $\hat{f}$ is modified by including the *hard* constraint generated by the disjunction of all the terms of $\hat{f}$ unsatisfied by $\mathbf{x}^*$. Unlike the weighted terms, which may or may not be satisfied, *hard* constraints do not have a weight value and have to be satisfied. For example, let t_1 and t_5 be the terms of $\hat{f}$ unsatisfied by $\mathbf{x}^*$, then the hard constraint becomes:

$$(t_1 \vee t_5) \tag{3}$$

If the term t_i is associated with a weight $w_i < 0$, it is satisfied when its value is false. Therefore, if, e.g., the weight w_1 is negative, the hard constraint in Eq. (3) becomes $(\neg t_1 \vee t_5)$. If $\mathbf{x}^*$ satisfies all the terms of $\hat{f}$, i.e., $\hat{f}(\mathbf{x}^*) = 0$, the additional *hard* constraint generated is:

$$(\neg x_1^* \vee \neg x_2^* \ldots \vee \neg x_n^*)$$

which excludes $\mathbf{x}^*$ from the set of feasible solutions. The inclusion of the hard constraints is motivated by the second principle, as the satisfaction of at least one of the terms unsatisfied by $\mathbf{x}^*$ is usually obtained by configurations differing in more than a single Boolean value from $\mathbf{x}^*$. The modified formulation of $\hat{f}$ is then optimized (first principle) to obtain the the second candidate solution $\mathbf{x}^{**}$. The catalog features not included in the formulation of $\hat{f}$ are set uniformly at random in $\mathbf{x}^{**}$ (third principle). Let us note that if these features are truly irrelevant for the DM, setting them at random should not affect the evaluation of the candidate solutions. If on the other hand some of them are needed to explain the DM preferences, driving their elicitation can allow to identify the deficiencies of the current approximation $\hat{f}$ and recover previously discarded relevant features.

The process is repeated, progressively adding hard constraints for each of the previously generated solutions, until the desired number of configurations have been generated or the hard constraints made the MAX-SAT instance unsatisfiable. Finally, the DM will rank the new candidate solutions based on her preferences and the ranking information will be included in the training examples set D for the following refinements of $\hat{f}$.

3. Analysis of our approach

The number of refinement iterations does not need to be fixed at the beginning. The DM may ask for an additional iteration by comparing the recovered candidates with her own preferences. The termination criterion is thus represented by the satisfaction of the DM with the presented candidate solutions. Furthermore, the number of candidates to be evaluated at each iteration is arbitrary. In the settings used here, we use s configurations at the first iteration and $s/2$ examples at each following iteration. A larger number of configurations is suggested at the first iteration to stimulate the preference expression of the DM, as discussed in Sec. 2.1, and to generate a good initial model. The following iterations generate solutions distributed in the promising regions of the search space. The goal of our approach is indeed the identification of the solution preferred by the user (*learning to optimize*) rather than an accurate global approximation of the DM utility function (*learning per se*). This requires a shift of paradigm with respect to standard machine learning strategies, in order to model the relevant areas of the optimization fitness surface rather than reconstruct it entirely.

3.1. Convergence of the algorithm

The limited size of the MAX-SAT instances (or MAX-SMT, as it will be explained in Sec. 4) enables the systematic investigation of the search space by means of a complete solver (the adoption of local search for larger instances is discussed in Sec. 7), which ensures the identification of the solution $\mathbf{x}^*$ maximizing the learnt utility model $\hat{f}$

(completeness property). However, our algorithm cannot guarantee the quality of the model $\hat{f}$ approximating the true DM utilities, and therefore the optimality of $\mathbf{x}^*$ (or bounds on its quality) w.r.t. the true DM utilities cannot be proved. As a matter of fact, the learning task in Eq. (2) is convex, and thus guaranteed to converge to its global optimum, but the consistency of the learning algorithm with the true underlying user utility is only guaranteed asymptotically (i.e., provided that enough training data is available). On the other hand, our algorithm does not need to learn the exact form of the DM utility function. The goal of our approach is indeed to elicit as few preference information from the DM as possible in order to identify her favourite solution (*learning to optimize*). For example, consider the toy DM utility function represented by the negation of a single ternary term: $\neg(x_1 \wedge x_2 \wedge x_3)$. The approximation of the DM utility function consisting, e.g., of the formula $\neg x_1$ is sufficient to find the favourite DM solution. More in general, only the shape of the utility function locally guiding the search to the correct direction is actually needed. Indeed the experimental results reported in Sec. 6 show the ability of our method in identifying the optimal solution and the improvements in the quality of the candidate solutions when increasing the number of refinement iterations (anytime property).

3.2. User cognitive load

In preference elicitation systems the acquisition of feedback from a human DM, characterized by limited patience and bounded rationality, is a crucial issue. Although providing explicit weights and mathematical formulas is in general prohibitive for the DM, she can definitely evaluate the returned solutions. In the preliminary version of our algorithm [11] quantitative judgments are asked to the DM. However, asking the final user for precise scores is in many cases inappropriate or even impossible. Most of the users are typically more confident in comparing solutions, providing qualitative judgments like “I prefer solution $\mathbf{x}'$ to solution $\mathbf{x}''$ ”, rather than in specifying how much they prefer $\mathbf{x}'$ over $\mathbf{x}''$. In order to reduce the embarrassment of the decision maker when specifying precise preference scores, in this paper the evaluation by the user consists of just comparing and ranking candidate solutions.

3.3. Robustness to inaccurate preference information

Assuming that the user provides accurate and consistent preference information at any time is not realistic, due to the bounded rationality and the limited capabilities of the humans when making decisions. Different factors may generate uncertain and inconsistent feedback from the DM, including occasional inattentions, embarrassment when comparing very similar solutions or solutions which are very different from her favourite one, DM fatigue increasing with the number of queries answered. The adoption of regularized machine learning strategies in our algorithm enables a robust approach that can handle ties and inaccurate rankings from the DM.

3.4. Computational complexity

The cognitive capabilities of the humans when making decisions limit the number n of catalog features and the size d of the Boolean terms. Therefore, the learning phase (problem (2)) is accomplished in a negligible amount of time (w.r.t. the user response time). Analogous observation holds for the computational effort required by the search phase. Proposing a query consists of generating c candidates to be ranked. Each candidate is obtained by a run of the complete MAX-SAT solver. Even if the search space size is 2^n, the bounded value of n and the efficient performance of modern SAT solvers, that can manage problems with thousands of variables and millions of clauses, enable the completion of the search phase in a negligible amount of time. An efficient implementation of the search phase is also achieved in the more general case represented by MAX-SMT instances, that will be discussed in Sec. 4, as the state-of-the-art SMT solvers are built on top of modern SAT solvers. Finally, the estimation of the cognitive load of the DM is $O(c \log c)$, with c being the number of candidates to be ranked. In the experiments reported in Sec. 6, the value of c used in the refinement phase varies from 5 to 50. Keeping c low limits the computational effort of the DM. As a matter of fact, the DM typically prefers to rank small batches of good quality solutions rather than a single large batch of candidates, including also lower quality ones. Furthermore, the chance of obtaining inaccurate feedback from the DM increases when she evaluates solutions which are very different from her favourite one. Therefore, a rule of thumb for the algorithm configuration consists of keeping the value of c low and increasing the number of refinement iterations.

3.5. *Scalability of the algorithm*

The adoption of 1-norm regularization for the formulation of the learning problem requires that the input catalog features are explicitly projected in the space of all possible Boolean terms which can be generated by their combination. Dealing with the explicit projection Φ in Eq. (2) is tractable only for a rather limited number of catalog features and size of conjunctions d. However, this will typically be the case when interacting with a human DM. The bounded rationality of humans indeed allows them to handle non-linear interactions just among a small number of features.

A possible alternative formulation consists of directly learning a non-linear function of the features, without explicitly projecting them to the resulting higher dimensional space. This can be achieved by replacing 1-norm regularization with 2-norm in Eq. (2), thus recovering the original support vector ranking formulation [16, 17], and considering the kernelized version of the resulting dual problem. A similar solution for the regression (rather than ranking) formulation and plain MAX-SAT problems (rather than MAX-SMT, see later) was considered in our previous work [11]. As expected, the experimental results showed the superior performance of 1-norm over 2-norm regularization in a setting with many irrelevant noisy features, due to its sparsity-inducing property. See Section 7 for a discussion on how to adapt the full MAX-SMT preference learning formulation to deal with implicit feature spaces.

4. Satisfiability Modulo Theory

In the previous section, we assumed our optimization task could be cast into a *propositional* Satisfiability problem. However, the formalism of plain propositional logic is not suitable or expressive enough to represent many real-world applications, arising, for example, in the fields of real-time control system design and formal software verification. For example, software verification applications need reasoning about equalities, arithmetic operations and data structures. These problems require or are more naturally described in more expressive theories as first-order logic (FOL), involving quantifiers, functions and predicates. Consider the toy example represented by the following formula from the theory of arithmetic over integers:

$$x + y + z \leq 4, \; x, y, z \in \{1, 2, 3\}$$

We are interested in deciding whether there is an assignment of integer values to the variables x, y and z satisfying the formula. A possible encoding into an equisatisfiable SAT proposition is given by:

$$(x_1 \wedge y_1 \wedge z_1) \vee (x_1 \wedge y_1 \wedge z_2) \vee (x_1 \wedge y_2 \wedge z_1) \vee \ldots$$

where x_i, y_i, z_i, with $i = 1, 2, 3$, is a Boolean variable which is true when the integer-valued variable x, y, z is assigned the value i, respectively. Let us note the blow-up in the translation affecting the SAT instance size. In general, propositional logic is not expressive enough for the efficient encoding of many real world tasks: important structural information may be lost or exponential blow-up in the formula size may be caused (e.g., up to a 2^{32} factor to represent all the candidate values that a 32-bit integer variable may assume).

Satisfiability Modulo Theory (SMT) [10, 18] problems generalize SAT problems by considering the satisfiability of a FOL formula with respect to a certain *background theory T* fixing the interpretation of (some of the) predicate and function symbols. Any procedure designed to solve a SMT problem is called SMT solver. Popular examples of useful theories include various theories of arithmetic over reals or integers such as linear or difference ones. Linear arithmetic considers $+$ and $-$ functions alone, applied to either numerical constants or variables, plus multiplication by a numerical constant. Difference arithmetic is a fragment of linear arithmetic limiting legal predicates to the form $x - y \leq c$, where x, y are variables and c is a numerical constant. A number of theories have been studied apart from standard arithmetic ones. Machine arithmetic, for instance, is more naturally modeled by the theory of bit-vector arithmetic, which includes bit-wise operations. Other theories exist for data structures such as lists and strings [10].

Different approaches have been developed to solve SMT problems. When deciding the satisfiability of a first-order formula φ in a given theory T, a general purpose FOL reasoning system such as Prolog, based on the resolution calculus, needs to add to the formula a conjunction of all the axioms in T. This is, e.g., the standard setting in inductive logic programming when verifying whether a certain hypothesis covers an example given the available background knowledge. Let us note that, whenever the cost of including such additional background theory is affordable, our optimization algorithm can be applied rather straightforwardly. Unfortunately, adding all axioms of T is not viable for many theories of interest: consider for instance the theory of *arithmetic*, which defines the interpretation of symbols such as $+, \geq, 0, 5$.

An alternative approach is represented by *eager* SMT solvers. They translate the original formula φ taken from the input theory T into an *equisatisfiable* propositional formula *in one single step*. In this way, any off-the-shelf SAT solver can be used to check the satisfiability of the generated propositional formula. The SAT solver is called once. However, a specific translator has to be developed for each theory of interest. Furthermore, the translation of *all* the theory-specific information is required at the *beginning* of the search process (hence the name “eager”), likely resulting in large SAT formulas. Although optimizations in the translation are possible, there is a trade-off between the degree of optimization and the time required by the SAT encoding. Let us note the analogy with compilers, optimizing the low-level object code (the SAT formula in our context) generated from a high-level program (the SMT problem formulation) [10].

A more efficient approach is based on incremental translations and calls to the SAT solver. This is the case of *lazy* SMT solvers, where the theory-specific information is incrementally encoded in the SAT formulation of the problem. In particular, the lazy approach exploits *specialized* reasoning methods for the background theory of interest. When integrated as submodules in a SMT solver, these theory-specific reasoning methods are often referred as *T-solvers*. T-solvers are efficient decision procedures typically developed to check the satisfiability of *conjunctions* of literals (i.e., atomic formulas and their negations) over the given theory T. The generalization to arbitrary propositional structures is handled in conjunction with the SAT solver integrated in the SMT solver.

In this work we focus on lazy SMT solvers and their optimization variant, which we integrated in our optimization algorithm. The rest of this section provides details about lazy SMT solvers search process and introduces Optimization Modulo Theory (OMT) solvers handling weighted Maximum Satisfiability Modulo Theory (MAX-SMT) problems. Finally, the integration of MAX-SMT solvers in our optimization approach is discussed.

4.1. Lazy Satisfiability Modulo Theory solvers

The search process of a lazy SMT solver alternates calls to the Satisfiability and the theory solver respectively, until a solution satisfying both solvers is retrieved or the problem is found to be unsatisfiable. Let φ be a formula in a certain theory T, made of a set of n predicates $A = \{a_1, \dots, a_n\}$. A mapping α maps φ into a propositional formula $\alpha(\varphi)$ by replacing its predicates with propositional variables $p_i = \alpha(a_i)$. The inverse mapping β replaces propositional variables with their corresponding predicates, i.e., $\beta(p_i) = a_i$. For example, consider the following formula in the arithmetic theory over integers:

$$x + y + z \leq 3 \wedge (x \leq y \vee z = 2) \wedge (x \geq 2 \vee x \neq z) \tag{4}$$

where x, y, z are integer-valued variables. Then, $p_1 = \alpha(x + y + z \leq 3)$, $p_2 = \alpha(x \leq y)$, $p_3 = \alpha(z = 2)$, $p_4 = \alpha(x \geq 2)$ and $p_5 = \alpha(x \neq z)$. The resulting propositional formula $\alpha(\varphi)$ is:

$$p_1 \wedge (p_2 \vee p_3) \wedge (p_4 \vee p_5)$$

Let us note that the truth assignment

$$p_1 = \top, p_2 = \bot, p_3 = \top, p_4 = \top, p_5 = \bot$$

where $\top, \bot$ symbols encode true and false truth values, respectively, is equivalent to the statement

$$x + y + z \leq 3 \wedge x > y \wedge z = 2 \wedge x \geq 2 \wedge x = z$$

in the theory T.

The SAT solver integrated in the SMT solver searches a solution of the propositional formula $\alpha(\varphi)$. If the propositional formula in unsatisfiable, the original formula φ is also unsatisfiable and the whole SMT solver stops. Otherwise, the SAT solver provides a truth assignment satisfying $\alpha(\varphi)$. Considering the above example, it may be:

$$p_1 = \top, p_2 = \top, p_3 = \top, p_4 = \top, p_5 = \top \tag{5}$$

The T-sover is used to validate the assignment (i.e., the conjunction of truth values) produced by the SAT solver. The predicates are evaluated using the rules of the theory T. If the validation is successful, the SMT solver stops returning the assignment of values to the variables in T satisfying φ. Otherwise, when the T-solver detects unsatisfiability, an

```
1. procedure SMT solver($\varphi$)
2.     $\psi = \alpha(\varphi)$
3.     while (true)
4.         (r,M) $\leftarrow$ SAT($\psi$)
5.         if r = unsat then return unsat
6.         (r,J) $\leftarrow$ T-Solver($\beta(M)$)
7.         if r = sat then return sat
8.         C $\leftarrow \bigvee_{l \in J} \neg\alpha(l)$
9.         $\psi \leftarrow \psi \wedge C$
```

Figure 2: Pseudocode for a basic lazy SMT solver.

additional constraint explaining (i.e, justifying) the unsatisfiability is included in $\alpha(\varphi)$ and the SAT solver is asked for a new assignment. For example, the assignment in Eq. 5 is not valid in the arithmetic theory over integers. Applying the inverse mapping β, we obtain:

$$x + y + z \leq 3 \wedge x \leq y \wedge z = 2 \wedge x \geq 2 \wedge x \neq z$$

which is unsatisfiable as z is set to the value two while x and y must be larger than value two. A possible justification explaining the unsatisfiability is given by the following constraint:

$$\neg(p_1 \wedge p_2 \wedge p_3 \wedge p_4)$$

which will be included in the propositional formula $\alpha(\varphi)$ for the following calls to the SAT solver. Continuing with the toy example, assume that the second call to the SAT solver returns the following truth assignment:

$$p_1 = \top, p_2 = \bot, p_3 = \top, p_4 = \bot, p_5 = \top \quad (6)$$

satisfying the above justification. Restoring the interpretation of the propositional variables p_i, we obtain:

$$x + y + z \leq 3 \wedge x > y \wedge z = 2 \wedge x < 2 \wedge x \neq z$$

which is satisfied by posing, e.g., $x = 1, y = 0, z = 2$.

The term *lazy* denoting this approach is due to the incremental strategy generating constraints on demand. On the contrary, eager methods produce all the constraints in one single step before the execution of the SAT solver.

Figure 2 reports the basic form [18] of a SMT algorithm. `SAT`(ψ) calls the SAT solver on the ψ instance, returning a pair (r, M), where r is `sat` if the instance is satisfiable, `unsat` otherwise. In the former case, M is a truth assignment satisfying ψ. `T-Solver`(S) calls the theory solver on the formula S and returns a pair (r, J), where r indicates if the formula is satisfiable. If $r =$ `unsat`, J is a *justification* for S, i.e., any unsatisfiable subset $J \subset S$. The next iteration calls the SAT solver on an extended instance accounting for this justification.

Off-the-shelf solvers introduce a number of refinements to this basic strategy, by pursuing a tighter integration between the SAT and the theory solvers. A common approach consists of pruning the search space for the SAT solver by calling the theory solver on partial assignments and propagating its results. Finally, combination methods exist to jointly employ different theories, see [19] for a basic procedure.

4.2. Weighted MAX-SMT and Optimization Modulo Theory solvers

Weighted MAX-SMT generalizes SMT problem much like weighted MAX-SAT does with SAT ones. Given a cost function c, an assignment $\mathbf{s}$ in the input theory T is sought with minimum $c(\mathbf{s})$. The simplest formulation for the MAX-SMT problem consists of assigning a weight to each part (i.e., constraint) of the Boolean formula to be jointly satisfied. Weights represent penalties or costs for violating the constraints and are expressed by positive natural or real numbers. The cost function $c(\mathbf{s})$ is defined by the sum of the weights of the constraints unsatisfied under the assignment $\mathbf{s}$.

While a body of works exist addressing weighted MAX-SAT problems, MAX-SMT task has been tackled only recently [8, 9, 20] and very few solvers have been developed. In particular, optimization in SMT was first introduced by the work in [8]. Let $\{(C_1, w_1), \ldots, (C_m, w_m)\}$ the set of the weighted constraints, where C_i and w_i identify the i-*th* constraint and its associated weight, respectively. Clearly, the cost function c takes values from the interval $[0, W]$, with $W = w_1 + \ldots + w_m$. A simple effective method [8] to obtain solutions with cost $c(\mathbf{s})$ at most $\overline{W}$, with $\overline{W} < W$, consists of generating m additional constraints $(C_i \vee p_i)$, $i = 1 \ldots m$, with p_i a fresh propositional variable. The initial background theory T is augmented with the integers and with the following assertions:

$$\begin{aligned} p_i \rightarrow (k_i = w_i),\ i = 1 \ldots m \\ \neg p_i \rightarrow (k_i = 0),\ i = 1 \ldots m \\ k_1 + \ldots + k_m \leq \overline{W} \end{aligned}$$

In this way, any assignment $\mathbf{s}$ with cost larger than the upper bound $\overline{W}$ on the cost function is inconsistent with the augmented input theory. The search for the optimal assignment $\mathbf{s}^*$ is based on a branch and bound technique. Each time a lower upper bound $\overline{W}' < \overline{W}$ on the cost function is computed, the theory is updated by replacing $\overline{W}$ with $\overline{W}'$ (this operation is called "theory strengthening"). A sequence of theories with an increasingly tight upper bound on the cost function is therefore generated. This strategy is referred to as "SMT with progressively stronger theories". A different approach to handle MAX-SMT is used in the paper [9], where a theory of costs C is introduced to handle cost functions in SMT and a specialized theory-solver for C is developed within the standard lazy SMT schema.

4.3. Preference elicitation with the MAX-SMT formalism

The adoption of the MAX-SMT formalism in our approach enables an efficient representation of non-Boolean decisional features. The framework introduced in Sec. 2 does not need to be changed in the case of non-Boolean decisional features, because the effort required to handle non-Boolean encodings is completely performed by the MAX-SMT solver. As a matter of fact, when representing user preferences in the SMT setting, the DM utility function f is expressed as a weighted sum of terms, where a term is the conjunction of up to *d predicates* defined over the variables in the theory T. The set of *all* possible predicates represents the search space S of the MAX-SAT solver integrated in the MAX-SMT solver. Our approach learns an approximation $\hat{f}$ of f and gets one of its optimizers from the MAX-SMT solver, as an assignment to the decision variables in the input theory T involved in the predicates of $\hat{f}$. The decision variables in T that are not used in the current formulation $\hat{f}$ are assigned values selected independently and uniformly at random from the their domains. In this way, a complete assignment to the variables in the input theory (i.e., a candidate solution in T) is obtained. The candidate solution in turn determines an assignment $\mathbf{p}^* = (p_1^*, \ldots, p_n^*)$ of Boolean values ($p_i^* = \{\mathbf{true}, \mathbf{false}\}$) to the predicates in S.

The diversification strategy to obtain multiple candidates solutions is the same as described in Sec. 2.3. The sequential optimization of $\hat{f}$ is performed, with the additional *hard* constraint generated by the disjunction of all the terms of $\hat{f}$ unsatisfied by $\mathbf{p}^*$. If $\mathbf{p}^*$ satisfies all the terms of $\hat{f}$, the additional *hard* constraint consists of:

$$(\neg p_1^* \vee \neg p_2^* \ldots \vee \neg p_n^*)$$

which excludes $\mathbf{p}^*$ from the feasible solutions set of $\hat{f}$.

5. Related work

The problem of automatically learning utility functions and eliciting preferences from users is widely studied within the Artificial Intelligence community [21]. A popular approach to model the uncertain knowledge about the DM preferences consists of assuming a set of hypotheses, with no belief on the strength of the hypotheses. The set of hypotheses contains the feasible utility functions and reflects the partial knowledge about the DM preferences. The uncertainty about the DM preferences is decreased by restricting the feasible hypothesis set, when relevant preference information is received during the elicitation process. This approach is often referred to as reasoning under strict uncertainty [22]. An alternative uncertainty model consists of defining a probability distribution over the hypotheses. This is the case of Bayesian approaches to preference elicitation [23, 15, 24, 25].

Our work casts the preference elicitation task into the problem of learning combinatorial optimization instances to be optimized for the generation of additional candidate solutions. The learning phase is accomplished by applying techniques from Machine Learning, while the optimization (or search) stage is based on Combinatorial Optimization methods. Our approach satisfies the main principles [15] needed for practical applicability of preference elicitation techniques: the need for multi-attribute models, i.e., candidate configurations represented by means of multiple decisional features or attributes; the quest for real-time interaction with the DM, where both the query generation and the solutions recommendation must be accomplished in no more than few seconds; the robustness to inconsistent and contradictory feedback from the DM; the need for cognitively affordable queries to the user, i.e., comparison queries; the demand for scalable methods, that evaluate at each preference elicitation stage a number of candidate queries that grows linearly in the cardinality of the solutions space.

From the perspective of Machine Learning/Artificial Intelligence, the closest approaches to ours are the methods based on the minimax regret criterion [4, 5, 6], while a recent technique [7] developed within the Constraint Programming community shares with our algorithm the combinatorial formulation of the DM preferences. In the following, we review these alternative approaches and compare them with our technique.

5.1. Minimax regret-based approaches

The methods developed in the papers [4, 5, 6] perform preference elicitation under strict uncertainty. They assume a parametric formulation of the candidate utility function (hypothesis) in the feasible utility set U. The parametrization, based on the generalized additive independence models discussed in Sec. 1, enables a compact way to specify the feasible set, which is represented by bounds and constrains on the parameters. Uncertainty is thus reduced by tightening the constraints or increasing (decreasing) the lower (upper) bounds.

To make decisions with the partial utility information under strict uncertainty and, in particular, to select the final configuration to be returned to the DM, the *minimax regret* decision criterion is used. It prescribes the configuration that minimizes the maximum regret with respect to all the possible realizations of the DM utility function in the set U. Thus, the minimax regret criterion minimizes the *worst-case loss* with respect to the possible realizations of the DM utility function. In detail, the minimax regret criterion is defined in two stages, building on the maximum pairwise regret and the maximum regret. The maximum pairwise regret of configuration $\mathbf{x}$ with respect to configuration $\mathbf{x}'$ over the feasible utility set U is defined as:

$$\mathrm{R}(\mathbf{x}, \mathbf{x}', \mathrm{U}) = \max_{u \in \mathrm{U}} \; u(\mathbf{x}') - u(\mathbf{x}) \tag{7}$$

This formulation can be interpreted by assuming an adversary that can impose any DM utility function u in U and chooses the one that maximizes the regret of selecting solution $\mathbf{x}$. The function $u^w = \operatorname{argmax} \mathrm{R}(\mathbf{x}, \mathbf{x}', \mathrm{U})$ is thus termed the "adversary's utility" or "witness utility". The maximum regret of choosing solution $\mathbf{x}$ with respect to the feasible utility set U is defined as:

$$\mathrm{MR}(\mathbf{x}, \mathrm{U}) = \max_{\mathbf{x}'} \; \mathrm{R}(\mathbf{x}, \mathbf{x}', \mathrm{U}) \tag{8}$$

Within the "adversary metaphor", let us note that the $\mathbf{x}'$ chosen by the adversary for the specific u^w is the optimal decision under u^w (i.e., $\mathbf{x}'$ maximizes u^w) and any alternative choice would give the adversary less utility and thus reduce the user regret. Finally, the minimax regret of the feasible utility set U is as follows:

$$\mathrm{MMR}(\mathrm{U}) = \min_{\mathbf{x}} \; \mathrm{MR}(\mathbf{x}, \mathrm{U}) \tag{9}$$

and the solution $\mathbf{x}^r = \operatorname{argmin} \mathrm{MR}(\mathbf{x}, \mathrm{U})$ minimizing the maximum regret is the configuration recommended to the DM by the minimax regret decision criterion. The quality of the solution $\mathbf{x}^r$ is guaranteed to be no more than MMR(U) away from the quality of the DM favourite solution, and no alternative solution has a better guarantee, i.e., for all $\mathbf{x} \neq \mathbf{x}^r$, $\mathrm{MR}(\mathbf{x}, \mathrm{U}) \geq \mathrm{MMR}(\mathrm{U})$.

The initial bounds about the utility parameters defined by the DM are not usually tight enough to identify configurations with provably low regret, and a configuration satisficing the DM cannot be recommended without eliciting additional preference information. This is achieved through an interactive elicitation algorithm that asks queries to the DM and, based on the information elicited, refines the bounds and the constraints on the utility parameters. The generic framework of the approach is as follows:

```
input: initial constraints (e.g., bounds) on the utility parameters defining
           the initial feasible set U

compute minimax regret MMR(U);
```

```
repeat until termination criterion
    ask query $q$;
    refine U by updating the constraints over utility parameters to reflect the
    response to $q$;
    recompute MMR(U) with respect to the refined set U;
    return to the DM the solution $\mathbf{x}^r$ minimizing MR($\mathbf{x}$, U)
```

Computationally tractable techniques have been proposed [4, 5, 6] to compute the minimax regret MMR (U). The iterative algorithm may be stopped by the DM when she is satisfied by the returned solution $\mathbf{x}^r$ or when the minimax regret MMR(U) reaches a certain level τ. When the minimax regret is reduced to the value zero, the solution $\mathbf{x}^r$ returned by the algorithm is guaranteed to be the DM favourite solution. The minimax regret-based approach also enables a principled method to define informative queries that will be asked the DM (query optimization), and different query strategies have been proposed [4, 5, 6].

5.1.1. Comparison with our method

While the target of our work is a preference elicitation method approximately correct with high probability, the minimax regret-based approaches assume an adversarial entity that acts to maximize the DM regret and they aim at beating the adversary by recommending the best solution with respect to the worst case loss. However, this adversarial model is not always strongly motivated by real-world applications, where users are typically interested in the actual obtained results rather than in regret. The main advantage of the regret-based approaches with respect to our algorithm is the ability to provide a lower bound about the quality of the recommended solution and guarantee the convergence to provably-optimal results. However, these theoretical guarantees are valid under the assumption that the feasible set U contains the true DM utility function at *any* iteration of the elicitation process. That is, the regret-based methods do not consider the uncertain and inconsistent preference information characterizing the typical human decision processes. As a matter of fact, uncertain feedback from the DM translates into constraints on the utility parameters that can potentially rule out the true utility from the feasible set U. Furthermore, the best performance observed in the experiments presented in the paper [4] is achieved by query strategies that include standard gamble queries, which require the users to state their preference over a probability distribution of configurations. These queries demand a higher DM cognitive load than the comparison queries adopted in our work, and thus in real-world applications they are more prone to errors and inconsistent answers from the users. Without suitable modifications (e.g., constraints relaxation) to recover from the inevitable uncertain and inconsistent preference information elicited from the DM, regret-based approaches cannot be applied in the realistic problem settings and the noisy test cases that we consider in this work.

5.2. Preference elicitation methods based on constraint satisfaction

Recent work in the field of constraint programming [7] shares with our technique the combinatorial approach to model user preferences. It defines the user preferences in terms of *soft* constraints and introduces constraint optimization problems where the DM preferences are not completely known before the solving process starts. Let us first briefly describe the c-semiring formalism [26] adopted in paper [7] to model soft constraints.

In soft constraints, a generalization of hard constraints, each assignment to the variables of one constraint is associated with a preference value taken from a preference set. The preference value represents the level of desirability of the assignment to the variables of the constraint. As the preference score is associated to a partial assignment to the problem variables, it represents a *local* preference value. The desirability of a complete assignment is defined by a *global* preference score, computed by applying a combination operator to the local preference values. A set of soft constraints generates an order (partial or total) over the complete assignments of the variables of the problem. Given two solutions of the problem, the preferred one is selected by computing their global preference levels. Soft constraints are represented by an algebraic structure, called *c-semiring* (where letter "c" stays for "constraint"), providing two operations for combining ($\times$) and comparing ($+$) preference values. In detail, the c-semiring is a tuple $(A, +, \times, \mathbf{0}, \mathbf{1})$ where:

- A is a set and $\mathbf{0}, \mathbf{1} \in A$;
- $+$ is commutative, associative and idempotent; $\mathbf{0}$ is its unit element and $\mathbf{1}$ is its absorbing element;

- $\times$ is commutative, associative, distributes over $+$; **1** is its unit element and **0** is its absorbing element.

Let us note that a c-semiring is a semiring with additional properties for the two operations: the operation $+$ must be idempotent and with **1** as absorbing element, the operation $\times$ must be commutative. The relation $\leq_A$ over A, $a_2 \leq_A a_1$ iff $a_2 + a_1 = a_1$, is a partial order, with **0** and **1** its minimum and maximum elements, respectively. The relation $\leq_A$ allows to compare (some of) the desirability levels, with $a_2 \leq_A a_1$ meaning that a_1 is "better" than a_2; **0** and **1** represent the worst and the best preference levels, respectively, and the operations $+$ and $\times$ are monotone on $\leq_A$. Consider, e.g., the following instance of c-semiring:

$$(\{5, 10, 15, \ldots, 50\}, \max, \min, 5, 50)$$

with preference values from the set $\{5, 10, 15, \ldots, 50\}$ and elements **0** and **1** represented by the values 5 and 50, respectively. The desirability of a complete assignment is obtained by taking its minimum local preference value. A complete assignment c_1 with preference score a_1 is preferred to a complete assignment c_2 with lower preference score a_2. That is, $a_2 \leq_A a_1$ iff $\max(a_2, a_1) = a_1$.

The generality of the semiring-based soft constraint formalism permits to express several kinds of preferences, including partially ordered ones. For example, different instances of c-semirings encode weighted or probabilistic soft constraint satisfaction problems [27]. However, the c-semiring formalism can model just *negative* preferences. First, the best element in the ordering induced by $\leq_A$, denoted by **1**, behaves as indifference, since $\forall a \in A, 1 \times a = a$. This result is consistent with intuition: when using only negative preferences, indifference is the best level of desirability that can be expressed. Furthermore, the combination of desirability levels returns a lower overall preference, since $a \times b \leq_A a, b$, again consistently with the fact of dealing with negative preferences.

Preference elicitation strategies have been introduced [7] within this formalism in order to deal with scenarios where preference value information is partially unknown. Some of the local preference values attached to soft constraints are assumed to be missing, and the DM is asked for an explicit feedback on specific assignments for these constraints, in terms of score values quantifying her preference for a certain assignment. The elicitation strategy is aimed at minimizing the number of queries to the DM.

5.2.1. Comparison with our method

Concerning expressivity of the representation formalisms, the work in [28] shows how to encode semiring-based soft constraint satisfaction problem (SCSP) instances into equivalent weighted MAX-SAT formulations. Each solution of the latter instance corresponds to a solution of the former one. Details on the encoding algorithm can be found in the Appendix. The rationale for the MAX-SAT encoding is the exploitation of the efficient and widely studied techniques implemented in modern SAT solvers, which can efficiently handle large size structured problems [29]. The encoding can in principle be applied also to SCSPs with continuous decision variables or discrete variables defined over large size finite domains, possibly however at the cost of a significant blow-up in the translation. In this case, one may cast the SCSP instance into a weighted MAX-SMT rather than a weighted MAX-SAT formulation.

Concerning the preference elicitation setting, our formulation assumes a much more limited amount of initial knowledge about the problem to be optimized. In the work on preference elicitation for SCSPs [7], decision variables, soft constraint topology and structure are assumed to be known in advance and the incomplete initial information consists only of missing local preference values. We assume complete ignorance about the structure of the constraints over the decisional variables of the user. The initial problem knowledge is limited to a set of catalog features. Our algorithm extracts the decisional items of the DM from the set of catalog features and learns the weighted terms constructed from them modeling the DM preferences. If the MAX-SAT encoding is applied to the SCSP with missing preferences, it produces a Boolean formula where some of the weights of the terms are not known. On the other hand, our preference elicitation algorithm handles MAX-SAT instances where both the terms and their associated weights are initially unknown and are learnt by interacting with the DM.

Furthermore, the technique in [7] is based on *local* elicitation queries, with the final user asked to reveal her preferences about assignments for specific soft constraints. *Global* preferences or bounds for global preferences associated to complete solutions of the problem are derived from the local preference information. Our technique goes in the opposite direction: it asks the user to compare complete solutions and learns local utilities (i.e., the weights of the terms of the logic formula) from global preference values. In many cases, recognizing appealing or unsatisfactory global solutions may be much easier than defining local utility functions, associated to partial solutions. For example, while scheduling a set of activities, the evaluation of complete schedules may be more affordable than assessing how specific ordering choices between couples of activities contribute to the global preference value. Furthermore the preference elicitation technique in [7] asks the DM for quantitative evaluations of partial solutions: she does not just rank couples of activities, she provides score values quantifying her preference for the partial activity rankings, a much more demanding task.

In order to reduce the embarrassment of the decision maker when specifying precise preference scores, interval-valued constraints [30] allow users to state an interval of utility values for each instantiation of the variables of a constraint. As a matter of fact, the informal definitions of degrees of preference such as “quite high”, “more or less”, “low” or “undesirable” cannot be naturally mapped to precise preference scores. However, the technique described in [30] requires the user to provide all the information she has about the problem (in terms of preference intervals) *before* the solving phase, without seeing any optimization result.

Even if interval-valued constraints [30] have been introduced to handle uncertainty in the evaluations of the DM, inconsistent preference information is not addressed [7]. This is a requirement to retain the optimality guarantees provided by the preference elicitation strategy. Conversely, our algorithm trades optimality for robustness and can effectively deal with imprecise information from the DM, modeled in terms of inaccurate ranking of the candidate solutions.

Finally, while the work in [7] considers *unipolar* preference problems, modeling just negative preferences, our approach naturally accounts for bipolar preference problems, with the final user specifying what she likes and what she dislikes. Bipolar preference problems provide a better representation of the typical human decision process, where the degree of preference for a solution reflects the compensation value obtained by comparing its advantages with the disadvantages. Let us note that the work in [27] extends the soft constraint formalism to account for bipolar preference problems.

6. Experimental results

The following empirical evaluation demonstrates the versatility and the efficiency of our preference elicitation approach. First, the simplest formulation of our method, which considers Boolean decisional features only, is evaluated over a set of synthetic MAX-SAT benchmarks. Then, our approach is tested over a benchmark of MAX-SMT problems, formulating realistic preference elicitation tasks. Because the Satisfiability Modulo Theory formalism encompasses the propositional logic, our algorithm simply handles MAX-SAT preference models as instantiations of the MAX-SMT task without any auxiliary information. The MAX-SMT tool used for the experiments is the “Yices” solver [20], which is publicly available at `http://yices.csl.sri.com/` (as of September 2012). Each point of the curves depicting our results is the median value over 400 runs with different random seeds, unless otherwise stated.

6.1. Weighted MAX-SAT

Our algorithm was tested over a benchmark of randomly generated utility functions according to the triplet (*number of features, number of terms, max term size*), where *max term size* is the maximum allowed number of Boolean variables per term. We generate functions for the following values: $\{(5,3,3),(6,4,3),(7,6,3),(8,7,3),(9,8,3),(10,9,3)\}$. Each utility function has two terms with maximum size. Terms weights are integers selected uniformly at random in the interval $[-100,0)\cup(0,100]$. We consider as *gold* standard solution the configuration obtained by optimizing the DM unknown utility function (henceforth the *target* utility function).

The number of catalog features is 40. The maximum size of terms is assumed to be known. Furthermore, the probability of inaccurate feedback swapping a pairwise preference is fixed to the value 0.1.

We run a set of experiments where the training set is initialized by the ranking of $10,20,\ldots 100$ configurations generated by sampling uniformly at random the Boolean feature values. Fig. 3 reports the utility loss (or regret) of the *best* configuration, i.e., the solution optimizing the current approximation of the target utility function, at the different iterations for an increasing number s of initial configurations to be ranked. At the following iterations, $s/2$ configurations are generated and their ranking is added to the training set (see Sec. 2). The utility loss is the amount of utility lost by recommending the best rather than the gold configuration and it is measured as the difference between their respective costs (approximation error). Considering the simplest problems with three and four terms, our algorithm can identify the solution preferred by the DM at the first iteration, provided that at least 90 initial

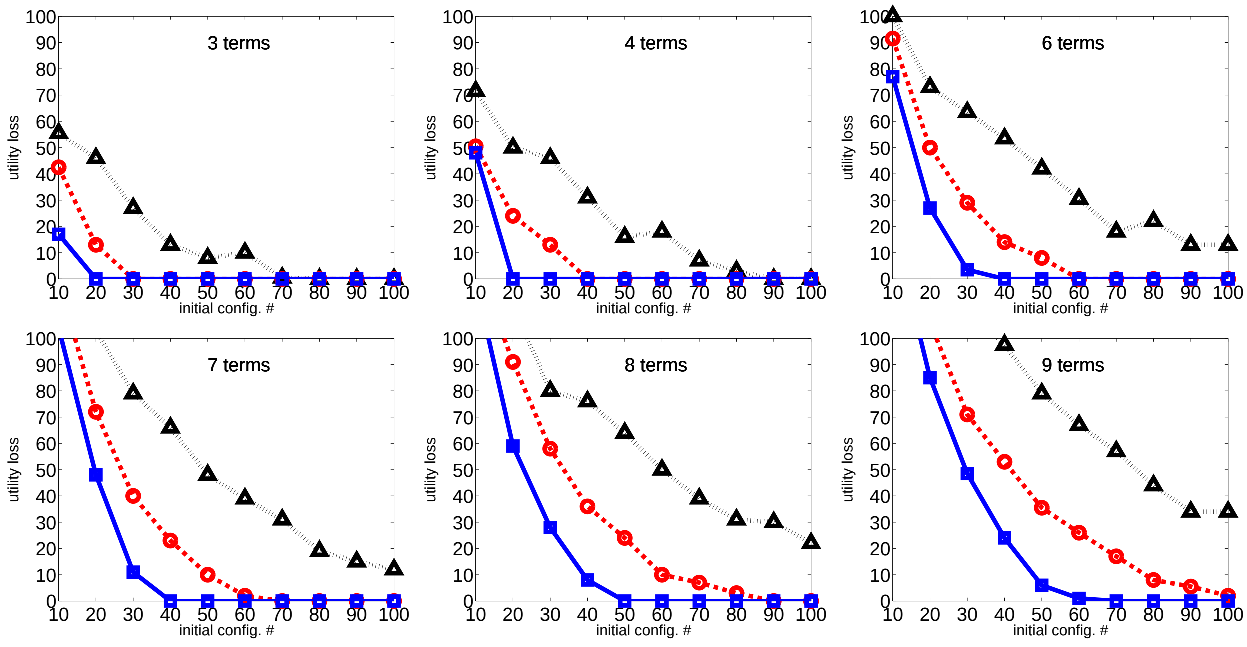

Figure 3: Learning curves for different iterations of our algorithm observed when increasing the number of initial configurations to be ranked. The *y*-axis reports the utility loss, while the *x*-axis contains the number of initial configurations. The dotted line with triangular markers, the dashed line with circular markers and the solid line with square markers show the performance of the algorithm at the first, the second and the third iteration, respectively. The number of configurations for each iteration following the first one is half the number of initial configurations. See text for details.

configurations are available. With three iterations, the number of configurations needed to recover the optimal solution for both problems is reduced to 20+10+10=40 configurations divided in three batches. Utility functions including more than five terms are quite unrealistic in the case of a human decision maker, however they are reported here to show the scalability of our approach to more complex elicitation tasks. With more than four terms in the target utility function, the optimal solution cannot be recovered at the first iteration. However, our algorithm succeeds in exploiting its active learning strategy and converges to the optimal solution when enough iterations are provided. For instance, in the case of eight terms in the target utility function, the optimal solution is discovered at the second and third iterations with 90 and 50 and initial configurations, respectively.

Fig. 4 and Fig. 5 show the learning curves for our approach at the first and third iterations in the case of target utility functions with three and nine terms, respectively. Error bars indicate the range between the 25th and 75th percentiles of the underlying data distributions. As expected, in both cases the sample percentiles demonstrate a more stable behavior of our technique at the third iteration. In particular, at the third iteration the stability of our algorithm increases with additional training examples, while the variability of performance observed at the first iteration does not decrease to the same extent. Considering the more challenging case represented by target utility functions with nine terms, at the third iteration our algorithm consistently finds the gold solution with at least 70 initial configurations (the interquartile range value is within the 20 units). An unstable behavior is still observed at the first iteration even in the case of 100 configurations.

6.2. *Weighted MAX-SMT*

MAX-SMT is a recent research area. Even if existing results [8] indicate that MAX-SMT solvers can efficiently address real-world problems, to the best of our knowledge no well-established publicly available MAX-SMT benchmarks exist and preference elicitation tasks have not been encoded into MAX-SMT instances yet.

In this work, we modeled a *scheduling* problem as a MAX-SMT instance, where the DM expresses her preferences about the candidate schedules of a set of jobs. In order to mimick a realistic recommendation task, we also designed a *housing* problem aimed at selecting a location for building a house. The formulation consists of both unknown soft constraints representing the user preferences and known hard constraints defining the feasible search space.

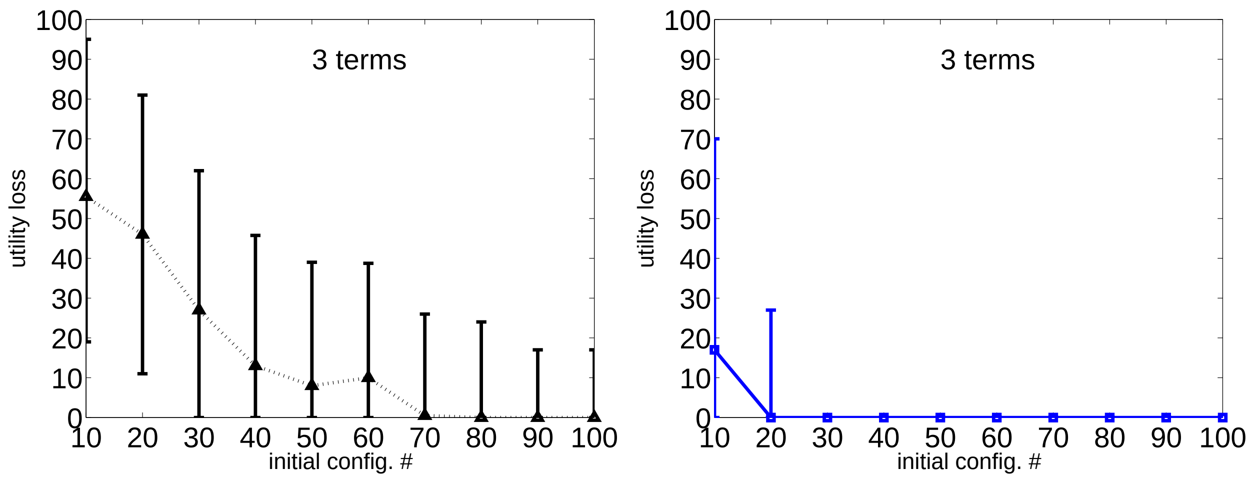

Figure 4: Learning curves at the first (left figure) and the third (right figure) iterations in the case of target utility functions with three terms. Error bars denote the range among the 25th and the 75th percentiles of the measurements.

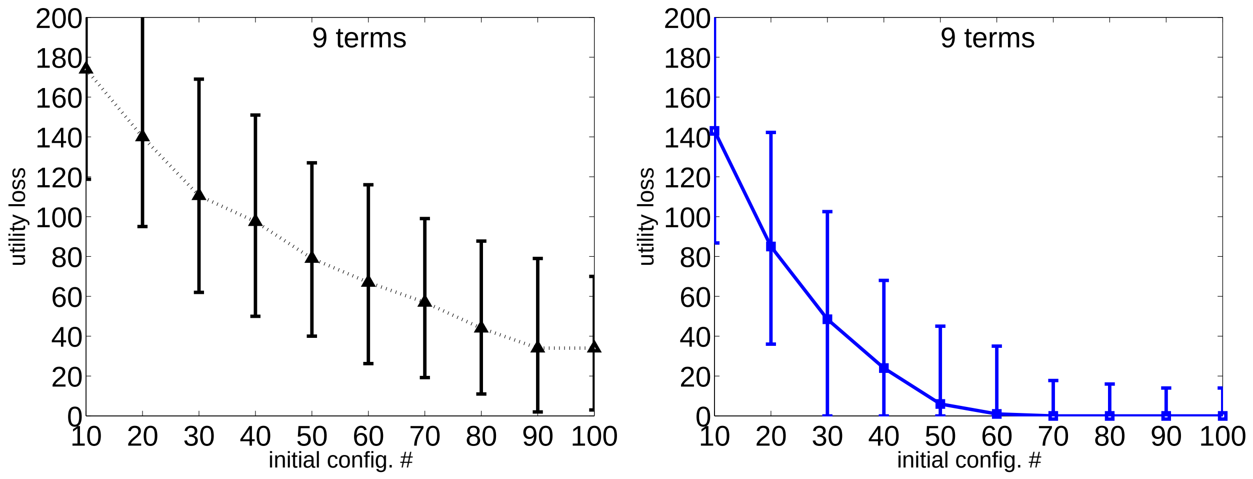

Figure 5: Learning curves at the first (left figure) and the third (right figure) iterations in the case of target utility functions with nine terms. Error bars denote the range among the 25th and the 75th percentiles of the measurements.

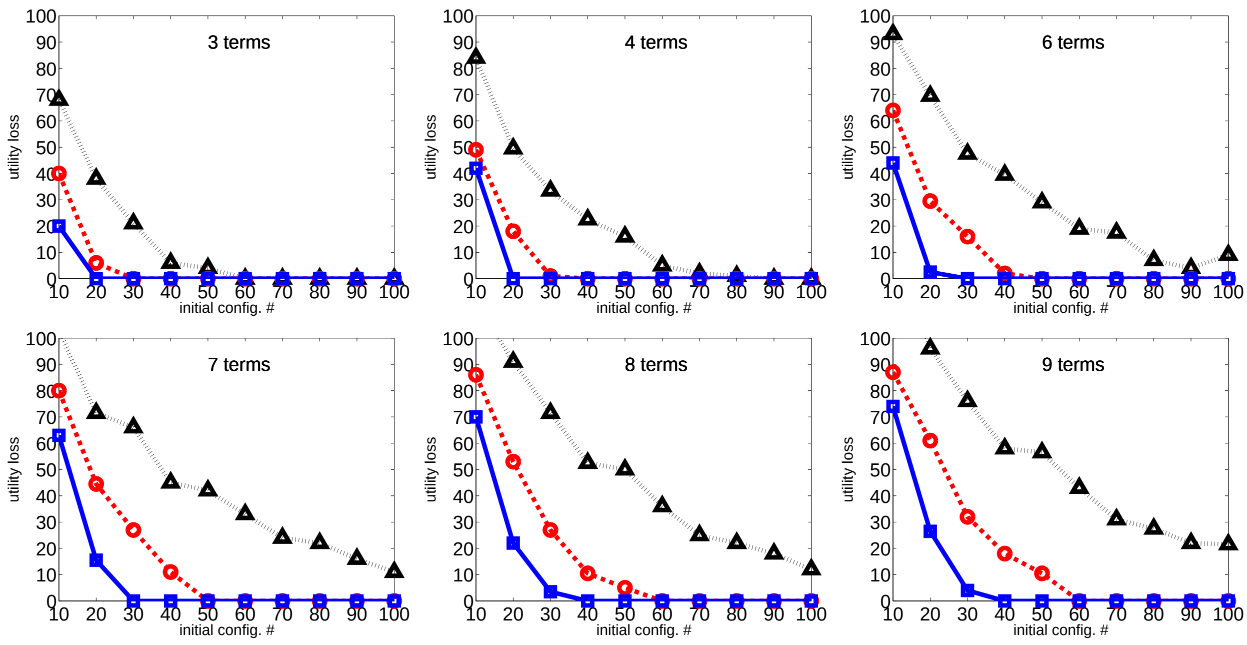

Figure 6: Learning curves observed at different iterations of our algorithm while solving the scheduling problem. The y-axis reports the utility loss, while the x-axis contains the number of initial configurations. The dotted line with triangular markers, the dashed line with circular markers and the solid line with square markers show the performance of the algorithm at the first, the second and the third iteration, respectively. The number of configurations for each iteration following the first one is half the number of initial configurations. See text for details.

6.2.1. Scheduling problem

A set of five jobs must be scheduled over a given period of time. Each job has a fixed known duration, the constraints define the overlap of two jobs or their non-concurrent execution. The target utility function is generated by selecting uniformly at random weighted terms over the constraints. The solution of the problem is a schedule assigning a starting date to each job and minimizing the cost, where the cost of the schedule is the sum of the weights of the violated terms of the target utility function. The temporal constraints are expressed by using the difference arithmetic theory. In detail, let s_i and d_i, with $i = 1 \ldots 5$, be the starting date and the duration of the i-*th* job, respectively. If s_i is scheduled before s_j, the constraint expressing the overlap of the two jobs is $s_j - s_i < d_i$, while their non-concurrent execution is encoded by $s_j - s_i \geq d_i$. Let us note that there are 40 possible constraints for a set of 5 jobs. The maximum size of the terms of the target utility function is three and it is assumed to be known. Their weights are distributed uniformly at random in the range $[1, 100]$. Similarly to the MAX-SAT case, the probability of inaccurate ranking examples swapping the correct positions of the solutions is fixed to the value 0.1.

Fig. 6 depicts the performance of our algorithm for the cases of 3, 4, 6, 7, 8, 9 terms in the target utility function. The y-axis reports the utility loss measured in terms of deviation from the cost of the gold solution, while the x-axis contains the number s of initial configurations to be ranked.

As expected, the learning problem becomes more challenging for increasing number of terms. However, the results for the scheduling problem are promising: our approach identifies the gold standard solution in all of the cases. In detail, less than 60 initial configurations are required to identify the gold solution at the second iteration. With three iterations, in the simplest cases of three up to six terms, the gold solution is recovered by using 20+10+10=40 configurations divided in three batches. When considering the more complex utility functions with seven, eight and nine terms, our algorithm needs at most 40 initial configurations for convergence to the gold solution at the third iteration. Let us note that the bounded human rationality when making decision limits the number of non-linear interactions among the user features. Human utility function with more than three/five terms are therefore quite unrealistic. Analogously to the MAX-SAT experiments, utility functions with six up to nine terms are used to test the scalability of our approach.

The plots in Fig. 7 and Fig. 8 show that at the third iteration our approach finds the gold solution consistently in

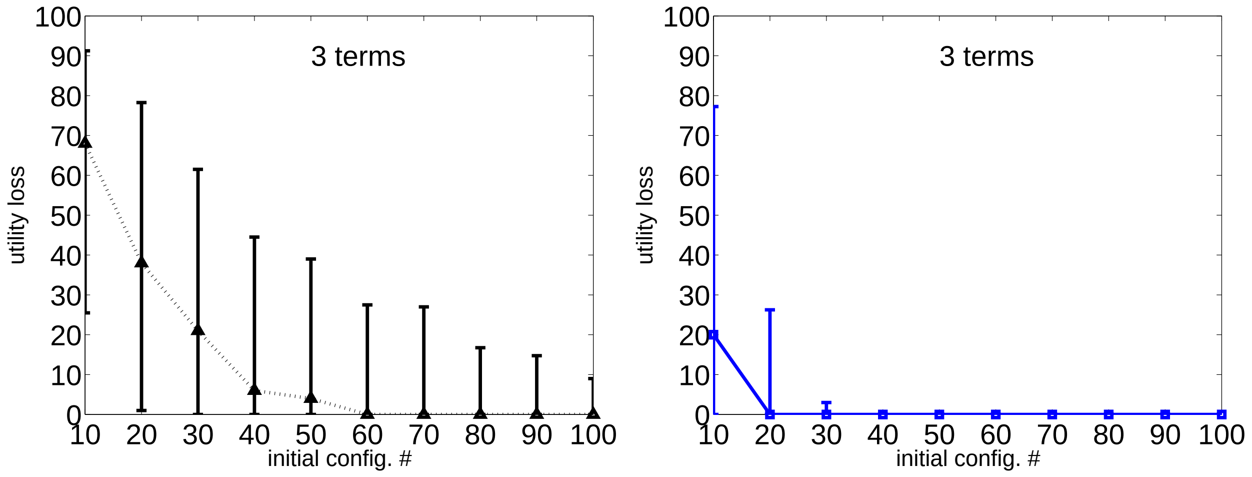

Figure 7: Learning curves at the first (left figure) and the third (right figure) iterations observed while solving the scheduling problem with three terms in the target utility function. Error bars denote the range among the 25th and the 75th percentiles of the measurements.

the case of three terms target utility functions. When considering nine terms target utility functions, the value of the 75th percentile is within 20 units from the median value, provided that at least 60 initial configurations are used. As expected, a more unstable behavior is observed at the first iteration for both three and nine terms cases.

6.2.2. Housing problem

For a second realistic application of our preference elicitation technique, we consider a customer planning to build her own house and judging potential housing locations provided by a real estate company (henceforth the *housing* problem). There are different locations available, characterized by different housing values, prices, constraints about the design of the building (e.g., usually in the city center you cannot have a family house with a huge garden and pool), etc. The customer may formulate her judgments by considering a description of the housing locations based on a predefined set of parameters, including, e.g., crime rate, distance from downtown, location-based taxes and fees, public transit service quality, cultural resources accessibility, walking and cycling facilities, etc. In addition, she is free to express her own requirements, consisting of financial issues, working opportunities, personal interests (e.g., the proximity to commercial facilities or green areas), etc. As a result, this problem is characterized by a plethora of decisional features whose contribution to the definition of the user preferences cannot be quantified in advance. Many of them may be uninformative, as they do not represent any decisional criterion for the customer. Furthermore, while specifying in advance hard constraints for the locations may be straightforward (consider, e.g., cost bounds stated by the user or building design requirements asserted by the company), assessing the user preferences in terms of the combination of this redundant set of decisional features may demand a prohibitive effort. In the real world, the elicitation process is usually driven by the sales personnel of the company in collaboration with the customer. Their joint effort identifies the customer decisional features from the catalog set and defines the (non-linear) relationships among the selected features. For example, consider the following preference information from the decision maker: "I like family houses with a big garden and I'm not interested in living near the place where I work. On the other hand, I would like a location near the school of my children. However, in the case of good price, I could accept a flat downtown, provided that commercial facilities are reachable on foot and there is free parking in the neighborhood". Finally, in order to provide satisfactory solutions to the customer, the sales personnel has to assess a rank for the (possibly conflicting) stated preferences. Considering the previous preference information statement, the sales personnel should quantify, e.g., how much a family house with big garden is preferred to a location near the children's school (or vice versa).

However, this process may often produce poor results, which do not fulfill the expectations of the user. In most cases, a complete and precise formulation of the user preferences cannot be elicited before the customer becomes aware of some possible solutions. As a result, soft constraints remain in the mind of the decision maker, and revisions of the stated preferences after seeing the actual optimization results are an inescapable fact. To complicate things,

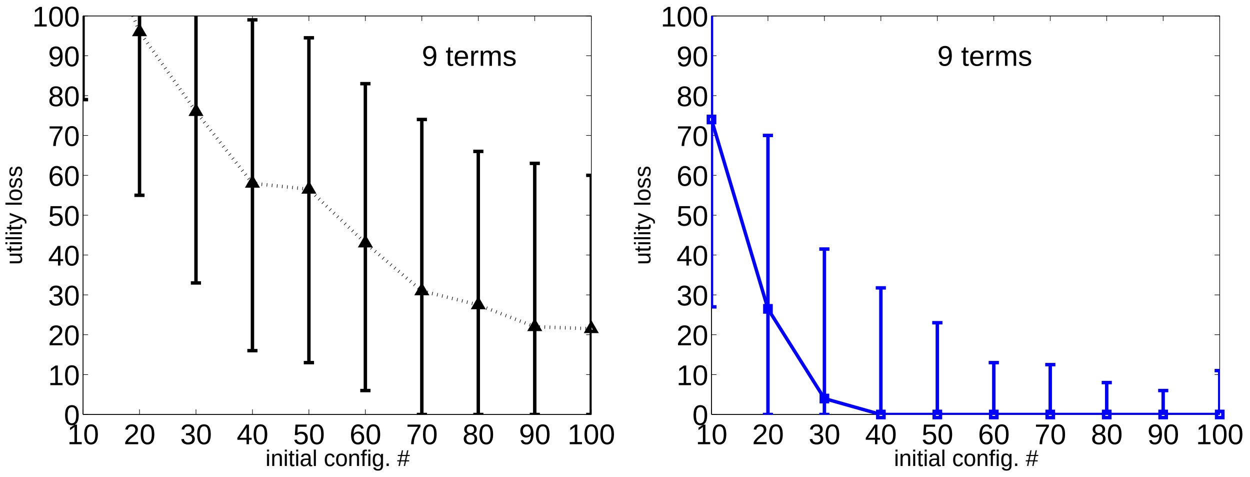

Figure 8: Learning curves at the first (left figure) and the third (right figure) iterations for scheduling problems with target utility functions of nine terms. Error bars denote the range among the 25th and the 75th percentiles of the measurements.

misunderstanding between the persons may arise and possibly imprecise and inconsistent answers of the user to the elicitation queries have to be considered. In this context, our preference elicitation technique provides a robust housing location recommendation system that can evaluate the suitability of the solutions and optimize them for the customer. On the other side, the application of the preference elicitation technique introduced in [7] is difficult, as it assumes to know in advance both the decisional features of the user and their detailed combination (represented in terms of soft constraints), while the elicitation process focuses exclusively on assessing the preferences for the different instantiations of the variables of the constraints.

In our experiments, the formulation of the housing problem is as follows. The set of catalog features is listed in Table 1. A set of 10 hard constraints (Table 2) defining feasible housing locations and known in advance is

Table 1: Decisional features for the housing problem.

num	feature	type
1	house type	ordinal
2	garden	Boolean
3	garage	Boolean
4	commercial facilities in the neighborhood	Boolean
5	public green areas in the neighborhood	Boolean
6	cycling and walking facilities in the neighborhood	Boolean
7	distance from downtown	numerical
8	crime rate	numerical
9	location-based taxes and fees	numerical
10	public transit service quality index	numerical
11	distance from high schools	numerical
12	distance from nearest free parking	numerical
13	distance from working place	numerical
14	distance from parents house	numerical
15	price	numerical

considered. The hard constraints are stated by the customer (e.g., cost bounds) or by the company (e.g, constraints about the distance of the available locations from user-defined points of interest). Let us note that constraints 5, 6, 7 define a linear bi-objective problem among distances from user-defined points of interest. Prices of potential housing locations are defined as a function of the other features. For example, price increases if a semi-detached house rather than a flat is selected or in the case of green areas in the neighborhood. On the other side, e.g., when crime index of potential locations increases, price decreases. Soft constraints are represented by weighted terms including predicates in the linear arithmetic theory or Boolean variables, in the case of features number $2, 3, \ldots, 6$ in Table 1. For example, one predicate may model the preference for a location with distance from nearest free parking smaller than a given threshold, while a Boolean variable encodes, e.g., the aspiration for houses with garage.

Table 2: hard feasibility constraints for the housing problem. Parameters ρ_i, $i = 1 \ldots 13$, are threshold values specified by the user or by the sales personnel, depending on who states the hard constraint which they refer to.

num	hard constraint
1	price $\leq \rho_1$
2	location-based taxes and fees $\leq \rho_2$ => *not* public green ares in the neighborhood *and not* public transit service quality index $\leq \rho_3$
3	commercial facilities in the neighborhood => *not* (garden *and* garage)
4	crime rate $\leq \rho_4$ => distance from downtown $\geq \rho_5$
5	distance from working place + distance from parents house $\geq \rho_6$
6	distance from working place + distance from high schools $\geq \rho_7$
7	distance from parents house + distance from high schools $\geq \rho_8$
8	distance from nearest free parking $\leq \rho_9$ => *not* public green areas in the neighborhood
9	distance from parents house $\leq \rho_{10}$ => distance from downtown $\geq \rho_{11}$ *and* crime rate $\geq \rho_{12}$
10	garden => house type $\geq \rho_{13}$

We generated a set of 40 predicates. The target utility function is composed of terms with two or three predicates, with at least one term with three predicates. Term weights are integer values selected uniformly at random in the range [1, 100]. Inaccurate preference information can be due to occasional inattention of the DM which with probability 0.1 swaps the correct positions of the solutions in the ranking examples.

Fig. 9 reports the results over a benchmark of 400 randomly generated utility functions for each of the following instantiation of the triplet (*number of features, number of terms, max term size*): $\{(5, 3, 3), (6, 4, 3), (7, 6, 3), (8, 7, 3), (9, 8, 3), (10, 9, 3)\}$. The promising results observed for the scheduling problem are confirmed. A stable behavior is observed for our approach at the third iteration: the quality of the solution rapidly improves with a larger number of examples and the algorithm succeeds in exploiting its active learning strategy. As a consequence, the gold solution is quickly identified.

Fig. 10 and 11 show the substantial stability improvement achieved by three iterations over a single one for the three and nine term problems respectively, confirming the effectiveness of our incremental approach.

7. Discussion

We presented an interactive optimization strategy for combinatorial problems over an unknown utility function. The algorithm alternates a search phase using the current approximation of the utility function to generate candidate solutions, and a refinement phase exploiting feedback received to improve the approximation. We introduced a generic framework, enabling the adoption of well-assessed learning methods and MAX-SMT solvers. 1-norm regularization is employed to enforce sparsity of the learned function. The DM is asked to rank the solutions optimizing the generated weighted MAX-SMT instance. Thanks to the MAX-SMT formalism, our approach can handle a large class of relevant optimization tasks. Experimental results on both weighted MAX-SAT and MAX-SMT problems demonstrate the effectiveness of our approach in focusing towards the optimal solutions, its robustness, as well as its ability to recover

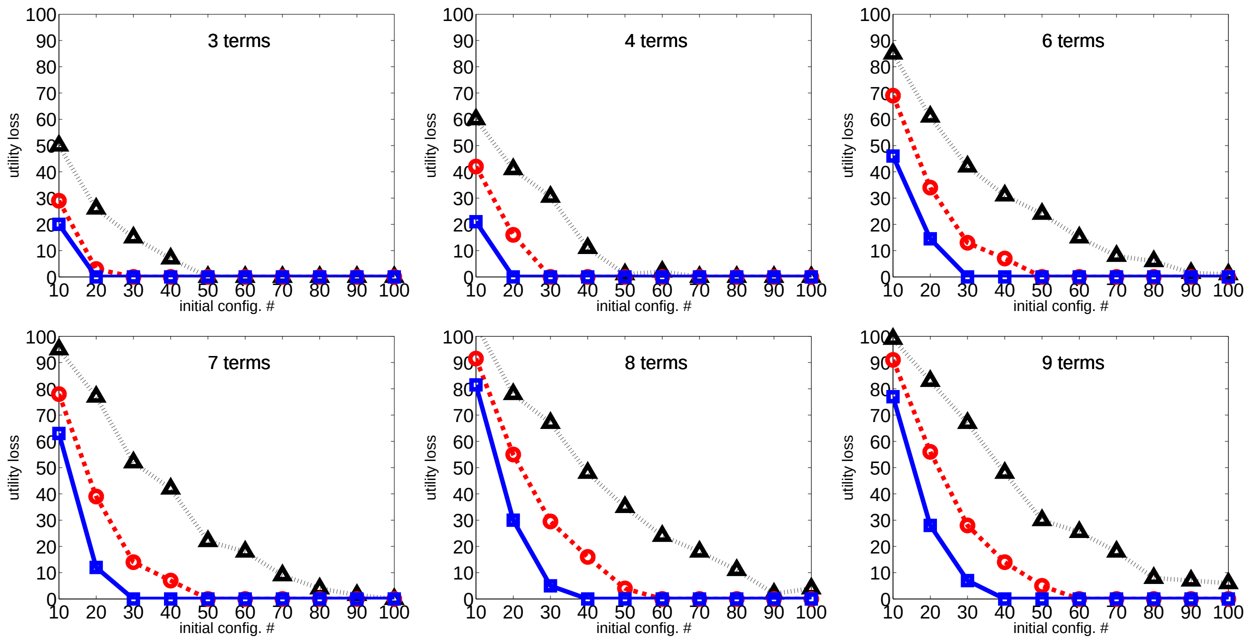

Figure 9: Learning curves observed at different iterations of our algorithm while solving the Housing problem. The data are presented analogously to that in Fig. 6.

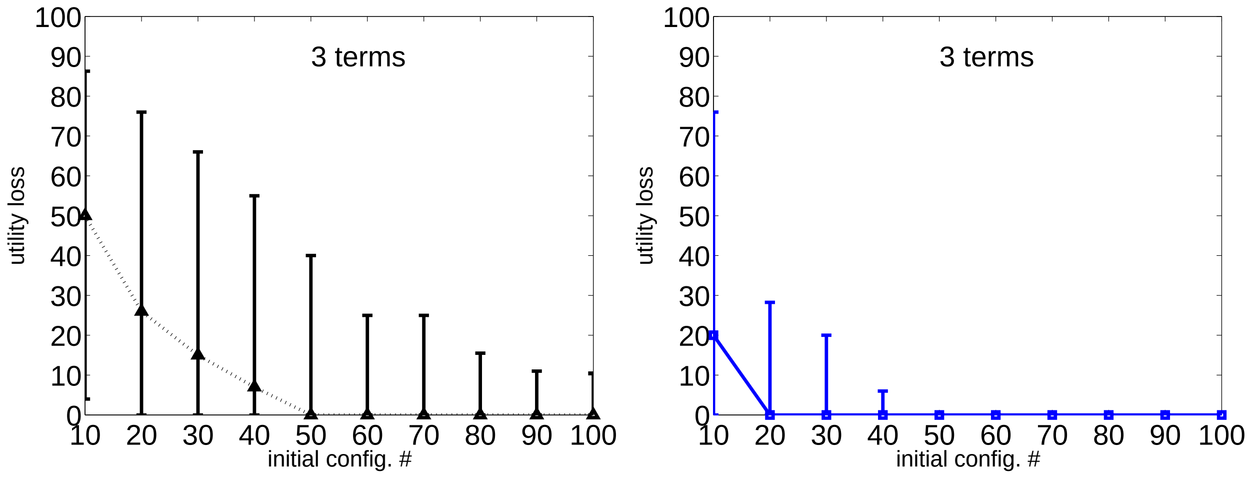

Figure 10: Learning curves at the first (left figure) and the third (right figure) iterations obtained while solving the Housing problem with target utility function of three terms. Error bars represents the range among the 25th and the 75th percentiles of the measurements.

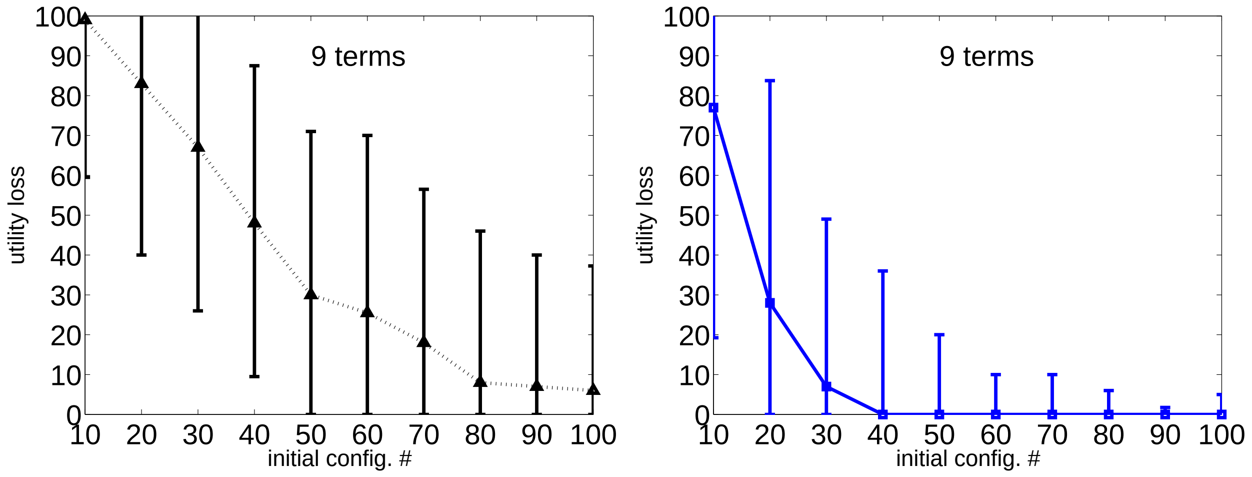

Figure 11: Learning curves at the first (left figure) and the third (right figure) iterations obtained while solving the Housing problem with target utility function of nine terms. Error bars represents the range among the 25th and the 75th percentiles of the measurements.

from suboptimal initial choices. Our tests include a preference elicitation task with both known hard constraints limiting the set of feasible solutions and unknown user preferences. In the housing problem the hard constraints define the available house locations and the preferences of the DM drive the search within the set of feasible solutions.

The algorithm introduced in this work can be generalized in a number of directions. The learning stage employs a ranking loss function based on pairwise preference evaluation. More complex ranking losses have been proposed in the literature (see for instance [31]), especially to increase the importance of correctly ranking the highest scoring solutions, and could be combined with 1-norm regularization.

Active learning is a hot research area and a broad range of different approaches has been proposed (see [32] for a review). The simplest and most common framework is that of *uncertainty sampling*: the learner queries the instances on which it is least certain. However, the ultimate goal of a recommendation or optimization system is selecting the best instance(s) rather than correctly modeling the underlying utility function. The query strategy should thus tend to suggest good candidate solutions and still learn as much as possible from the feedback received. Typical areas where research on this issue is quite popular are single- and multi-objective interactive optimization [1] and information retrieval [33]. The need to trade-off multiple requirements in this active learning setting is addressed in [34] where the authors consider relevance, diversity and density in selecting candidates. Our future research will consider the application of these active learning techniques. The performance of our method indeed depends on the trade-off between the identification of candidates solutions satisfying the DM (i.e., solutions optimizing the current learnt preference model) and the generation of *informative* training examples for the following refinement of the learnt model.

Bayesian preference elicitation models [23, 15, 24, 25] express the knowledge about the DM preferences in terms of probabilistic beliefs. The uncertainty about the user preferences is expressed by a probability distribution, or beliefs, over the space W of candidate utility functions. In this probabilistic uncertainty setting, the expected utility of a configuration is defined as the average utility with respect to the probability distribution over the utility functions and the configuration maximizing the expected utility is considered as the best configuration to be recommended to the user. This choice enables a robust approach that minimizes risk in expectation. The beliefs about the candidate utility functions are refined by Bayesian inference. Given a training dataset D, Bayesian inference requires the specification of both a prior $p(w)$ over the utility functions w and a likelihood model $p(D|w)$ of the data given the utility function. However, depending on the selected prior and likelihood formulations, Bayesian inference may not be analytically tractable and approximation techniques have to be adopted [24]. Furthermore, the quality of the approximation has to be traded off against the strict time constraints characterizing interactive preference elicitation [24, 15]. In the context of preference elicitation, Bayesian approaches are attractive as they quantify the uncertainty in the learnt DM utility models and provide a principled approach to estimate the value of the information obtained by asking a certain query to the DM. In particular, the value of the information estimates the extent to which a certain query helps in improving the quality of the learnt preference model. The value of information is exploited to design efficient query strategies consisting of informative queries (see, e.g, the approach in [15] for an example). Adapting these concepts to our setting, where the utility function is a complex formula over one or more theories and is optimized by specific solvers, is an interesting and challenging direction for future work.

Another research direction is the extension of our approach to handle feedback from multiple DMs [35]. In particular, an interesting case study is the exploitation of preferences of previous DMs to minimize the elicitation effort for a new user [23, 25]. We also plan the extension of our algorithm to tackle preference drift [36], i.e., the tendency of the decision maker to change her preferences during the interactive utility elicitation stage. As a matter of fact, the DM has typically a limited initial knowledge of the problem at hand and only when seeing the actual tentative solutions she becomes aware of “what is possible”. Confronted with this new knowledge, her preferences may thus evolve. Furthermore, the DM may not be aware of her preference changes and may not explicitly alert the utility elicitation system. In our combinatorial utility settings, the DM preference drift can be modeled by weights of soft constraints evolving over time and by logic formulas gradually changing (e.g., the Boolean term $x_1 \wedge x_2$ becoming $x_1 \wedge x_2 \wedge x_4$ when the DM realizes to have a more complex requirement).

Finally, in this paper the experimental evaluation is focused on small-scale problems, typical of an interaction with a human DM. In principle, when combined with appropriate SMT solvers, our approach could be applied to larger real-world optimization problems, whose formulation is only partially available. In this case, a *local search* algorithm rather than a complete solver will be used during the optimization stage. However, the cost of requiring an explicit representation of all possible conjunction of predicates (even if limited to the unknown part) would rapidly produce an explosion of computational and memory requirements. An option consists of resorting to an implicit representation of the function to be optimized, like the kernelized one we used in [11] when learning quantitative scores. As our previous results seem to confirm [11], this can produce a degradation in the quality of returned solutions when the utility function is very sparse. Kernelized versions of zero-norm regularization [37] could be tried in order to enforce sparsity in the projected space if needed. Let us note however that the lack of an explicit formula would prevent the use of all the efficient refinements of SMT solvers, based on a tight integration between SAT and theory solvers. A possible alternative is that of pursuing an incremental feature selection strategy and iteratively solving increasingly complex approximations of the underlying problem.

Acknowledgments

This research was partially supported by grant PRIN 2009LNP494 (Statistical Relational Learning: Algorithms and Applications) from Italian Ministry of University and Research.

Appendix A. Econding SCSP into weighted MAX-SAT instances

The work in [28] introduces a method to encode a semiring-based soft constraint satisfaction problem (SCSP) instance into a weighted MAX-SAT instance, with each solution of the generated MAX-SAT instance corresponding to a solution of the original SCSP. With no loss of generality, assume a soft constraint problem with n variables $v_1, \ldots v_n$ having domain $D_1, \ldots D_n$, and m constraints $c_1, \ldots c_m$. Each instantiation of the variables of a constraint c_j, $j = 1 \ldots m$, is associated with a value from the c-semiring $(A, +, \times, \mathbf{0}, \mathbf{1})$. For each variable v_i, $i = 1 \ldots n$, and each value $d \in D_i$, a Boolean variable $b_{i,d}$ is introduced. When $b_{i,d}$ is set to true then v_i is assigned the value $d \in D_i$. The variables $b_{i,d}$, $i = 1 \ldots n$, $d \in D_i$, represent the Boolean variables of the weighted MAX-SAT problem.

The set of Boolean constraints of the MAX-SAT problem consists of clauses ensuring that each variable v_i, $i = 1 \ldots n$, is assigned exactly one value $d \in D_i$, and of terms representing the soft constraints of the original SCSP. In the former case, for each variable v_i, $i = 1 \ldots n$, the *at-least-one-value* hard clause:

$$(b_{i,d_1} \vee b_{i,d_2} \vee \ldots \vee b_{i,d_{|D_i|}})$$

and the set of $(|D_i|\dot{(}|D_i| - 1))/2$ binary *at-max-one-value* hard clauses:

$$(\neg b_{i,d_j} \vee \neg b_{i,d_k}) \text{ for every pair } (d_j, d_k) \text{ with } d_j, d_k \in D_i \text{ and } 1 \leq j < k \leq |D_i|$$

v_1	v_2	preference value
1	1	10
1	2	40
1	3	50
2	1	5
2	2	10
2	3	30
3	1	5
3	2	5
3	3	10

num	term	weight
1	$(b_{1,1} \wedge b_{2,1})$	10
2	$(b_{1,1} \wedge b_{2,2})$	40
3	$(b_{1,1} \wedge b_{2,3})$	50
4	$(b_{1,2} \wedge b_{2,1})$	5
5	$(b_{1,2} \wedge b_{2,2})$	10
6	$(b_{1,2} \wedge b_{2,3})$	30
7	$(b_{1,3} \wedge b_{2,1})$	5
8	$(b_{1,3} \wedge b_{2,2})$	5
9	$(b_{1,3} \wedge b_{2,3})$	10

Table A.3: (left) example of soft constraint. The DM prefers assignments with $v_1 < v_2$. (Right) weighted Boolean terms encoding the soft constraint defined in the left table. When the Boolean variable $b_{i,d} : d \in D_i$, is set to true then v_i is assigned the value d.

are generated. They ensure that for each $i \in \{1 \ldots n\}$ exactly one variable $b_{i,j},\ j \in \{1, 2, \ldots, |D_i|\}$ is set to true.

Each soft constraint of the original SCSP is represented by a set of weighted Boolean terms encoding all the possible assignments of values (i.e., configurations) to its variables. The weight of a term is set to the c-semiring value associated to the encoded configuration. For example, consider a binary soft constraint over variables v_1 and v_2 both with discrete domain $D = \{1, 2, 3\}$ and with preference scores defined by the semiring $(\{5, 10, 15, \ldots, 50\}, \max, \min, 5, 50)$. The possible configurations are specified in Table A.3 (left). Each row shows an assignment of values to v_1 and v_2 and the c-semiring value associated to the assignment. Given the six Boolean variables $b_{1,d}$ and $b_{2,d}$ with $d = 1, 2, 3$ defined as above, the soft constraint in Table A.3 (left) is encoded into the set of Boolean terms in Table A.3 (right).

A structured MAX-SAT formulation can be obtained by considering generalized Boolean clauses which are the disjunction of the terms encoding for a given soft constraint the assignments with the same preference value. For example, the terms defined at rows number 1, 5, 9 in Table A.3 (right) can be merged into a single generalized weighted clause:

$$(b_{1,1} \wedge b_{2,1}) \vee (b_{1,2} \wedge b_{2,2}) \vee (b_{1,3} \wedge b_{2,3})$$

with weight equal to 10. Furthermore, each *at-least-one-value* and *at-max-one-value* hard clause h can be cast into a soft clause represented by its negation $\neg h$ and with associated the semiring value $\mathbf{0}$ [28]. The value $\mathbf{0}$ is indeed both the minimum value in the partial order defined by the relation $\leq_A$ and the absorbing element for the operator $\times$ combining the semiring values. Therefore, a candidate solution $\mathbf{b}$ of the generated MAX-SAT instance that does not satisfy one of these soft clauses receives the minimum semiring value $\mathbf{0}$. However, this implementation of the hard clauses does not allow to discern infeasible solutions from feasible ones with lowest possible preference, i.e., feasible solutions getting the lowest semiring value.

Given the generated MAX-SAT formulation, the optimization task consists of finding the assignment $\mathbf{b}^*$ to the Boolean variables $b_{i,d},\ i = 1 \ldots n,\ d \in D_i$, maximizing $f(\mathbf{b})$, with $f(\mathbf{b})$ the semiring value obtained by combining by the operator $\times$ the weights of the solution components satisfied by $\mathbf{b}$. Each candidate solution $(\mathbf{b},\ f(\mathbf{b}))$ of the generated MAX-SAT instance identifies an assignment of values to the variables $v_i,\ i = 1 \ldots n$, of the original SCSP with associated semiring value $f(\mathbf{b})$.

References

[1] J. Branke, K. Deb, K. Miettinen, R. Słowiński (Eds.), Multiobjective Optimization: Interactive and Evolutionary Approaches, Springer Verlag, 2008.

[2] R. Battiti, A. Passerini, Brain-Computer Evolutionary Multiobjective Optimization: A Genetic Algorithm Adapting to the Decision Maker, IEEE Transactions on Evolutionary Computation 14 (2010) 671–687.

[3] F. Bacchus, A. Grove, Graphical Models for Preference and Utility, in: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95), Morgan Kaufmann, 1995, pp. 3–10.

[4] D. Braziunas, C. Boutilier, Minimax regret based elicitation of generalized additive utilities, in: Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence (UAI-07), Vancouver, pp. 25–32.

[5] C. Boutilier, K. Regan, P. Viappiani, Simultaneous Elicitation of Preference Features and Utility, in: Proceedings of the Twenty-fourth AAAI Conference on Artificial Intelligence (AAAI-10), AAAI press, Atlanta, GA, USA, 2010, pp. 1160–1167.

[6] C. Boutilier, R. Patrascu, P. Poupart, D. Schuurmans, Constraint-based Optimization and Utility Elicitation using the Minimax Decision Criterion, Artificial Intelligence 170 (2006) 686–713.

[7] M. Gelain, M. S. Pini, F. Rossi, K. B. Venable, T. Walsh, Elicitation Strategies for Soft Constraint Problems with Missing Preferences: Properties, Algorithms and Experimental Studies, Artificial Intelligence Journal 174 (2010) 270–294.

[8] R. Nieuwenhuis, A. Oliveras, On SAT Modulo Theories and Optimization Problems, in: Theory and Applications of Satisfiability Testing, LNCS, Springer, 2006, pp. 156–169.

[9] A. Cimatti, A. Franzn, A. Griggio, R. Sebastiani, C. Stenico, Satisfiability Modulo the Theory of Costs: Foundations and Applications, in: J. Esparza, R. Majumdar (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, volume 6015 of *LNCS*, Springer, 2010, pp. 99–113.

[10] C. Barrett, R. Sebastiani, S. A. Seshia, C. Tinelli, Satisfiability Modulo Theories, in: Handbook of Satisfiability, IOS Press, 2009, pp. 825–885.

[11] P. Campigotto, A. Passerini, R. Battiti, Active Learning of Combinatorial Features for Interactive Optimization, in: Proceedings of the 5th Learning and Intelligent OptimizatioN Conference (LION V), Rome, Italy, Jan 17-21, 2011, LNCS, Springer Verlag, 2011.

[12] P. Pu, L. Chen, User-Involved Preference Elicitation for Product Search and Recommender Systems, AI magazine 29 (2008) 93–103.

[13] R. Tibshirani, Regression Shrinkage and Selection Via the Lasso, Journal of the Royal Statistical Society, Series B 58 (1996) 267–288.

[14] J. Friedman, T. Hastie, S. Rosset, R. Tibshirani, Discussion of boosting papers, Annals of Statistics 32 (2004) 102–107.

[15] S. Guo, S. Sanner, Real-time Multiattribute Bayesian Preference Elicitation with Pairwise Comparison Queries, Journal of Machine Learning Research - Proceedings Track 9 (2010) 289–296.

[16] W. W. Cohen, R. E. Schapire, Y. Singer, Learning to order things, Journal of Artificial Intelligence Research 10 (1999) 243–270.

[17] M. Collins, N. Duffy, Convolution kernels for natural language, in: Advances in Neural Information Processing Systems 14, MIT Press, 2001, pp. 625–632.

[18] L. De Moura, N. Bjorner, Satisfiability Modulo Theories: An Appetizer, in: Formal Methods: Foundations and Applications, LNCS, Springer, 2009, pp. 23–36.

[19] G. Nelson, D. C. Oppen, Simplification by Cooperating Decision Procedures, ACM Transactions on Programming Languages and Systems 1 (1979) 245–257.

[20] B. Dutertre, L. de Moura, A Fast Linear-Arithmetic Solver for DPLL(T), in: Proceedings of the 18th Computer-Aided Verification conference, LNCS, Springer, 2006, pp. 81–94.

[21] C. Domshlak, E. Hüllermeier, S. Kaci, H. Prade, Preferences in AI: An overview, Artificial Intelligence 175 (2011) 1037–1052.

[22] D. Braziunas, Computational Approaches to Preference Elicitation, Technical Report, Department of Computer Science, University of Toronto, 2006.

[23] E. Bonilla, S. Guo, S. Sanner, Gaussian Process Preference Elicitation, in: J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, A. Culotta (Eds.), Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems, 2010, pp. 262–270.

[24] P. Viappiani, Monte Carlo Methods for Preference Learning, in: Proceedings of the 6th Learning and Intelligent OptimizatioN Conference (LION VI), LNCS, Springer Verlag, Paris, France, 2012.

[25] A. Birlutiu, P. Groot, T. Heskes, Efficiently learning the preferences of people, Machine Learning (2012) 1–28.

[26] S. Bistarelli, U. Montanari, F. Rossi, Semiring-based Constraint Solving and Optimization, Journal of ACM 44 (1997) 201–236.

[27] S. Bistarelli, M. S. Pini, F. Rossi, K. B. Venable, From soft constraints to bipolar preferences: modelling framework and solving issues, Journal of Experimental and Theoretical Artificial Intelligence 22 (2010) 135–158.

[28] L. Leenen, Anbulagan, T. Meyer, A. K. Ghose, Modeling and Solving Semiring Constraint Satisfaction Problems by Transformation to Weighted Semiring Max-SAT, in: 20th Australian Joint Conference on Artificial Intelligence, volume 4830 of *LNCS*, Springer, 2007, pp. 202–212.

[29] C. P. Gomes, H. Kautz, A. Sabharwal, B. Selman, Satisfiability Solvers, in: Handbook of Knowledge Representation, volume 3 of *Foundations of Artificial Intelligence*, Elsevier, 2008, pp. 89–134.

[30] M. Gelain, M. Pini, F. Rossi, K. Venable, N. Wilson, Interval-valued soft constraint problems, Annals of Mathematics and Artificial Intelligence 58 (2010) 261–298.

[31] S. Chakrabarti, R. Khanna, U. Sawant, C. Bhattacharyya, Structured learning for non-smooth ranking losses, in: 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08, ACM, 2008, pp. 88–96.

[32] B. Settles, Active Learning Literature Survey, Technical Report Computer Sciences Technical Report 1648, University of Wisconsin-Madison, 2009.

[33] F. Radlinski, T. Joachims, Active exploration for learning rankings from clickthrough data, in: 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07), ACM Press, 2007, pp. 570–579.

[34] Z. Xu, R. Akella, Y. Zhang, Incorporating Diversity and Density in Active Learning for Relevance Feedback, in: G. Amati, C. Carpineto, G. Romano (Eds.), Advances in Information Retrieval, volume 4425 of *LNCS*, Springer, 2007, pp. 246–257.

[35] Y. Yan, R. Rosales, G. Fung, J. Dy, Active Learning from Crowds, in: L. Getoor, T. Scheffer (Eds.), Proceedings of the 28th International Conference on Machine Learning (ICML-11), ACM, New York, NY, USA, 2011, pp. 1161–1168.

[36] P. Campigotto, A. Passerini, R. Battiti, Handling concept drift in preference learning for interactive decision making, in: Online proceedings of the 1st International Workshop on Handling Concept Drift in Adaptive Information Systems (HaCDAIS 2010), Barcelona, Spain, Sept 24, 2010.

[37] J. Weston, A. Elisseeff, B. Schölkopf, M. Tipping, Use of the zero norm with linear models and kernel methods, Journal of Machine Learning Research 3 (2003) 1439–1461.