Learning Modulo Theories for constructive preference elicitation

Paolo Campigotto[a,*], Stefano Teso[b], Roberto Battiti[b], Andrea Passerini[b]

[a] *Gradient Zero GmbH,*
Grünbergstraße 15, 1120 Vienna, Austria.
[b] *DISI - Dipartimento di Ingegneria e Scienza dell'Informazione,*
Università degli Studi di Trento,
Via Sommarive 5, I-38123 Povo, TN, Italy.

Abstract

This paper introduces CLEO, a novel preference elicitation algorithm capable of recommending complex configurable objects characterized by both discrete and continuous attributes and constraints defined over them. While existing preference elicitation techniques focus on searching for the best instance in a database of candidates, CLEO takes a *constructive* approach to recommendation through interactive optimization in a space of feasible configurations. The algorithm assumes minimal initial information, i.e., a set of catalog attributes, and defines decisional features as logic formulae combining Boolean and algebraic constraints over the attributes. The (unknown) utility of the decision maker is modelled as a weighted combination of features. CLEO iteratively alternates a preference elicitation step, where pairs of candidate configurations are selected based on the current utility model, and a refinement step where the utility is refined by incorporating the feedback received. The elicitation step leverages a Max-SMT solver to return optimal configurations according to the current utility model. The refinement step is implemented as learning to rank, and a sparsifying norm is used to favour the selection of few informative features in the combinatorial space

*The main part of this work has been done while the author was with DISI, the remaining part while the author was with the Informatik Lehrstuhl 4, Technische Universität Dortmund, Otto-Hahn-Str. 16, D-44227 Dortmund, Germany.

Email addresses: `paolo.campigotto@yahoo.com` (Paolo Campigotto), `stefano.teso@unitn.it` (Stefano Teso), `battiti@disi.unitn.it` (Roberto Battiti), `passerini@disi.unitn.it` (Andrea Passerini)

of candidate decisional features.

A major feature of CLEO is that it can recommend optimal *configurations* in hybrid domains (i.e., including both Boolean and numeric attributes), thanks to the use of Max-SMT technology, while retaining uncertainty in the decision-maker's utility and noisy feedback. In so doing, it adapts the recently introduced learning modulo theory framework to the preference elicitation setting. The combinatorial formulation of the utility function coupled with the feature selection capabilities of 1-norm regularization allow to effectively deal with the uncertainty in the DM utility while retaining high expressiveness. Experimental results on complex recommendation tasks show the ability of CLEO to quickly identify optimal configurations, as well as its capacity to recover from suboptimal initial choices. Our empirical evaluation highlights how CLEO outperforms a state-of-the-art Bayesian preference elicitation algorithm when applied to a purely discrete task

Keywords: preference elicitation, learning while optimizing, (Maximum) Satisfiability Modulo Theory, constructive machine learning.

---

1. Introduction

Automatically discovering the solution preferred by a decision maker (DM) from a large set of candidate ones is a key component of many systems, including decision-support, recommendation algorithms and personal agents. This task is usually referred to as *preference elicitation* [1]. In principle, one could first ask the user to express her preferences and then translate them into a utility function defined over the search space of candidate solutions. The solution maximizing the utility function would then be recommended to the DM. However, this approach is impractical, for several reasons [2]:

- The user cannot usually define her preferences upfront, without seeing any tentative recommendations. Only after seeing and evaluating a few candidate solutions, she may realize "what is possible" and articulate her actual objectives;
- The cognitive effort and the time required to the user for completely specifying her preferences are usually not affordable;
- More generally, formalizing the user's preferences as a mathematical model is not trivial: a model should capture the qualitative notion of preference and represent it as a quantitative function.

A better alternative is to incrementally refine an initially incomplete model of the user's utility by employing an iterative strategy, where a solution is recommended to the user based on *partial* preference information [1, 3, 4]. If the user is not satisfied by the tentative solution, she is asked for additional preference information and a refined solution is suggested. This incremental process needs techniques that can reason with partially-specified utility functions and take decisions under uncertain preference information. Furthermore, an efficient approach must elicit as few preference information as possible to identify the DM preferred configuration. Indeed, human decision makers have limited patience and bounded rationality. This limits the number and the complexity of queries asked during the elicitation process, bounds the time needed for providing recommendations, and forces the recommender to deal with inaccurate and inconsistent feedback. The main requirements for practical applicability of preference elicitation are [5]:

1. *real-time* interaction with the DM, where both the query generation and the solution recommendation must be accomplished in the order of few seconds;
2. *multi-attribute*: the number of options can be vastly larger than the number of queries that can be reasonably asked to the DM; multi-attributes utilities are crucial in generalizing local preferences over the product catalogue;
3. *cognitively affordable* queries to the user, i.e., comparison queries;
4. *robustness* to inconsistent feedback from the DM characterizing the typical human decision making process;
5. *scalable* methods capable of dealing with high-dimensional solution spaces.

Different approaches to preference elicitation have been proposed. Usually, a *parametric* formulation of the space of possible DM utility functions is adopted. A set of basis functions are defined on subsets of the attributes, and the utility model is formulated as a weighted linear combination of these basis functions. Approaches to preference elicitation can be classified by the different ways of making recommendations under uncertainty. Uncertainty in DM utility can be represented for instance by defining a space of *feasible* weights, identified by bounds or constraints on the values. These constraints are learned from the preference information elicited from the DM. This popular approach, known in the literature as *reasoning under strict uncertainty*, is adopted in [6, 7, 8]. In these papers, decisions under uncertainty are taken

according to the *minimax regret criterion*: the solution minimizing the worst-case loss with respect to the feasible utility functions is recommended. On the other hand, *Bayesian* approaches maintain a probability distribution over weight values [9, 5, 10, 11]. Decisions are taken according to this probability distribution: the recommended solution is usually the one with greatest *expected utility*. Recent work in the field of constraint programming [12] 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. The work in [12] introduces a preference elicitation strategy for soft constraint problems with missing preference values.

A major limitation of existing approaches is that they are focused on finding the best solution among a set of available candidates. While this setting is reasonable for several applications, like movie or restaurant recommendation, it becomes infeasible in fully *constructive* scenarios. Here the solution to be recommended requires arranging a set of components into a *configuration*, subject to some constraints defining the feasible space. Examples include customizing laptops, adapting recipes or insurance packages, and many other cases with a combinatorial explosion of candidate configurations. Existing approaches rely on an exhaustive enumeration of candidates for both recommendation and preference elicitation, which makes them unusable for these applications. The problem gets even worse in *hybrid* domains, characterized by both Boolean and numeric attributes, like designing an apartment or choosing its furniture [13], and more generally addressing layout synthesis problems [14, 15]. In this case the number of candidate configurations is virtually infinite (depending on whether it is reasonable to discretize continuous attributes) and one needs to formalize the task as a configuration optimization problem.

This paper introduces a novel algorithm capable of performing constructive preference elicitation in hybrid domains by casting the problem into a "learning to optimize" framework, while retaining all the main principles for practical applicability of preference elicitation. The approach adopts a combinatorial formulation of the user utility function, modelled as a weighted combination of first-order logic formulae. Each formula combines predicates in a certain theory of interest, like linear inequalities for numeric attributes, by using the logical connectives. 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). For example, consider the case of flight selection.

The predicate $\varphi_1 = (A_1 + A_2 \leq 5\ hours)$ defines the preference for a travel duration, calculated as flight duration (continuous attribute A_1) plus transfer time to the departure airport (A_2), smaller than five hours. The predicate $\varphi_2 = (A_3 < 2)$ states the desirability for a flight with a number of stopovers (discrete attribute A_3) smaller than two. The DM preferences about the candidate flights are expressed by associating the two predicates φ_1 and φ_2 with weights w_1 and w_2, respectively[1]. The flight maximizing the sum of the weights of the satisfied predicates is the one preferred by the DM.

The configuration maximizing the weighted combinations of the first-order logic formulae is identified by applying a weighted *Maximum Satisfiability Modulo Theory* (Max-SMT) solver [16]. Max-SMT is a powerful formalism to optimize weighted formulae in a decidable first-order theory. Max-SMT enables to describe candidate configurations of the preference elicitation task by using both discrete and continuous attributes simultaneously (*hybrid search domain*), and define costs in terms of weighted formulas over these attributes. A limitation of the MAX-SMT technology is that costs cannot incorporate continuous functions of the attributes (e.g., how much more than 5 hours this travel option takes). While we focus on MAX-SMT in this paper, our approach is more general and can be combined with solvers allowing from more expressive cost functions (e.g., OMT solvers [17]).

The approach presented in this paper assumes very limited prior information about the task to be solved. The initial knowledge is given by a set of *catalog attributes* used to describe the candidate configurations. The combinatorial formulation of the DM utility over the catalog attributes is initially unknown and needs to be learned by interacting with her. For this purpose, our approach consists of an iterative algorithm, alternating a preference elicitation step guided by the currently learned utility function and a refinement step where the quality of the utility function is improved according to the feedback received. In the preference elicitation step, two candidate configurations are selected according to the current utility and presented to the DM for comparison. The refinement step consists of solving a ranking problem which outputs a refined utility function consistent with the feedback received (soft consistency is allowed to deal with noisy feedback). The feature space of the utility function is given by all possible conjunctions of predicates up to

[1]In this simple example, each formula consists of a single predicate only. In the general case, composite logic formulae are considered.

a certain degree. Only a small fraction of these candidate features is actually part of the unknown utility for a certain DM [18]. A sparsifying norm [19] is used during training to favour utility functions with few non-zero weights, thus performing constraint selection in the combinatorial space of candidate features. In the rest of this paper the algorithm is referred to by the acronym CLEO, which stands for Combinatorial utility function joint LEarning and Optimization.

An experimental evaluation on realistic hybrid problems with inaccurate human feedback demonstrates the effectiveness of CLEO in focusing on an optimal configuration, its robustness to noisy learning signals and its ability to recover from suboptimal initial choices. While no competitors exist in the general case of hybrid domains, we provide an experimental comparison on the simplified task of learning purely Boolean combinatorial functions. Thanks to its ability to learn complex non-linear interactions between attributes, CLEO outperforms a state-of-the-art Bayesian preference elicitation approach [5]. Furthermore, the run time needed by CLEO to perform the preference elicitation and refinement steps is negligible when compared with the user response time.

Learning modulo theories was recently introduced [20] as a framework for adapting structured-output learning to hybrid domains by leveraging Max-SMT technology. This paper adapts the framework to deal with preference elicitation tasks, by incorporating Max-SMT in the query selection component generating (soft) constraints for the utility learning algorithm. A preliminary version of CLEO was presented in [21]. This manuscript extends it in a number of directions. First, it replaces quantitative judgments asked to the DM with less cognitive demanding queries, consisting of pairwise preferences of candidate configurations. Second, it extends the experimental evaluation, including a more realistic recommendation problem. Third, it provides a deeper comparison with the preference elicitation literature, and adds an experimental comparison with a state-of-the-art preference elicitation technique.

The organization of the paper is as follows. Section 2 introduces the terminology and the notation used in the paper, focusing in particular on the Max-SMT formalism. An introductory example of the preference elicitation tasks follows (Sec. 3). The CLEO algorithm is introduced in Sec. 4 and its main properties are analyzed in Sec. 5. Related work is discussed in Sec. 6, while Section 7 reports the experimental evaluation.

2. Background

This section provides the necessary background to introduce the CLEO algorithm. The Satisfiability Modulo Theory (SMT) formalism for solving decision problems over hybrid domains is explained, followed by its generalization (weighted Max-SMT) to handle optimization tasks.

Satisfiability Modulo Theory. Propositional logic considers formulae involving Boolean variables and logical connectives. The satisfiability (SAT) problem consists of deciding whether a formula in propositional logic can be satisfied by assigning truth value to the Boolean variables. Satisfiability Modulo Theory (SMT) [22, 23] extends SAT to decide about satisfiability of a first-order formula with respect to a *background theory* $\mathcal{T}$, like linear arithmetic over the reals ($\mathcal{LRA}$) or integers ($\mathcal{LIA}$), or a combination of theories. First-order logic involves variables, functions and predicates; the theory $\mathcal{T}$ fixes the interpretation of predicate and function symbols. For example, given the following SMT formula from the theory of arithmetic over integers:

$$x + y + z \leq 4$$

we are interested in deciding whether there is an assignment of integer values to the variables x, y and z satisfying the formula. In this paper, SMT($\mathcal{T}$) indicates satisfiability modulo theory $\mathcal{T}$, e.g., SMT($\mathcal{LRA}$) for satisfiability modulo linear arithmetic over the reals.

Current SMT solvers are based on the so-called *lazy* approach, where an outer SAT-solver interacts with one or more specialized $\mathcal{T}$-solvers (one for each theory) to progressively focus the search towards theory-consistent solutions or to state the unsatisfiability of the input SMT formula. We refer the reader to [23, 24] for an overview on lazy SMT solving.

Max-SMT. Max-SMT [25, 26, 27] generalizes SMT in the same way as Max-SAT does with SAT: rather than searching for an assignment satisfying the input SMT formula, one maximizes the number of satisfied constraints. The weighted version of Max-SMT associates a (typically positive) weight to each constraint, and the task is that of maximizing the weighted sum of the satisfied constraints.

Table 1 summarizes the notation used in the paper.

Symbol	Meaning
$\top$, $\bot$	Boolean values `true` and `false`
$x, y, z, \ldots$	Real variables
$A_1, A_2, \ldots, A_n$	Catalog attributes (Boolean or real variables)
$\boldsymbol{A}$	Configuration (assignment of values to all catalog attributes)
$\boldsymbol{A}^i$	i^{th} configuration
$\varphi_1, \varphi_2, \ldots, \varphi_m$	Constraints. Either atoms (Boolean attributes or predicates over real attributes, e.g., $x + y < 3$) or logical combinations thereof (e.g., $\neg \mathit{has\_car} \rightarrow \mathit{dist\_supermarket} \leq \theta$)
$\mathbb{I}_k(\boldsymbol{A})$	Indicator function for constraint φ_k over $\boldsymbol{A}$. It evaluates to one if φ_k is satisfied, to zero otherwise.
$\psi_k(\boldsymbol{A}) = \mathbb{I}_k(\boldsymbol{A})$	Feature associated to constraint φ_k
$\boldsymbol{\psi}(\boldsymbol{A})$	Feature representation of configuration $\boldsymbol{A}$
$\boldsymbol{w}$	Weights

Table 1: Explanation of the notation used throughout the text.

3. An introductory example

Consider a customer that aims at building her own house. For this purpose, she starts interacting with a real-estate agency and a construction company. A very clear-headed person could formulate a request like:

> *I would like to build a house in a safe area, close to my parents and to the kindergarten, with a garden if there are no parks nearby. Free parking lots close to my house would help. If this is not possible, a garage should be built together with the house. I would also like to live close to cycling and walking facilities. Of course, to fully enjoy these outdoor activities, the area should not be affected by air pollution. If the house location is close to my parents and to the kindergarten, cycling and walking facilities gets really necessary, since I want the option to reach my parents and bring my kids without car or public transportation. My maximum budget is 300,000 Euro.*

In addition, the real-estate company states that there are no available locations within 3 km from both kindergarten and customer's parents house. The customer desiderata can be encoded as an SMT problem as follows:

$$
\begin{aligned}
&\text{solve:}\\
&\qquad (\varphi_1 \vee \varphi_2) \wedge \varphi_3 \wedge \varphi_4 \wedge \varphi_5 \wedge \varphi_6 \wedge \varphi_7 \wedge (\varphi_8 \vee \varphi_9) \wedge\\
&\qquad (\neg\varphi_4 \vee \neg\varphi_5 \vee \varphi_6) \wedge (\varphi_{10} \vee \varphi_{11})\\
&\text{subject to:}\\
&\qquad \varphi_1 = A_1 \qquad\qquad \varphi_2 = A_2\\
&\qquad \varphi_3 = (A_3 < 2) \qquad\qquad \varphi_4 = (A_4 \leq 1)\\
&\qquad \varphi_5 = (A_5 \leq 1) \qquad\qquad \varphi_6 = A_6\\
&\qquad \varphi_7 = (A_7 < 2) \qquad\qquad \varphi_8 = A_8\\
&\qquad \varphi_9 = (A_9 \leq 1) \qquad\qquad \mathit{price}(\boldsymbol{A}) \leq 300000\\
&\qquad \varphi_{10} = (A_4 > 3) \qquad\qquad \varphi_{11} = (A_5 > 3)
\end{aligned}
$$

where the characteristics of the options are defined by the set of catalog attributes $\boldsymbol{A}$ listed in Table 2, and function *price* computes the price of option $\boldsymbol{A}$ based on the values of its attributes.

name	description	type
A_1	garden	Boolean
A_2	park nearby	Boolean
A_3	crime rate	integer
A_4	distance from parents	real
A_5	distance from kindergarten	real
A_6	cycling and walking facilities in the neighborhood	Boolean
A_7	air-pollution index	integer
A_8	garage	Boolean
A_9	distance from nearest free parking	real
A_{10}	commercial facilities in the neighborhood	Boolean
A_{11}	distance from downtown	real

Table 2: Catalog attributes for the housing example.

If none of the options available at the agency satisfies *all* constraints, above problem has no solution. Indeed, the requirement of proximity to both

parents and kindergarten ($\varphi_4 \wedge \varphi_5$) conflicts with the availability constraints ($\varphi_{10} \vee \varphi_{11}$). A more reasonable alternative consists of solving the optimization version of above problem, which maximizes the weighted sum of the satisfied constraints (i.e., a Max-SMT problem):

$$\underset{\boldsymbol{A}}{\text{argmax}} \quad \sum_{i=1}^{7} \bar{w}_i$$

subject to:

/* *soft constraints* */	/* *hard constraints* */
$\varphi_1 = (A_2 \vee A_1)$	$price(\boldsymbol{A}) \leq 300000$
$\varphi_2 = (A_3 < 2)$	$\varphi_7 = ((A_4 > 3) \vee (A_5 > 3))$
$\varphi_3 = ((A_4 \leq 1) \wedge (A_5 \leq 1))$	
$\varphi_4 = ((A_4 > 1) \vee (A_5 > 1) \vee A_6)$	/* *weights definition* */
$\varphi_5 = (A_6 \wedge (A_7 < 2))$	$\varphi_i \rightarrow (\bar{w}_i = w_i) \quad \forall i \in [1, 7]$
$\varphi_6 = (A_8 \vee (A_9 \leq 1))$	$\neg\varphi_i \rightarrow (\bar{w}_i = 0) \quad \forall i \in [1, 7]$

where each soft constraint φ_i is associated to a weight w_i quantifying the (relative) importance of the constraint. The bound on the price and the availability constraint (φ_7) are hard constraints that need to be satisfied, thus they have no weight.

A fully specified scenario like the one described is however not realistic when a human DM is involved. An exact specification of the set of relevant constraints is hard to obtain, let alone their respective weights. A possible solution consists of an interactive process, with the customer evaluating candidate options and the realtor updating her understanding of the customer preferences according to the feedback received. The rest of this paper introduces the CLEO algorithm, a preference elicitation method that automates this process.

Let us finally note that not all the catalog attributes describing candidate options may be relevant for a customer: in above example the customer decides without considering the last two attributes in Table 2. A large list of catalog attributes enables both a fine-grained description of the locations and the interaction with different classes of customers, having different decisional items. On the other hand, users are expected to take decisions based on a

limited set of attributes in the large catalogue. This is even more evident when considering constraints involving combinations of attributes. CLEO incorporates an implicit feature selection mechanism to account for these aspects, as will be shown in the next section.

4. The CLEO algorithm

This section introduces the CLEO algorithm, first describing its components and then combining them into the overall algorithm.

Configuration space. CLEO assumes a *catalogue* of attributes which can be used to describe the configurations. These attributes can be either Boolean (e.g., `park nearby`), ordinal (`crime rate`) or real (`distance to parents`) variables. See the examples in Table 2. We define by $\mathcal{A}$ the space of candidate configurations, where a configuration is an instantiation of all catalogue attributes.

Feature space. Each configuration is mapped to a *feature space*, in which the user utility will be computed, by a *feature mapping* function $\Psi : \mathcal{A} \rightarrow \{0,1\}^m$, where m is the size of the feature space. Features are defined in terms of *constraints* over configuration attributes, each constraint being a logic formula, by taking the indicator function over the constraint (see Table 1). *Atomic* constraints are built by simply evaluating Boolean attributes (i.e., $\varphi_i = A_i$) and thresholding ordinal or real attributes (i.e. $\varphi_{i_1} = (A_i < \theta_{i_1})$, $\varphi_{i_2} = (A_i < \theta_{i_2})$, etc), where the number of thresholds determines the granularity of the discretization and is decided *a-priori*, for instance by computing the empirical quantiles of the variables of interest, although alternatives can be conceived[2]. We assume that the set of atomic constraints is given as an input to the algorithm. Arbitrary *composite* constraints can be built by combining atomic ones with logical connectives (e.g., `distance to kindergarten` $< \theta \wedge$ `distance to parents` $< \theta$, so that a car is not needed). The space of constructible constraints is clearly exponential in the

[2]Note that this approximate solution becomes inefficient for a large number of thresholds. We are planning to overcome this limitation by generalizing the approach to deal with features which are continuous functions of the variables in the constraints. While this is out of reach of "standard" weighted Max-SMT solvers, a recent technology called Optimization Modulo Theory [17] can be leveraged to achieve this goal, as will be discussed in the conclusion of the paper.

maximum number of atomic constraints allowed in a composite constraint. In this work we consider only *conjunctions* of up to d atomic constraints, d being a parameter of the algorithm:

$$\varphi_I = \bigwedge_{i \in I} \varphi_i$$

where $I \subseteq [1, |A|] : |I| \leq d$ identifies any subset of at most d attributes ($|A|$ is the number of catalogue attributes). The resulting feature space has dimension $m = \sum_{j=1}^{d} |A|^j$. Note that most of these constraints, as many of the catalogue attributes, will likely be irrelevant for a certain DM, while others will have different degrees of relevance. We thus treat all of them as candidate *soft constraints*, whose relevance needs to be discovered in the elicitation process and which can be traded-off in the search for feasible configurations. We also assume a set of *hard constraints*, known in advance, which define the space of feasible configurations (e.g., customer budget, company constraints on feasible configurations).

Utility function. The DM utility function is assumed to be a weighted combination of features associated to the soft constraints [28]:

$$f(\boldsymbol{A}) = \boldsymbol{w}^T \boldsymbol{\psi}(\boldsymbol{A}) \tag{1}$$

where each weight indicates how much the DM likes (or dislikes, if negative) the corresponding feature, a zero weight indicating irrelevance. Due to their bounded rationality and limited information-processing capabilities, humans can handle only a limited number of features to make decisions. Thus only *very few* of the candidate soft constraints will be considered by the DM, resulting in an extremely sparse weight vector $\boldsymbol{w}$. This *sparsity assumption* will be accounted for in the learning stage.

Learning phase. Learning amounts to finding a weight vector $\boldsymbol{w}$ that makes the utility function in Eq. 1 match the unknown DM preferences. Asking quantitative feedback, such as real-valued utility scores, is typically not affordable for a human DM [5]. We thus cast the problem as a form of *learning to rank*. Training examples are obtained by asking the DM to rank pairs of configurations according to her own preferences. We adapt SVM for ranking [29], which learns from pairwise rankings by enforcing a (soft) large margin between the score of the two candidates. Note however that the feature

vector contains all possible constraints (up to a certain complexity) for all catalogue attributes, most of which are likely to be irrelevant for a specific DM. We favour discarding irrelevant constraints by encouraging sparsity on $\boldsymbol{w}$. Therefore, most of the weight mass will be associated to the truly relevant features. Feature selection is known to improve generalization in data sets with redundant and irrelevant features [30]. To this end, we replace the standard 2-norm regularizer of SVM with a sparsifying 1-norm regularizer [31]; this choice will be validated empirically in Section 7. The resulting learning problem is:

$$\begin{aligned} \min_{\boldsymbol{w}\in\mathbb{R}^{\mathrm{m}},\boldsymbol{\xi}\geq 0} \quad & ||\boldsymbol{w}||_1 + C\sum_{(i,j)\in\mathcal{D}} \xi_{ij}^2 \\ \text{s.t.} \quad & \boldsymbol{w}^T(\boldsymbol{\psi}(\boldsymbol{A}^i) - \boldsymbol{\psi}(\boldsymbol{A}^j)) \geq 1 - \xi_{ij} \qquad \forall\ (i,j)\in\mathcal{D} \end{aligned} \tag{2}$$

where the dataset $\mathcal{D}$ consists of pairs (i, j) for which the DM prefers configuration $\boldsymbol{A}^i$ over configuration $\boldsymbol{A}^j$ (written as $\boldsymbol{A}^i \succ \boldsymbol{A}^j$). The constraints enforce pairwise rankings to match the DM preferences. A quadratic penalty ξ_{ij}^2 is incurred when the utility of a more preferred configuration is not sufficiently larger than that of the less preferred one. The soft ranking constraints allow to accommodate occasional inconsistencies in the DM feedback. The objective to be minimized is the combination of two components: the sum of penalties for not satisfying ranking feedback from the DM, and the 1-norm of the weights which favours solutions with few non-zero weights. The regularization parameter C trades-off these two components and is optimized during the learning process, as discussed at the end of this section when describing the overall algorithm. Note that weights can also take negative values, where a negative weight can be interpreted as a DM disliking the corresponding constraint.

Optimization phase. The goal of the algorithm is to recommend the best possible configuration given the true DM utility function f. However the latter is unknown. The preference elicitation phase allows to gather information on DM preferences and refine the current approximation $\hat{f}$ accordingly. CLEO asks the DM for comparisons between pairs of configurations. Given the form of the utility function (Eq. 1) and of the underlying features, optimizing $\hat{f}(\boldsymbol{A})$ boils down to solving a weighted MAX-SMT problem (see also the example in Section 3). To generate two configurations to be compared by the DM, we design two optimization tasks considering the following

principles:

1. the generation of *high-quality* configurations, consistent with the learned DM preferences;
2. the generation of *diversified* configurations, i.e., alternative possibly suboptimal configurations with respect to the learned utility $\hat{f}$;
3. the search for catalog attributes relevant to the DM not recovered by the current approximation $\hat{f}$, i.e., attributes not appearing in any of the soft constraints in $\hat{f}$.

The rationale for the first principle is to focus on the relevant areas of the utility surface, those of interest to 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 DM [5]. In addition, the goal of CLEO is the identification of the configuration preferred by the user (*learning to optimize*) rather than an accurate global approximation of the DM utility function (*learning per se*). This requires to model the relevant areas of the optimization fitness surface rather than reconstructing it entirely. The second principle advocates the introduction of some diversification in the search, by exploring the neighbourhood of the best configuration for the currently learned preference model $\hat{f}$. Finally, as the learned formulation of $\hat{f}$ may miss some of the user decisional attributes, their search is explicitly promoted by the third principle. The need for a set of good and diverse configurations to be evaluated by the user is suggested also in [32].

Our optimization phase works as follows. First, $\hat{f}$ is maximized (first principle), generating the first candidate configuration $\boldsymbol{A}^*$. Then, a *hard* constraint is added to the Max-SMT problem as the disjunction of all soft constraints not satisfied by $\boldsymbol{A}^*$, and maximization is run again. This instantiates the second principle, by synthesizing a new configuration $\boldsymbol{A}^{**}$ which differs from $\boldsymbol{A}^*$ by at least one soft-constraint. If $\boldsymbol{A}^*$ satisfies all soft constraints in $\hat{f}$, the additional *hard* constraint generated is: $(\neg A_1^* \vee \neg A_2^* \ldots \vee \neg A_n^*)$ which excludes $\boldsymbol{A}^*$ from the set of feasible configurations. Finally, each unassigned attribute, i.e., catalog attribute not appearing in any hard constraint or soft constraint with non-zero weight, in both $\boldsymbol{A}^*$ and $\boldsymbol{A}^{**}$ is given a random value in its domain, thus incorporating the third principle. Indeed, if these catalog attributes are truly irrelevant for the DM, setting them at random should not affect the evaluation of the candidate configurations. On the other hand, if some of them are needed to explain the DM preferences, driving their elici-

algorithm `CLEO` *($\mathcal{A}$, $\mathcal{C}_S$, $\mathcal{C}_H$, d)*:

input : Space of configurations $\mathcal{A}$, set of atomic soft constraints $\mathcal{C}_S$, set of hard constraints $\mathcal{C}_H$, maximum size of non-atomic constraints d

output : Most preferred configuration $\boldsymbol{A}^*$

`// Initialization`

Build feature map Ψ from $\mathcal{A}$, $\mathcal{C}_S$ and d

Select two configurations at random

$\mathcal{D} \leftarrow$ ranking of configurations by DM

$C \leftarrow 1$

`// Refinement`

while *termination criterion not satisfied* **do**

$\hat{f} \leftarrow \text{argmin}_{\boldsymbol{w},\boldsymbol{\xi}\geq 0} ||\boldsymbol{w}||_1 + C \sum_{(i,j)\in\mathcal{D}} \xi_{ij}^2$ `// learning phase`

s.t. $\boldsymbol{w}^T(\boldsymbol{\psi}(\boldsymbol{A}^i) - \boldsymbol{\psi}(\boldsymbol{A}^j)) \geq 1 - \xi_{ij}$

$\forall\ (i,j) \in \mathcal{D}$

$\boldsymbol{A}^* = \text{argmax}_{\boldsymbol{A}\in\mathcal{A}} \hat{f}(\boldsymbol{A})$ `// optimization phase`

s.t. $\boldsymbol{A} \models \mathcal{C}_H$

if *DM satisfied* **then**

return $\boldsymbol{A}^*$

end

else

`// preference elicitation`

Generate $\boldsymbol{A}^{**}$ by diversification strategy

$\mathcal{D} \leftarrow \mathcal{D} \cup$ ranking of pair $(\boldsymbol{A}^*, \boldsymbol{A}^{**})$ by DM

end

$C \leftarrow \textsc{UpdateC}(\Psi,\mathcal{D})$ `// Update C by inner CV`

end

`// final recommendation`

return $\boldsymbol{A}^*$

Algorithm 1: Pseudocode for CLEO.

tation will reveal the deficiencies of the current approximation $\hat{f}$ and recover previously discarded relevant decisional items.

Overall algorithm. The pseudocode of the full CLEO algorithm is shown in Algorithm 1. It takes as input the space of candidate configurations $\mathcal{A}$, the set of atomic soft constraints $\mathcal{C}_S$, the set of hard constraints $\mathcal{C}_H$, and the maximum size of composite constraints d, and returns the configuration

which is most preferred by the DM.

During initialization, the DM is asked to compare two configurations chosen from $\mathcal{A}$. The first one is obtained by solving an SMT problem instance formed by the conjunction of the hard constraints defining $\mathcal{A}$. Indeed, we have not learned any formulation of the user utility function yet. The second one is obtained by optimizing a set of diversification constraints that both guarantee the generation of a configuration different from the first one and favour diversity between the two configurations. Then the refinement loop begins. The refinement of the utility function amounts to solving the "learning to rank" problem in Eq. (2), where $\mathcal{D}$ is the dataset of all pairwise preferences collected so far. With a slight abuse of notation, we write $\hat{f} \leftarrow \text{argmin}$ to indicate that $\hat{f}$ is the function whose weights $\boldsymbol{w}$ are the result of the minimization. The regularization parameter C is set to one during the first three iterations, and fine-tuned by an internal three-fold cross validation procedure on the training set in the following ones. This tuning step, performed by the UpdateC procedure (line 19), works by splitting $\mathcal{D}$ in three equally sized sets, and in turn learning on two sets by solving Eq. (2) and testing on the left-out set, for different values of C. The ranking loss (i.e., the number of incorrect rankings obtained using the learned utility) averaged on the three resulting test sets is used to select the best C.

After the learning step, a new recommendation is made by selecting the configuration $\boldsymbol{A}^*$ maximizing the learned utility function $\hat{f}$, subject to the hard constraints. If the DM is not satisfied with the suggested configuration, an additional candidate configuration $\boldsymbol{A}^{**}$ is generated, favouring diversity between $\boldsymbol{A}^*$ and $\boldsymbol{A}^{**}$ based on the diversification strategy defined above (see the Optimization phase paragraph). The dataset $\mathcal{D}$ is then updated by including the comparison between $\boldsymbol{A}^*$ and $\boldsymbol{A}^{**}$ performed by the DM.

Being an interactive process involving a human DM, the most natural termination condition is the DM satisfaction with the current recommendation. Additional conditions could be conceived, for instance, by estimating the improvement one could expect by further refining the utility function. We will discuss this and other potential extensions in the conclusion.

5. Features of CLEO

The CLEO algorithm has *no free parameters* to be manually tuned (except for the θ thresholds of the soft constraints, that should be chosen by the domain expert when compiling the catalogue of attributes). The number

of iterations does not need to be fixed upfront, as the termination criterion is represented by the satisfaction of the DM with $\boldsymbol{A}^*$. The regularization parameter C in Eq. (2) is set to one in the first three iterations, and fine-tuned by internal cross-validation on the training set in the following ones. At each iteration, a single pairwise comparison is asked to the DM. The configurations to be compared at the first iteration are generated by sampling at random the feasible search space. The evaluation of diverse examples stimulates the preference expression, especially when the user is still uncertain about her final preference [32]. In particular, the diversity of the proposed configurations helps the user reveal 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 preferences for intermediate stops until a configuration suggests an airplane change in a place she dislikes [32].

The human cognitive capabilities bound the number of catalog attributes and the size d of the soft constraints. The limited size of the Max-SMT problems generated by CLEO enables the systematic investigation of the search space by means of a complete solver, which ensures the identification of a global maximum $\boldsymbol{A}^*$ of the learned utility model $\hat{f}$ (completeness property). However, CLEO cannot guarantee the quality of the model $\hat{f}$ approximating the true DM utilities, and therefore the optimality of $\boldsymbol{A}^*$ (or bounds on its quality) w.r.t. the *true* DM utilities cannot be proved. 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, CLEO does not need to learn the exact form of the DM utility function. The goal of our approach is to elicit as few preference information from the DM as possible to identify her favourite configuration (*learning to optimize*). For example, consider the toy DM utility function represented by the negation of a single ternary term: $\neg(\varphi_1 \wedge \varphi_2 \wedge \varphi_3)$. The approximation of the DM utility function consisting of the formula $\neg\varphi_1$ is sufficient to find one of the favourite DM configurations. In general, only the shape of the utility function locally guiding the search to the correct direction is actually needed. The experimental results in Sec. 7 demonstrate that CLEO rapidly improves the quality of the candidate configurations when the number of refinement iterations increases (anytime property).

Finally, CLEO satisfies the main requirements for practical applicability of preference elicitation. In detail:

1. *real-time* interaction with the DM. Due to the limited number of pairwise preference constraints elicited from the DM, the learning phase at each refinement step (problem (2)) is accomplished in a time which is negligible when compared with the user response time[3]. Analogous observation holds for the computational effort required by the optimization phase, provided that the feasible space $\mathcal{A}$ is defined by a reasonable number of hard constraints (this is usually the case in preference elicitation problems[4]). Proposing a query consists of generating two candidates to be compared. Each candidate is obtained by a run of the complete Max-SMT solver. The bounded value of n and the efficient performance of modern SMT solvers, that can efficiently manage problems with thousands of variables and millions of constraints, enable the completion of the optimization phase in a negligible amount of time;
2. *multi-attribute* models. Candidate configurations are described by multiple decisional attributes. Since these attributes usually vary with different decision makers, CLEO assumes a set of catalog attributes, from which the decisional items of a specific DM are automatically selected. Unlike the state-of-the-art methods (see Sec. 6) for preference elicitation, CLEO can handle both discrete and continuous-valued attributes simultaneously, thanks to the Max-SMT formalism which can efficiently tackle hybrid domains;
3. *user cognitive load*. CLEO asks the user just for pairwise comparisons of candidate configurations. Pairwise comparisons require low cognitive load for decision makers [33]. Most users are typically more confident in comparing configurations, providing qualitative judgments like "I prefer configuration $\boldsymbol{A}^*$ to configuration $\boldsymbol{A}^{**}$ ", rather than in specifying how much they prefer $\boldsymbol{A}^*$ over $\boldsymbol{A}^{**}$.
4. *robustness* to inconsistent human feedback. Assuming that a user always provides accurate and consistent preference information is not realistic. Different factors may generate inconsistent feedback from the DM, including occasional inattention, embarrassment when comparing very similar configurations or configurations which are very differ-

[3]Note however that we cannot provide formal guarantees about the optimality of the learned preference model w.r.t. the actual user utility.

[4]The adoption of CLEO for tackling large combinatorial problems arising in industrial applications of combinatorial optimization is discussed in the conclusion of the paper.

ent from her favourite one, DM fatigue increasing with the number of queries answered. The adoption of regularized machine learning strategies in CLEO enables a robust approach that can handle inaccurate (pairwise) comparisons of configurations from the DM.

5. *scalability.* At each preference elicitation stage, just one candidate query is considered by CLEO, independently of the cardinality of the configuration space. The explicit enumeration of candidate soft constraints (see the mapping function Φ in Eq. (2)) is tractable only for a rather limited number of catalog attributes and size d of their combination. However, this will typically be the case when interacting with a human DM. Research in psychology has shown that humans cannot handle simultaneously more than few (7 ± 2) factors [18]. Various experiments agree on our limited capacity for processing information. Baddeley and Hitch [34] present a series of experiments on the role of memory in reasoning, language comprehension, and learning. Cowan [35] collects a wide variety of data on capacity limits. In the conclusion of the paper we discuss about alternative approaches which could be pursued when the size of the constraint space makes exhaustive enumeration infeasible.

6. Related work

The problem of automatically learning utility functions and eliciting preferences is widely studied within the Artificial Intelligence community [36, 3, 37]. In this section we first compare CLEO to state-of-the-art preference elicitation methods, roughly classifying them based on how they implement uncertainty over the utility function. We then briefly discuss the relationship between CLEO and product configuration tools. Finally, we overview existing query selection strategies used in active learning, which are not designed for interaction with human DMs, and contrast them to the strategy implemented by CLEO.

6.1. Strict uncertainty

A popular approach to model the uncertain knowledge about the DM preferences consists of assuming a set of hypotheses, with no belief on their strength. The set of hypotheses contains the feasible utility functions and reflects the partial knowledge about the DM preferences. The uncertainty 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* [36].

The *minimax regret* criterion [38] from statistical decision theory provides a way to make decisions under uncertainty. Given a certain decision $\boldsymbol{A}$, the maximum regret is the difference in utility between the DM's most preferred configuration $\boldsymbol{A}^*$ and $\boldsymbol{A}$ assuming the worst-case scenario, where the DM utility is the one in the feasible set for which this difference is maximal. By adopting the minimax regret criterion, the decision that minimizes this regret is taken. This criterion identifies a robust decision w.r.t. the worst possible case. The recent work in [6, 7, 8] introduces an approach to preference elicitation based on the minimax regret criterion. Queries to be asked to the DM are selected so as to reduce the minimax regret by restricting the feasible hypothesis set. An advantage of minimax regret approaches with respect to our formulation is that they can provide theoretical guarantees in terms of bounds on the configuration quality and convergence to provably-optimal results. On the other hand, these approaches assume perfect feedback from the DM and cannot handle the occasional inconsistencies which are typical of interactions with human DM. Therefore, they are not suitable for the realistic preference elicitation tasks considered in this work.

6.2. Bayesian uncertainty

An alternative uncertainty model consists of defining a *probability distribution* (or belief) over the candidate utility functions [9, 5, 10, 11], which is incrementally refined as new feedback is obtained. The Bayesian framework offers a flexible approach, handling the uncertainty in both utility and DM feedback. The queries are chosen as to increase the posterior probability of the true utility. A typical query selection strategy consists of selecting the query with the maximal *value of information* (VOI). Exact computation of VOI, as well as exact computation of the posterior distribution, however, are extremely expensive. The state-of-the-art approaches [9, 5] resort to approximate solutions.

The closest approach to CLEO is the Bayesian method introduced by Guo and Sanner in [5] (referred to as GSM). In GSM, utility functions are represented by a weight vector $\mathbf{w}$ (distributed according to a multivariate distribution) which specifies the utility of *each* possible value for *each* attribute. Unlike CLEO, this modelling choice assumes *preferential independence* among the set of attributes.

GSM offers several query selection strategies [5]. The principled *informed VOI* strategy requires to compute the VOI of all possible pairwise comparisons, and therefore scales quadratically with the number of configurations. Therefore, it quickly becomes unaffordable as the number of options increases. In [5], informed VOI was empirically shown not to scale to datasets with more than 20 attributes. The computational load can be decreased by restricting the set of candidate pairwise comparisons, e.g., by fixing one element of each candidate pair to the configuration $\mathbf{x}^*$ with greatest expected utility (*restricted informed VOI* strategy). Fixing one configuration in the query pair to the current optimal configuration $\mathbf{x}^*$ is the approach adopted also by CLEO. The authors of GSM also suggest an alternative query strategy which does not rely on the VOI criterion, namely the comparison between $\mathbf{x}^*$ and the configuration $\mathbf{x}^{EL}$ maximizing the expected loss of recommending $\mathbf{x}^*$ instead of $\mathbf{x}^{EL}$ (*simplified VOI* strategy).

Unlike the other techniques discussed in this section, CLEO and GSM both satisfy [5] all the main principles needed for practical applicability of preference elicitation (see Sec. 1). Nonetheless, GSM requires the exhaustive enumeration of candidate configurations, which is infeasible in hybrid search domains including continuous attributes. In our experiments (Sec. 7), an empirical comparison of CLEO w.r.t. GSM is thus performed over a simplified experimental setting involving discrete decisional attributes only.

6.3. Constraint-based preference elicitation

The work in [12] articulates the user preferences in terms of *soft* constraints and introduces constraint optimization problems where the DM preferences are not completely known beforehand. The decision variables and the soft constraint structure are assumed to be known in advance. On the contrary, CLEO can work even when the structure of the constraints is completely unobserved: the initial problem knowledge is limited to a set of catalog attributes, while the weighted constraints are learned on-the-fly from the DM feedback.

Another issue with the algorithm in [12] is that it requires the DM to disclose the score values of her utility function. CLEO instead relies on pairwise ranking queries, a much more cognitively affordable task. Finally, the approach in [12] assumes consistent and accurate quantitative feedback from the DM, and cannot be applied in noisy scenarios.

Conversational, or example critiquing, recommenders are another class of interactive constraint-based methods. In conversational interaction, the

user is free to provide the system with critiques to the recommended options. Newly acquired critiques are used to constraint the search to the high utility regions of the product catalogue. Few critiquing recommenders model the user preferences directly. Those that do, like [39] and [40], assume noiseless user responses. The method proposed in [41] learns a linear utility through a heuristic multiplicative update, and could in principle deal with imperfect DMs, but handles discrete configurations only.

6.4. Product Configuration

CLEO is also related to the problem of product configuration [42, 43]. In knowledge-based configuration systems, both the product catalogue and the taxonomy of product components are encoded in some constraint-based modelling language. As in CLEO, inference exploits constraint satisfaction techniques. Most configurators, however, do not tackle the issue of interactive preference learning: the full specification of the user requirements is assumed to be available beforehand. This is of course unachievable in recommendation domains where the system interacts with non-experts (e.g., online sales).

An exception is given by Conditional Preference Networks (CP-nets) [44, 45], a class of graphical models that have been proposed for reasoning with preferences in product configuration and other contexts. CP-nets leverage conditional preferential independence assumptions to compactly represent rich preference relations. Whereas some preference learning approaches for CP-nets have been devised (see [46], Sec. 3.5), CP-nets cannot represent continuous numerical attributes, and, unlike CLEO, cannot be applied to hybrid problem domains.

6.5. Query Diversity

Recommendation and query diversity is a key issue in recommendation [47, 48, 49] and other settings, such as active learning and information retrieval. The peculiar nature of the recommendation task however poses challenges for applying diversification strategies proposed in other settings, as explained below.

Active learning is a hot research area and a broad range of different approaches has been proposed (see [50, 51] 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 suggest candidate configurations that are both of good quality and maximally informative. Such a strategy also addresses the issue of user engagement. A human DM may decide to leave the elicitation session when presented with low quality configurations, an issue neglected by active learning. In CLEO, the quality of the query set is encouraged by maximizing the (estimated) utility of the two query configurations.

Typical active learning query selection strategies are not designed to deal with humans. For instance, in linear classifiers, uncertainty sampling boils down to selecting items close to the margin (although several alternatives are available [52]). In a pairwise ranking setting, this strategy equates to choosing pairs of configurations with similar utilities. However, distinguishability is crucial for eliciting meaningul preferences from human DMs: the more similar the two query configurations, the more unreliable the user feedback [49]. The inverse correlation between utility difference and noise is also reflected by many widely used response models, like the Thurstone-Mosteller [53], Bradley-Terry [54], and Plackett-Luce [55, 56] models. CLEO favors distinguishability by perturbing the feature representation of query configurations.

Other related research areas are single- and multi-objective interactive optimization [57] and information retrieval [58]. The need to trade-off multiple requirements in this active learning setting is addressed in [59] 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 CLEO indeed depends on the trade-off between the identification of candidate configurations satisfying the DM (i.e., configurations optimizing the current learned preference model) and the generation of *informative* training examples for the following refinement of the learned model.

7. Experimental results

The following empirical evaluation shows that CLEO can handle realistic preference elicitation tasks defined over hybrid domains and with noisy human feedback. First, CLEO is tested over two realistic preference elicitation tasks with the above features. In these experiments, we evaluate the performance of CLEO for increasingly complex problems, and the effectiveness of its sparsifying 1-norm against the classical 2-norm. In a second step, a set of simplified synthetic problems with discrete decisional variables is introduced, to compare CLEO with the existing preference elicitation algorithms,

which cannot handle inference with continuous attributes. In particular, we consider Boolean decisional attributes only and generate a set of synthetic Maximum-Satisfiability (Max-SAT) benchmarks. In this simplified setting, benchmarking is the method by Guo and Sanner [5].

For the experiments, the mapping function $\boldsymbol{\psi}$ in CLEO projects configurations into the space of all possible conjunctions of up to three atomic constraints (i.e., $d = 3$). The Max-SMT tool used is the Yices solver[5] with the $\mathcal{LIA}$ and $\mathcal{LRA}$ theories. The SVM implementation used to solve the learning-to-rank problem (2) is the LIBLINEAR package [60].

We model user responses with the classic Thurstone-Mosteller (aka Probit) model [53], widely used in economics and psychology to describe the individual choice behaviour of humans [61, 62, 63]. The user ranks configurations based on a latent utility function f. In particular, configuration $\boldsymbol{A}^i$ is preferred to configuration $\boldsymbol{A}^j$ if and only if $f(\boldsymbol{A}^i) > f(\boldsymbol{A}^j)$. However, each evaluation $f_i = f(\boldsymbol{A}^i)$ is corrupted by additive independent and identically distributed (IID) Gaussian noise $\varepsilon_i \sim \mathcal{N}(0, \sigma^2_{noise})$, resulting in a noisy utility value $y_i = f_i + \varepsilon_i$. Under these assumptions, the probability that the user prefers $\boldsymbol{A}^i$ to $\boldsymbol{A}^j$ is:

$$\begin{aligned} \mathrm{P}(\boldsymbol{A}^i \succ \boldsymbol{A}^j | f_i, f_j) = \mathrm{P}(y_i > y_j | f_i, f_j) = \\ \mathrm{P}(f_i + \varepsilon_i > f_j + \varepsilon_j) = \mathrm{P}(\varepsilon_i - \varepsilon_j > f_j - f_i) \end{aligned} \tag{3}$$

The quantity $\delta = \varepsilon_i - \varepsilon_j$ is the difference of two IID Gaussian variables with zero mean and variance σ^2_{noise}, and therefore follows the Gaussian distribution $\mathcal{N}(0, 2\sigma^2_{noise})$. By computing the standardized variable $z = \delta/(\sqrt{2}\sigma_{noise})$, Eq. (3) can be rewritten as:

$$\mathrm{P}(\varepsilon_i - \varepsilon_j > f_j - f_i) = 1 - \Phi\left(\frac{f_j - f_i}{\sqrt{2}\sigma_{noise}}\right)$$

where Φ is the cumulative distribution function of the standard normal distribution. In our experimental setting σ^2_{noise} is fixed to 10, to have noise values comparable with the latent utility values $f(\boldsymbol{A})$.

[5] Version 1.0, available at `http://yices.csl.sri.com/`.

7.1. Scheduling problem

Here the problem is that of learning DM preferences about the scheduling of a set of jobs. A set of five jobs must be scheduled over a given period of time. Each job has a fixed known duration, the atomic constraints define the overlap of two jobs or their non-concurrent execution. The user's unknown utility function is generated by selecting uniformly at random weighted conjunctions of atomic constraints. A solution to the problem is a schedule assigning a starting date to each job and maximizing the utility, where the utility of the schedule is the sum of the weights of the satisfied constraints in the user's utility function. The atomic soft constraints define temporal constraints 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 ith 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 as $s_j - s_i \geq d_i$. By considering the concurrent and non-concurrent execution of each pair of jobs from the set of five jobs, 40 *atomic* soft constraints are generated. A total of $m = 10700$ soft constraints is obtained by the conjunction of up to three ($d = 3$) atomic constraints.

CLEO and its 2-norm variant are tested using a benchmark of randomly generated utility functions with a given number of atomic soft constraints and soft constraints. We generate functions with $(5, 3)$, $(10, 6)$, and $(15, 9)$ soft atomic and atomic constraints, respectively.

The DM utility functions are generated as follows. First, the given number of atomic soft constraints is selected from the 40 candidate atomic constraints. Then, the desired number of soft constraints is constructed from the selected atomic ones, by including at least two soft constraints with a size of three. The weights of soft constraints are distributed uniformly at random in the range $[1, 100]$. Let us stress once more that utility functions with more that few factors or factors with many terms are unrealistic when considering human DMs [18].

The results of the experiments are shown in Figure 1. The y-axis reports the percentage utility loss measured in terms of deviation from the utility of the DM preferred configuration, while the x-axis contains the number of pairwise comparisons asked so far. The curves report the median values observed over 400 runs, while the shaded area depicts the interquartile range (IQR) measuring the dispersion around the median.

As expected, the learning problem becomes more challenging for an increasing number of soft constraints. However, results are promising, as a

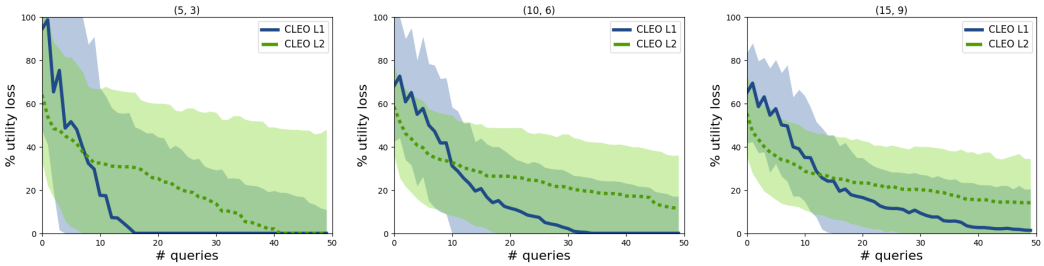

Figure 1: Performance of CLEO (solid blue line) and the 2-norm CLEO variant (dotted green line) on the scheduling problem. The plot shows how the percentage utility loss (y-axis) changes with increasing number of pairwise comparisons (x-axis). Best viewed in colour.

substantial improvement in the quality of the recommended configuration is achieved by CLEO when additional queries are asked to the DM (anytime property). Furthermore, CLEO identifies the DM preferred configuration in all cases. In detail, with the realistic cases of three and five soft constraints, less than 35 pairwise comparisons are asked to the DM to identify her preferred configuration. With 9 soft constraints, 64 pairwise comparisons are required on average to recommend the DM preferred configuration. However, with 40 queries, a percentage utility loss within 5.5% is obtained. The shaded area shows that CLEO identifies the DM preferred configuration quite consistently when increasing the number of queries (the IQR is within 25% after 35 queries even in the case of nine soft constraints).

The plots also show that CLEO (solid blue line) performs much better than the 2-norm variant (dotted green line), as expected, thanks to the automatic feature selection property of the 1-norm. For instance, in the simpler (and denser) $(5, 3)$ case, the 2-norm variant requires more than double the number of queries to satisfaction than the 1-norm. These results validate the performance advantage of the 1-norm versus non-sparsifying norms.

7.2. Housing problem

In this second experiment we consider a customer planning to build her own house by interacting with a real estate agency and a construction company (henceforth the *housing* problem). There are different locations available where the customer may potentially build her house. The locations are characterized by different housing values, prices, constraints about the design of the building (e.g., usually in the city center you cannot build a

family house with a huge garden and pool), etc. The customer may formulate her judgments by considering a description of the characteristics of the building to be constructed (e.g., house type, presence of a garage or a parking space) and of the candidate locations (e.g., distance from downtown, proximity to commercial facilities or green areas). Many of these parameters may be uninformative, as they do not represent any decisional criterion for the customer. Furthermore, hard constraints defining the feasible options may be specified in advance, e.g., cost bounds stated by the user or building design requirements asserted by the company.

In our experiments, the formulation of the housing problem is as follows. The set of catalog attributes is listed in Table 3. A set of ten hard constraints

Table 3: Catalog attributes for the housing problem.

num	attribute	type
1	house type	integer
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	real
8	crime rate	integer
9	location-based taxes and fees	integer
10	public transit service quality index	integer
11	distance from high schools	real
12	distance from nearest free parking	real
13	distance from working place	real
14	distance from parents house	real
15	price	real

(Table 4) defining feasible options is 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 op-

tions are defined as a function of the other attributes. 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.

Table 4: 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 areas 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 $\geq \rho_{13}$

Forty atomic constraints are generated to define the candidate decisional features of the user. They include the five Boolean attributes in Table 3 and 35 predicates over the integer and real attributes. The predicates discretize the integer and real attributes into different intervals. For example, by scaling the distance from downtown A_i in the range [0,10], four predicates $A_i = 0$ (the location lies in the city center), $1 \leq A_i < 3$, $3 \leq A_i < 6$, and $A_i > 6$ are generated. The actual user decisional features are an unknown subset of the 40 predicates. The unknown utility function is the weighted sum of soft constraints, each soft constraint being the conjunction of two to three atomic constraints. The maximum number of atomic constraints in a soft constraint is assumed to be known. The weights of soft constraints are integer values

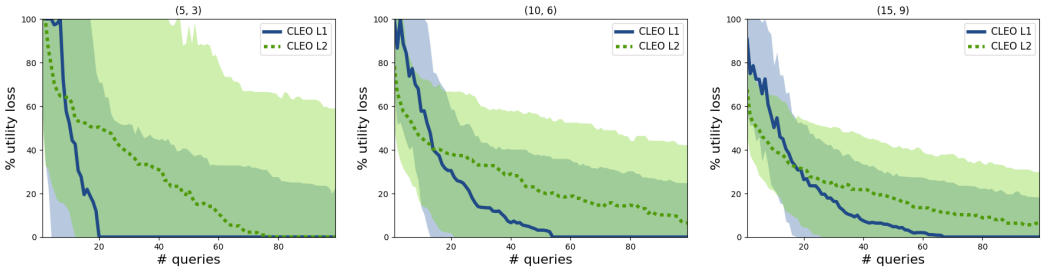

Figure 2: Performance of CLEO (solid blue line) and the 2-norm CLEO variant (dotted green line) on the housing problem. Best viewed in colour.

selected uniformly at random in the range $[1, 100]$. As in the scheduling problem, the user utility function is parametrized by the couple (*number of atomic soft constraints, number of soft constraints*).

Fig. 2 reports the results over a benchmark of 400 randomly generated utility functions for each of the following instantiations of the couple (*number of atomic soft constraints, number of soft constraints*): $\{(5, 3), (10, 6), (15, 9)\}$. The promising results observed for the scheduling problem are confirmed, including the superiority of the 1-norm over the 2-norm in this context, even though the housing problem is much harder, due to complex non-linear interactions among the decisional attributes. Once again, the 2-norm variant fails to identify the right soft constraints: whereas in the $(15, 9)$ problem the 1-norm penalty drives CLEO to uncover around 25 relevant soft constraints, the 2-norm wrongly allocates weight larger than 10^{-4} to more than 3000. This highlights that the choice of norm has a huge impact on the set of candidate utilities, as expected, which naturally influences the sample complexity of our learning problem.

When increasing the number of queries asked, the quality of the configuration rapidly improves and CLEO identifies the DM preferred configuration in all the cases. On average, 22 and 69 queries are needed by CLEO to converge to the DM preferred configuration in the case of three and nine soft constraints, respectively. Let us note again that utility functions involving nine soft constraints are quite unrealistic and are considered here just for testing the scalability of CLEO.

The dispersion of the performance values keeps decreasing when increasing the number of queries asked, showing that CLEO recommends better quality configuration more consistently. However, in the case of three soft constraints, the IQR observed when CLEO converges is equal to 70.8%. With

40 queries, the dispersion decreases down to 45.4%. These values are rather large. A deeper investigation of CLEO results revealed that the observed data dispersion is heavily affected by some runs where the configuration quality does not improve when asking additional feedback to the DM. In these runs CLEO cannot generate queries informative enough to recover from suboptimal initial choices. Smarter queries strategies could be studied to tackle these cases, as discussed in Sec. 8.

7.3. Experimental comparison with the state-of-the-art

Since existing methods cannot handle the preference elicitation tasks over hybrid domains defined in the previous section, for a comparison with the state-of-the-art we focus on Boolean attributes only. With this choice, the atomic constraints are just the Boolean attributes, and more complex soft constraints expressing the DM preferences are Boolean terms in plain propositional logic. That is, each soft constraint is the conjunction of (up to three) Boolean attributes and optimization of the utility function can be cast as a *weighted Maximum Satisfiability* (Max-SAT) problem. The benchmarking algorithm is the GSM method [5] described in Sec. 6.2. The parameter setting for GSM is the same one used by the authors in their experimental evaluation [5]. In particular, the mean and standard deviation of each dimension of the multidimensional Gaussian prior with diagonal covariance are set to 25 and 25/3, respectively.

A benchmark of random utility functions is generated for (*number of Boolean attributes, number of terms*) equal to $\{(5,3),(10,6),(15,9)\}$. Each utility function has two constraints with maximum size (three). Constraint weights are integers selected uniformly at random in the interval $[-100,0) \cup (0,100]$.

All the query selection strategies suggested in [5] for the GSM method have been tested in our experimental setting. For each of the three test cases $\{(5,3),(10,6),(15,9)\}$, we report here the results of the query strategy with best performance. However, with more than five attributes, the most sophisticated Bayesian query strategies proposed in [5] are too slow, as pointed out also by the authors and empirically verified in our preliminary experiments. They have thus been included in the $(5,3)$ case only. Based on our results, the best query strategy are the "restricted informed value of information (VOI)" for the test case $(5,3)$ and the "simplified VOI" for both remaining test cases.

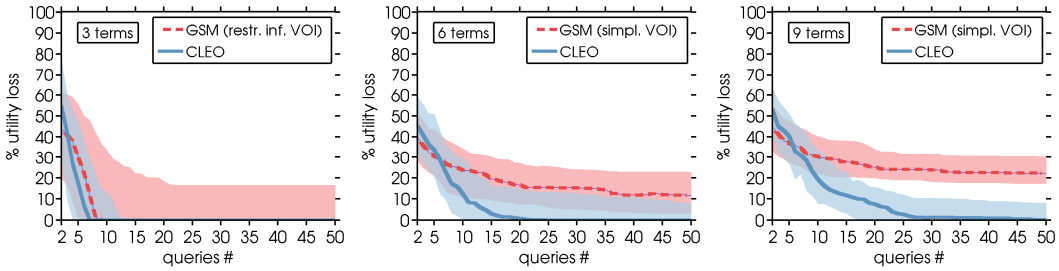

Figure 3: Performance of CLEO (solid blue line) and GSM (dashed red line) on the Boolean problem. Best viewed in colour.

Fig. 3 reports the percentage utility loss of the recommended configuration w.r.t. the DM preferred configuration for an increasing number of pairwise comparisons asked so far. The curves report the median values observed over 200 runs for CLEO and GSM. The shaded areas depict the IQR measuring the dispersion around the median.

The search space of the simplest problem with five Boolean attributes contains just 32 candidate configurations, thus any strategy asking more than few questions is not competitive with naïve exhaustive search. On average, seven and nine queries are asked to the DM by CLEO and GSM for discovering her preferred configuration. However, with 12 (or less) queries, the CLEO and GSM performance are statistically equivalent under a Two-sided Wilcoxon signed-rank test with a Bonferroni-corrected significance level of 10^{-3}. With more than 12 queries, there is statistical evidence for better results by CLEO, due to the much more unstable behavior of the GSM method: after 14 queries CLEO consistently identifies the DM preferred configuration with a null IQR, while the IQR of the GSM results remains above 16.6%.

The more challenging test cases are represented by the problems with 10 and 15 Boolean attributes, where the search space size is 1024 and 32768, respectively, preventing the application of exhaustive search techniques. In both these cases, the performance of CLEO is much better than that of GSM.

In detail, with 10 Boolean attributes, CLEO on average asks 25 pairwise comparisons to the DM for identifying her favourite configuration, while the average percentage utility loss of the configuration recommended by GSM remains above 10% even if 50 queries are asked to the DM. With 16 queries, the CLEO curve is within 2%, against a value of around 19% observed for GSM. The performance difference between CLEO and GSM is significant at

10^{-3} level after eight queries, and the significant level goes to 10^{-11} after 15 queries.

An analogous situation is observed for the $(15, 9)$ test case. The configuration returned by CLEO has an average loss of less than 2% after 26 queries and less than 1% after 38 ones. On the other hand, after 50 queries, GSM recommends on average configurations with a loss still above 22.3%. The performance difference after the first seven queries is statistically significant with a 10^{-3} level, which goes to 10^{-10} after ten queries.

Ultimately, the main reason why GSM underperforms is that it is not designed to recover sparse preferences, and we do expect sparsifying Bayesian techniques to perform much better. This is a promising direction for future research, keeping in mind the complexity of scaling Bayesian approaches to realistic constructive problems.

8. Conclusion

We introduced CLEO, a preference elicitation algorithm that, unlike existing approaches, handles constructive preference elicitation problems when recommending a complex configurable entity. It proceeds by optimizing the configuration subject to feasibility constraints and user preferences to be learned during the process. A combinatorial formulation of the unknown DM utility function is adopted. CLEO consists of an incremental procedure, iteratively optimizing the learned approximation of the DM utility function to generate candidate configurations and refining the approximation based on the human feedback received. Simple pairwise comparison queries are asked.

CLEO assumes *very limited initial knowledge*. Because different DM usually have different decisional criteria, the algorithm just assumes a set of *catalog* attributes describing the candidate configurations. The DM preferences are expressed by soft constraints over the attributes values. However, only a small subset of catalog attributes (and, by consequence, of soft constraints defined on them) may be relevant for a specific DM, resulting in a sparse learning setting, both in the number of relevant attributes and soft constraints. The algorithm employs 1-norm regularization, which enforces sparsity of the learned function, to identify the relevant attributes and constraints.

The learned function is a set of weighted soft constraints involving both discrete and continuous-valued attributes. The configuration maximizing the

weights of the satisfied constraints is recommended to the DM. To identify this configuration, a Max-SMT solver is used. CLEO is a generic framework, enabling the adoption of well-assessed learning methods and Max-SMT solvers.

Experimental results on realistic preference elicitation tasks demonstrate the effectiveness of CLEO in focusing on the optimal configurations, its robustness, as well as its ability to recover from suboptimal initial choices. Our experiments involve preference elicitation tasks over hybrid domains, with noisy human feedback, (known) hard constraints limiting the set of feasible configurations and complex non-linear interactions among the decisional attributes (e.g., the cost attribute in the case of the housing problem). CLEO has also been compared with a state-of-the-art Bayesian preference elicitation approach in a simplified setting with purely discrete attributes. The experimental results show that CLEO outperforms the benchmarking algorithm, with the performance difference becoming more pronounced when the complexity of the preference elicitation task increases.

CLEO can be generalized in a number of directions. CLEO currently turns algebraic constraints into Boolean features by considering their truth value for the configuration being evaluated. Therefore we can formalize the optimization task as a weighted Max-SMT problem and address it with off-the-shelf solvers like Yices. However, one would often like to penalize configurations proportionally to the distance from the satisfaction boundary of algebraic constraints (e.g., if a distance from the kindergarten of more than 1 km makes it uneasy to always go by foot, a distance of 1.1 km is clearly less problematic than a distance of 10 km). To address this limitation we are investigating the use of a recent technology called Optimization Modulo Theory (OMT) [17]. OMT generalizes weighted Max-SMT by allowing to optimize an arbitrary cost function defined on the variables of an SMT theory, provided its dependence on the numeric variables is linear. The linearity requirement poses some challenges in dealing with composite constraints (e.g., we cannot define the cost of a conjunction of predicates as the product of costs for atomic predicates, as one would do in the purely Boolean case), but we are planning to try out different fuzzy logic t-norms which OMT can handle (e.g., Lukasiewicz, minimum) to determine efficient encodings. As a more general alternative we plan to leverage on the research on hybrid non-linear arithmetics [64, 65, 66], when the solvers will reach the desired level of maturity.

The learning stage of CLEO employs a ranking loss function based on

pairwise preference evaluation. More complex ranking losses have been proposed in the literature (see for instance [67]), especially to increase the importance of correctly ranking the highest scoring configurations, and could be combined with 1-norm regularization.

In the context of preference elicitation, Bayesian approaches are attractive as they quantify the uncertainty in the learned 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 learned preference model. The value of information is exploited to design efficient query strategies consisting of informative queries, see, e.g., the GSM [5] algorithm we use as benchmark in the experimental comparisons. Adapting these concepts to our setting, where the utility function is defined over hybrid domains and models complex non-linear interactions between attributes, is highly non-trivial, as our comparisons suggest (see Section 7.3). Non-Bayesian approaches to this problem have been proposed in some follow-up work of ours, cf. [68].

Another research direction is the extension of our approach to handle feedback from multiple DMs [69]. In particular, an interesting case study is the exploitation of preferences of previous DMs to minimize the elicitation effort for a new user [9, 11]. We also plan to extend our algorithm to tackle preference drift [70], i.e., the tendency of the DM to change her preferences during the interactive utility elicitation process. In our combinatorial utility settings, the DM preference drift can be modelled by weights of soft constraints evolving over time and by logic formulae 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, this paper focused on preference elicitation tasks involving small-scale problems typical of an interaction with a human DM. From a more general perspective, CLEO provides a framework for the joint learning and optimization of unknown combinatorial functions, involving both discrete and continuous decision variables. In principle, when combined with appropriate SMT solvers, CLEO could be applied to large combinatorial optimization problems (e.g., arising from industrial applications of combinatorial optimization [71]), whose formulation is only *partially* available. However, the cost of requiring an explicit representation of all possible combinations 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 [21] when learning quantitative scores. As our previous results seem to indicate [21], this can degrade the quality of returned configurations when the utility function is very sparse. Kernelized versions of zero-norm regularization [72] could be tried to enforce sparsity in the projected space if needed. However, let us note 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.

Follow-up approaches to constructive preference elicitation are rooted on the same intuition as CLEO, namely that constructive preference elicitation can be solved by iteratively synthesizing high-quality, diverse recommendations and using these to elicit the user's preferences. Like CLEO, all of these algorithms generate candidate and query recommendations using constraint or mathematical optimization, like MILP [73] and MiniZinc [74]. These technologies do not require, for instance, to discretize the continuous features and to define a multitude of thresholds. The various algorithms differ mostly in what kind of queries they ask to the user, for instance choice queries over pairs [75] or sets [73] of candidate configurations or rather improvement queries on individual configurations [74]. A recent overview of constructive preference elicitation [68] shows how these algorithms all stem from ideas set forth by CLEO.

Acknowledgments

This research was partially supported by grant PRIN 2009LNP494 (Statistical Relational Learning: Algorithms and Applications) from Italian Ministry of University and Research. ST was supported by the CARITRO Foundation through grant 2014.0372. The work of RB was partially supported by the Russian Science Foundation through the project entitled Global Optimization, Supercomputing, and Applications, under grant 15-11-30022.

References

[1] B. Peintner, P. Viappiani, N. Yorke-Smith, Preferences in Interactive Systems: Technical Challenges and Case Studies, AI Magazine 29 (2008) 13–24.

[2] J. G. March, Bounded Rationality, Ambiguity, and the Engineering of Choice, The Bell Journal of Economics 9 (1978) pp. 587–608.

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

[4] G. Pigozzi, A. Tsoukiàs, P. Viappiani, Preferences in artificial intelligence, Annals of Mathematics and Artificial Intelligence 77 (2016) 361–401.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] L.-F. Yu, S.-K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, S. J. Osher, Make it home: Automatic optimization of furniture arrangement, ACM Trans. Graph. 30 (2011) 86:1–86:12.

[14] P. Merrell, E. Schkufza, V. Koltun, Computer-generated residential building layouts, ACM Trans. Graph. 29 (2010) 181:1–181:12.

[15] Y.-L. Yang, J. Wang, E. Vouga, P. Wonka, Urban pattern: Layout design by hierarchical domain splitting, ACM Trans. Graph. 32 (2013) 181:1–181:12.

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

[17] R. Sebastiani, S. Tomasi, Optimization Modulo Theories with Linear Rational Costs, ACM Transactions on Computational Logics (2015). In print.

[18] G. A. Miller, The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information, The Psychological Review 63 (1956) 81–97.

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

[20] S. Teso, R. Sebastiani, A. Passerini, Structured learning modulo theories, Artificial Intelligence 244 (2017) 166–187.

[21] 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, Lecture Notes in Computer Science, Springer Verlag, 2011.

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

[23] R. Sebastiani, Lazy Satisfiability Modulo Theories, Journal on Satisfiability, Boolean Modeling and Computation, JSAT 3 (2007) 141–224.

[24] C. Barrett, R. Sebastiani, S. A. Seshia, C. Tinelli, Satisfiability Modulo Theories, Frontiers in Artificial Intelligence and Applications, IOS Press, pp. 825–885.

[25] R. Nieuwenhuis, A. Oliveras, On SAT Modulo Theories and Optimization Problems, in: Proc. Theory and Applications of Satisfiability Testing - SAT 2006, volume 4121 of *Lecture Notes in Computer Science*, Springer, 2006.

[26] A. Cimatti, A. Franzén, A. Griggio, R. Sebastiani, C. Stenico, Satisfiability modulo the theory of costs: Foundations and applications, in: Proc. Tools and Algorithms for the Construction and Analysis of Systems, TACAS, volume 6015 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 99–113.

[27] A. Cimatti, A. Griggio, B. J. Schaafsma, R. Sebastiani, A Modular Approach to MaxSAT Modulo Theories, in: International Conference on Theory and Applications of Satisfiability Testing, SAT, volume 7962 of *Lecture Notes in Computer Science*, Springer, 2013.

[28] R. L. Keeney, H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, 1976.

[29] T. Joachims, Optimizing search engines using clickthrough data, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, ACM, New York, NY, USA, 2002, pp. 133–142.

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

[31] J. Zhu, S. Rosset, R. Tibshirani, T. J. Hastie, 1-norm support vector machines, in: S. Thrun, L. K. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems 16, MIT Press, 2004, pp. 49–56.

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

[33] V. Conitzer, Eliciting Single-Peaked Preferences Using Comparison Queries, Journal of Artificial Intelligence Research 35 (2009) 161–191.

[34] A. D. Baddeley, G. Hitch, Working memory, Psychology of learning and motivation 8 (1974) 47–89.

[35] The magical number 4 in short-term memory: A reconsideration of mental storage capacity, Behavioral and Brain Sciences 24 (2001) 87–114.

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

[37] G. Pigozzi, A. Tsoukiàs, P. Viappiani, Preferences in artificial intelligence, Ann. Math. Artif. Intell. 77 (2016) 361–401.

[38] L. J. Savage, The Theory of Statistical Decision, Journal of the American Statistical Association 46 (1951) 55–67.

[39] P. Viappiani, P. Pu, B. Faltings, Conversational recommenders with adaptive suggestions, in: RecSys'07, pp. 89–96.

[40] P. Viappiani, C. Boutilier, Regret-based optimal recommendation sets in conversational recommender systems, in: RecSys'09, pp. 101–108.

[41] J. Zhang, P. Pu, A comparative study of compound critique generation in conversational recommender systems, in: International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 234–243.

[42] D. Sabin, R. Weigel, Product configuration frameworks-a survey, IEEE Intelligent Systems and their applications 13 (1998) 42–49.

[43] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based configuration: From research to business cases, Newnes, 2014.

[44] C. Boutilier, R. Brafman, C. Geib, D. Poole, A constraint-based approach to preference elicitation and decision making, in: AAAI Spring Symposium on Qualitative Decision Theory, Citeseer, pp. 19–28.

[45] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, D. Poole, Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements, J. Artif. Intell. Res.(JAIR) 21 (2004) 135–191.

[46] T. E. Allen, Cp-nets: From theory to practice, in: International Conference on Algorithmic DecisionTheory, Springer, pp. 555–560.

[47] F. Ricci, Recommender systems: Models and techniques, in: Encyclopedia of Social Network Analysis and Mining, Springer, 2014, pp. 1511–1522.

[48] P. Viappiani, C. Boutilier, Recommendation sets and choice queries: there is no exploration/exploitation tradeoff!, in: AAAI.

[49] D. Bollen, B. P. Knijnenburg, M. C. Willemsen, M. Graus, Understanding choice overload in recommender systems, in: Proceedings of the fourth ACM conference on Recommender systems, ACM, pp. 63–70.

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

[51] B. Settles, Active learning, Synthesis Lectures on Artificial Intelligence and Machine Learning 6 (2012) 1–114.

[52] J. Kremer, K. Steenstrup Pedersen, C. Igel, Active learning with support vector machines, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 4 (2014) 313–326.

[53] D. HA, The method of paired comparisons, Griffin, London, 1963.

[54] R. A. Bradley, M. E. Terry, Rank analysis of incomplete block designs: I. the method of paired comparisons, Biometrika 39 (1952) 324–345.

[55] R. D. Luce, Individual choice behavior: A theoretical analysis, Courier Corporation, 2005.

[56] R. L. Plackett, The analysis of permutations, Applied Statistics (1975) 193–202.

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

[58] 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.

[59] 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.

[60] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: A library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.

[61] R. C. Weng, C.-J. Lin, A Bayesian Approximation Method for Online Ranking, Journal of Machine Learning Research 12 (2011) 267–300.

[62] K. Tsukida, M. R. Gupta, How to Analyze Paired Comparison Data, Technical Report No. UWEETR-2011-004, Washington University, Dep. of Electrical Engineering, Seattle, USA, May 2011. (as of June 2015).

[63] D. Mcfadden, Economic Choices, American Economic Review 91 (2001) 351–378.

[64] M. Fränzle, C. Herde, T. Teige, S. Ratschan, T. Schubert, Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure, JSAT 1 (2007) 209–236.

[65] D. Jovanovic, L. M. de Moura, Solving non-linear arithmetic, in: B. Gramlich, D. Miller, U. Sattler (Eds.), IJCAR, volume 7364 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 339–354.

[66] D. Larraz, A. Oliveras, E. Rodriguez-Carbonell, A. Rubio, Minimal-Model-Guided Approaches to Solving Polynomial Constraints and Extensions, in: C. Sinz, U. Egly (Eds.), Theory and Applications of Satisfiability Testing - SAT'14 - 17th International Conference, volume 8561 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 333–350.

[67] 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.

[68] P. Dragone, S. Teso, A. Passerini, Constructive preference elicitation, Frontiers in Robotics and AI 4 (2018) 71.

[69] 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.

[70] 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. Workshop held in conjunction with the European Conference on-Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD 2010.

[71] V. T. Paschos, Applications of combinatorial optimization, Mathematics and Statistics Series, Wiley-ISTE, 2nd ed., 2014.

[72] 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.

[73] S. Teso, A. Passerini, P. Viappiani, Constructive preference elicitation by setwise max-margin learning, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pp. 2067–2073.

[74] S. Teso, P. Dragone, A. Passerini, Coactive critiquing: elicitation of preferences and features, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI Press, 2017, pp. 2639–2645.

[75] P. Dragone, S. Teso, A. Passerini, Constructive preference elicitation over hybrid combinatorial spaces, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI Press, 2018, pp. 2943–2950.