Learning Modulo Theories for preference elicitation in hybrid domains

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

[a]*Informatik Lehrstuhl 4, Technische Universität Dortmund, Otto-Hahn-Str. 16, D-44227 Dortmund, Germany.*
[b]*DISI - Dipartimento di Ingegneria e Scienza dell'Informazione, Università degli Studi di Trento, Via Sommarive 14, I-38123 Povo, TN, Italy.*

Abstract

This paper introduces CLEO, a novel preference elicitation algorithm capable of recommending complex objects in hybrid domains, characterized by both discrete and continuous attributes and constraints defined over them. 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 (DM) is modelled as a weighted combination of features. CLEO iteratively alternates a preference elicitation step, where pairs of candidate solutions 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 hybrid solutions 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 of candidate decisional features.

CLEO is the first preference elicitation algorithm capable of dealing with hybrid domains, thanks to the use of Max-SMT technology, while retaining uncertainty in the DM utility and noisy feedback. In so doing it adapts the recently introduced learning modulo theory framework to the preference

---

*Most of this work has been done while the author was with DISI.

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

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 focus towards optimal solutions, as well as its capacity to recover from suboptimal initial choices. While no competitors exist in the hybrid setting, 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, hybrid optimization.

---

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 the *preference elicitation* problem [1]. In principle, one may 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 configuration maximizing the utility function is recommended to the DM. However, this approach is impractical, for several reasons [2]:

- the user cannot usually define her preferences *a priori*, without seeing any tentative results. Only when facing 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 preferences are usually not affordable;

- in general, formalizing the user preferences as a mathematical model is not trivial: a model should capture the qualitative notion of preference and represent it as a quantitative function.

To handle the initial incomplete knowledge of the user utility, an incremental approach is usually adopted, where a configuration is recommended to the user based on *partial* preference information only. If the user is not satisfied by the tentative solution, she is asked for additional preference information and a refined configuration is suggested. This incremental process

2

needs techniques that can reason with partially-specified utility functions and take decisions under uncertain preference information. Furthermore, the interaction with human decision makers, with limited patience and bounded rationality, limits both the number and the complexity of the queries asked during the elicitation process, bounds the time needed for providing the recommendations and has to deal with inaccurate and inconsistent human feedback.

The main requirements for practical applicability of preference elicitation are [3]:

1. *real-time* interaction with the DM, where both the query generation and the solutions recommendation must be accomplished in no more than few seconds;
2. *robustness* to inconsistent and contradictory feedback from the DM characterizing the typical human decision making process;
3. *cognitively affordable* queries to the user, i.e., comparison queries;
4. *scalable* methods, that evaluate at each preference elicitation stage a number of candidate queries that grows not more than linearly in the cardinality of the solutions space.

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 way they make recommendations under uncertainty in the weight values. 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 [4, 5, 6]. In these papers, decisions under uncertainty are taken according to the *minimax regret criterion*: the configuration minimizing the worst-case loss with respect to the feasible utility functions is recommended.

The *Bayesian approach* [7, 3, 8, 9] maintains a probability distribution over the space of all possible weight values. 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 [10] 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 [10] introduces a preference elicitation strategy for soft constraint problems with missing preference values.

However, neither the works [4, 5, 6] based on the minimax regret nor the constraint-based approach [10] can handle inaccurate and contradictory human feedback. The Bayesian method proposed in [3] satisfies all main requirements for practical applicability discussed above. However, it can handle discrete attributes only, and it is hardly generalizable to the continuous case.

This paper introduces a novel algorithm which satisfies all the main principles for practical applicability of preference elicitation, allows to deal with hybrid domains and when applied to purely boolean problems consistently improves the state-of-the-art in terms of number of queries and quality of the returned solution. 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 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 *Maximum Satisfiability Modulo Theory* (Max-SMT) solver [11]. Max-SMT is a powerful recent formalism to optimize weighted formulae in a decidable first-order theory. Max-SMT

---

[1]In this simple example, each formula consists of a single predicate only. In the general case, arbitrary logic formulae (e.g., conjunctions or disjunctions of possibly negated predicates) are considered.

enables to describe candidate solutions of the preference elicitation task by using both discrete and continuous attributes simultaneously (*hybrid search domain*), thus improving the state-of-the-art of preference elicitation, which cannot handle hybrid search domains. Furthermore, Max-SMT enables to manage complex non-linear interactions among the attributes (for example, a cost attribute defined as a function of the remaining attributes), increasing the expressiveness. *Learning modulo theories* was recently introduced [12] 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.

The approach presented in this paper assumes a very limited amount of prior information about the task to be solved. The initial knowledge is limited to a set of *catalog attributes* used to describe the candidate solutions. The combinatorial formulation of the DM utility over the catalog attributes is initially unknown and needs to be learned by interacting with DM. 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 first-order logic formulae combining the predicates up to a certain degree. Only a small fraction of these candidate features is actually part of the unknown utility for a certain DM [13]. A sparsifying norm [14] is used during training in order 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 unknown Combinatorial utility function joint LEarning and Otimization.

An experimental evaluation on realistic problems defined over hybrid domains (i.e., with both discrete and continuous decisional attributes) and with inaccurate human feedback demonstrates the effectiveness of CLEO in focusing towards the optimal solutions, 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 [3].

A preliminary version of CLEO was presented in [15]. 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 solutions. Second, it considerably 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. A small introductory example of the preference elicitation tasks follows (Sec. 3). The CLEO algorithm is introduced in Sec. 4 and some of its main properties are analyzed in Sec. 5. Related work is discussed in Sec. 6, while Section 7 reports the experimental evaluation. Finally, a discussion including potential future research directions concludes the paper.

2. Notation and 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 (Max-SMT) to handle optimization tasks. Table 1 summarizes the notation used throughout the paper.

Symbol	Meaning
$\top, \bot$	Boolean values `true` and `false`
$x, y, z, \ldots$	Rational variables
$A_1, A_2, \ldots, A_n$	Catalog attributes (Boolean or rational 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. They can be atomic (Boolean attributes or predicates over rational attributes, e.g. $x + y < 3$) or the combination of atomic constraints by the logical connectives (e.g. $\neg has\_car \rightarrow dist\_supermarket \leq \theta$)
$\mathbb{I}_k(\boldsymbol{A})$	Indicator function for constraint φ_k over $\boldsymbol{A}$. It evaluates to one if φ_k is satisfied, to zero otherwise.
$\boldsymbol{\psi}(\boldsymbol{A})$	Feature (i.e., constraint) representation of configuration $\boldsymbol{A}$
$\psi_k(\boldsymbol{A}) = \mathbb{I}_k(A)$	Feature associated to constraint φ_k
$\boldsymbol{w}$	Weights

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

2.1. 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 a truth value assignment of the Boolean variables. Satisfiability Modulo Theory (SMT) [16, 17] extends SAT to decide about satisfiability of a first-order formula with respect to a *background theory* $\mathcal{T}$, like linear arithmetic over the rationals ($\mathcal{LRA}$) 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:

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

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

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) in order to progressively focus the search towards theory-consistent solutions or to state the unsatisfiability of the input SMT formula. A *T-solver* is a *specialized* reasoning method for the theory $\mathcal{T}$ integrated as submodule in the SMT solver. Usually a T-solver is a decision procedure developed to check the satisfiability of *conjunctions* of literals (i.e., atomic formulae and their negations) over theory $\mathcal{T}$. The generalization to arbitrary propositional structures is handled in conjunction with the SAT solver integrated in the SMT solver. For ease of exposition, here a single theory is assumed, but all the machinery described can be applied to arbitrary combinations of theories.

Let Δ be an SMT formula made of predicates in a certain theory $\mathcal{T}$. Its *Boolean abstraction* Δ^- is obtained replacing each i-*th* theory-specific predicate in Δ with a Boolean variable φ_i, producing a formula in plain propositional logic. If this propositional formula in unsatisfiable, the original formula Δ is also unsatisfiable and the whole SMT solver stops. Otherwise, the SAT solver finds a truth value assignment to the Boolean variables φ_i satisfying Δ^-, and presents it to the $\mathcal{T}$-solver to check for theory consistency. The $\mathcal{T}$-solver searches for an assignment of values to the theory variables which is consistent with the solution provided by the SAT solver: if the Boolean variable φ_i is assigned a value true (false), the corresponding i-*th* predicate must (not) be satisfied by the values assigned to the theory variables. Predicates are evaluated using the rules of the theory $\mathcal{T}$. If the $\mathcal{T}$-solver detects an inconsistency, it returns `unsat`, plus a *justification*, i.e. a subset of the truth value assignment provided by the SAT solver which is unsatisfiable according to the theory. The justification is an explanation of the inconsistency detected. This justification is added to the original formula, and the process is repeated until a theory-consistent solution is found, or the refined formula is not satisfiable.

Example 2.1. *Let Δ be the following SMT($\mathcal{LIA}$) formula:*

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

where x, y, z are integer-valued variables. Its Boolean abstraction Δ^- is:

$$\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \wedge (\varphi_4 \vee \varphi_5).$$

Suppose the SAT solver finds the following truth assignment satisfying Δ^-:

$$\varphi_1 = \top, \varphi_2 = \top, \varphi_3 = \top, \varphi_4 = \top, \varphi_5 = \top.$$

It corresponds to the following SMT($\mathcal{LIA}$) formula:

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

When asked to evaluate this formula, the $\mathcal{T}$-solver detects that it is theory inconsistent, since if z is set to 2 and both x and y must be larger than 2, the sum of the three variables cannot be less than or equal to 3. A justification provided by the the $\mathcal{T}$-solver to explain the inconsistency may be, e.g., the following constraint:

$$\neg(\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4)$$

which is included in Δ^- for the following calls to the SAT solver. A possible solution provided by the SAT solver for the refined Boolean abstraction:

$$\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \wedge (\varphi_4 \vee \varphi_5) \wedge \neg(\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4)$$

is the following truth assignment:

$$\varphi_1 = \top, \varphi_2 = \bot, \varphi_3 = \top, \varphi_4 = \bot, \varphi_5 = \top,$$

corresponding to the theory formula:

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

The $\mathcal{T}$-solver detects that this formula is theory consistent. It is satisfied, e.g., by the assignment:

$$x = 1, y = 0, z = 2.$$

The search process of the overall SMT solver now stops, since a solution of the input formula Δ has been found.

These solvers are termed *lazy* because of this incremental approach which generates constraints on demand, progressively refining the Boolean abstraction Δ^- by including additional theory-specific information.

Modern lazy SMT solvers introduce a number of refinements to this basic procedure, by pursuing a tighter integration between SAT and theory solvers. A common approach consists of pruning the search space for the SAT solver by calling the theory solver on partial assignments and propagating its results. Furthermore, modern lazy SMT solvers combine solving techniques from very heterogeneous domains. We refer the reader to [17, 18] for an overview on lazy SMT solving.

Max-SMT [19, 20, 21] generalizes SMT in the same way as Max-SAT does with SAT: rather than an assignment satisfying the input SMT formula, one maximizing the number of satisfied constraints is searched for. 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.

Let $\{(\varphi_1, w_1), \ldots, (\varphi_m, w_m)\}$ be a set of constraints with associated non-negative weights. The utility of any assignment is clearly smaller or equal than the sum of all weights $W = \sum_{i=1}^{m} w_i$ and larger than or equal to zero. The maximum-utility solution is identified by a branch and bound strategy, which progressively tightens the upper and lower utility bounds and solves plain SMT problems encoding these bounds in their formulation. Given a lower bound $\hat{W} < W$, a solution is enforced to have a utility larger than $\hat{W}$ by generating a set of m fresh Boolean variables and weights $\{(\bar{\varphi}_1, \bar{w}_1), \ldots, (\bar{\varphi}_m, \bar{w}_m)\}$ combined with the following constraints [19]:

$$
\begin{aligned}
\varphi_i \vee \bar{\varphi}_i & \quad \forall i \in \{1, 2, \ldots, m\} \\
\bar{\varphi}_i \rightarrow (\bar{w}_i = 0) & \quad \forall i \in \{1, 2, \ldots, m\} \\
\neg \bar{\varphi}_i \rightarrow (\bar{w}_i = w_i) & \quad \forall i \in \{1, 2, \ldots, m\} \\
\sum_{i=1}^{m} \bar{w}_i \geq \hat{W} &
\end{aligned}
$$

These constraints make any assignment with overall weight smaller than $\hat{W}$ inconsistent with the theory.

3. An introductory example

Le us consider a customer that aims at building her own house. For this purpose, she asks a real-estate company about potential housing locations. A very clear-headed person could formulate a request like:

> *I would like a house in a safe area, close to my parents and to the kindergarten, with a garden if there are no parks nearby. 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. Finally, I prefer a site well served by*

public transport, with the nearest metro station easily reachable on foot. My maximum budget is 300,000 Euro.

These desiderata can be encoded as an SMT problem as follows:

solve:

$$(\neg\varphi_1 \vee \varphi_2) \wedge \varphi_3 \wedge \varphi_4 \wedge \varphi_5 \wedge \varphi_6 \wedge \varphi_7 \wedge \varphi_8 \wedge \varphi_9$$

subject to:

$$\varphi_1 = A_2 \qquad\qquad \varphi_2 = A_1$$
$$\varphi_3 = (A_3 < \theta_1) \qquad\qquad \varphi_4 = (A_4 < \theta_2)$$
$$\varphi_5 = (A_5 < \theta_3) \qquad\qquad \varphi_6 = A_6$$
$$\varphi_7 = (A_7 < \theta_4) \qquad\qquad \varphi_8 = (A_8 < \theta_5)$$
$$\varphi_9 = (A_9 < \theta_6)$$
$$price(\boldsymbol{A}) \leq 300000$$

where the characteristics of the locations are defined by the set of catalog attributes $\boldsymbol{A}$ listed in Table 2. Function *price* computes the price of location $\boldsymbol{A}$ based on the values of its attributes.

name	description	type
A_1	garden	Bool
A_2	park nearby	Bool
A_3	crime rate	Ordinal
A_4	distance from parents	Real
A_5	distance from kindergarten	Real
A_6	cycling and walking facilities in the neighborhood	Bool
A_7	air-pollution index	Ordinal
A_8	public-transit service quality-index	Ordinal
A_9	distance from nearest metro station	Real
A_{10}	commercial facilities in the neighborhood	Bool
A_{11}	distance from downtown	Real

Table 2: Catalog attributes for the housing example.

If none of the locations available at the agency satisfies *all* constraints, the above problem has no solution. A more reasonable alternative consists of solving the optimization version of the above problem, which maximizes

the weighted sum of the satisfied constraints (i.e., a Max-SMT problem):

$$\operatorname*{argmax}_{\boldsymbol{A}} \quad \sum_{i=1}^{5} \bar{w}_i$$

subject to:
$$\varphi_1 = (A_3 < \theta_1)$$
$$\varphi_2 = (\neg A_2 \vee A_1)$$
$$\varphi_3 = ((A_4 < \theta_2) \wedge (A_5 < \theta_3))$$
$$\varphi_4 = (A_6 \wedge (A_7 < \theta_4))$$
$$\varphi_5 = ((A_8 < \theta_5) \wedge (A_9 < \theta_6))$$
$$price(\boldsymbol{A}) \leq 300000$$
$$\varphi_i \rightarrow (\bar{w}_i = w_i) \quad \forall i \in \{1, 2, \ldots, 5\}$$
$$\neg\varphi_i \rightarrow (\bar{w}_i = 0) \quad \forall i \in \{1, 2, \ldots, 5\}$$

where each constraint φ_i is associated to a weight w_i quantifying the (relative) utility of the constraint. The bound on the price is a hard constraint that needs to be satisfied, thus it has no weight.

A fully specified scenario like the one described here 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. The most natural scenario consists of an interactive process, with the customer evaluating candidate locations 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 automatizes this process.

Let us finally note that not all the catalog attributes describing candidate house locations may be relevant for a customer: in the 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. The CLEO algorithm can identify the subset of catalog attributes relevant for a certain customer.

4. The CLEO algorithm

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

Catalog attributes. CLEO assumes a *catalogue* of attributes which can be used to describe the configurations. Each configuration is an instantiation of the catalog attributes. These attributes can be either Boolean (e.g., `there is a garden`), ordinal (e.g., `crime rate`) or real (e.g., `distance to kindergarten`) variables (see Table 2 in the previous example for a list). A large number of attributes can be included, in order to increase the expressiveness of the method and enable fine-grained descriptions of the configurations. However, only a limited subset of the attributes may be relevant for a specific decision maker, and, in general, the subset varies when different users are considered. This section will show how CLEO identifies the subset of relevant attributes.

Hard constraints. Some combinations of attribute values may be infeasible. For example, in the above housing example, house locations with cost value smaller than a given threshold may not be available. Arbitrarily-complex hard constraints define the *feasible search space* of candidate configurations. The hard constraints are assumed to be known in advance. The CLEO algorithm provides to the DM *only feasible solutions* during the preference elicitation process.

Soft constraints. *Soft constraints* are defined over the catalog attributes. A soft constraint may or may not be satisfied by a feasible configuration. Each soft constraint is associated with a weight, defining the utility value of the constraint. *Positive* weights are associated with constraints expressing positive preferences of the DM (i.e., features that the preferred configuration should have), while *negative* weights are associated with constraints articulating negative preferences (i.e., features that the ideal configuration should *not* have). The absolute value of the weight defines how much the soft constraint is relevant for the DM (w.r.t. to the other soft constraints). A *zero*-weight identifies a constraint not considered by the DM.

Space of soft constraints. Soft constraints are atomic constraints or their combination. *Atomic* constraints are constructed from catalog attributes, by simply taking their values for Boolean variables, and constraining each

13

ordinal and real variable to be below a certain (variable-specific) threshold. In the case of non-Boolean variables, atomic constraints are thus predicates in first-order logic. More complex constraints can be constructed by arbitrary combinations of these building blocks. For example, `distance to kindergarten` $< \theta \wedge$ `distance to parents` $< \theta$, so that a car is not needed, or, `house with garden` $\vee$ `distance from nearest park` $< \theta$, so that open-air activities are possible.

These combinations are arbitrary logic formulae (e.g, conjunctions or disjunctions) of up to d atomic constraints. The maximal degree d contributes to limit the size of the soft constraints space, and is grounded on the bounded rationality of humans, who can simultaneously handle only a limited number of features.

The space of constructible soft constraints is clearly exponential in the size of the catalogue. In the following we show how CLEO manages the large dimensionality of the soft constraints space. For this purpose, let us define here the mapping function $\boldsymbol{\psi}(\boldsymbol{A})$ which projects configuration $\boldsymbol{A}$ into the space of all possible soft constraints, i.e., combinations of up to d atomic constraints. Each soft constraint φ_k is associated with its indicator function $\mathbb{I}_k(\boldsymbol{A})$ which evaluates to one if the constraint is satisfied and to zero otherwise. The feature (i.e., constraint) representation of configuration $\boldsymbol{A}$ is the vector obtained by concatenating the evaluation of each indicator function:

$$\boldsymbol{\psi}(\boldsymbol{A}) = (\mathbb{I}_1(\boldsymbol{A}), \mathbb{I}_2(\boldsymbol{A}), \dots, \mathbb{I}_m(\boldsymbol{A}))$$

In the following, the vector returned by function $\boldsymbol{\psi}$ and the space of all possible vectors returned by function $\boldsymbol{\psi}$ will be referred to as *feature vector* and *feature space*, respectively. The terms *feature* and *constraint* will thus be used interchangeably.

Combinatorial utility function. The DM utility function is represented by a subset of the soft constraints defined over the catalog attributes. The soft constraints involved in the definition of the utility function are associated a weight different from zero and encode the DM preferences. The utility of a configuration is the sum of the weights of the soft constraints satisfied by the configuration.

The above introduced feature vector $\boldsymbol{\psi}(\boldsymbol{A})$ enables the following compact formulation of utility function f:

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

where the weight vector $\boldsymbol{w}$ contains the weights associated with the candidate soft constraints. Due 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 actually be considered by the DM, resulting in an extremely sparse weight vector $\boldsymbol{w}$. This *sparsity assumption* will be accounted for when introducing the learning stage.

Learning phase. Learning amounts to find the weights for the utility function formulation in Eq. 1 matching the unknown DM preferences. Training examples for this phase consist of candidate configurations with their evaluation from the DM. Asking quantitative feedback such as real-valued scores is typically not affordable for a human DM [3]. A more realistic scenario consists of asking the DM to rank solutions by preference. We can thus formulate the problem as *learning to rank*, where the task is learning a function returning the same ranking as the one provided by the DM. We focus on the adaptation of SVM for ranking [22], which assumes pairwise ranking preferences, and enforces a (soft) large margin between the two predictions. However, we have an additional requirement, which is the sparsity assumption in the weight vector $\boldsymbol{w}$. Indeed, the feature vector contains all possible constraints (up to a certain complexity), and the learning phase should also perform some form of *constraint learning* by selecting a small set of relevant ones. Feature selection is in fact crucial to maximize the learning accuracy with data sets characterized by redundant and irrelevant features [23]. We favour feature selection by replacing the 2-norm of SVM with a 1-norm, which is a sparsifying norm encouraging solutions with few non-zero weights [23]. The resulting learning problem is:

$$
\min_{\boldsymbol{w}, \boldsymbol{\xi} \geq 0} \quad ||\boldsymbol{w}||_1 + C \sum_{\boldsymbol{A}^i \succ \boldsymbol{A}^j} \xi_{i,j}^2 \tag{2}
$$
$$
\text{subject to:} \quad \boldsymbol{w}^T(\boldsymbol{\psi}(\boldsymbol{A}^i) - \boldsymbol{\psi}(\boldsymbol{A}^j)) \geq 1 - \xi_{i,j}
$$
$$
\forall \, \boldsymbol{A}^i \succ \boldsymbol{A}^j \in \mathcal{D}
$$

where $\boldsymbol{A}^i \succ \boldsymbol{A}^j$ indicates that configuration $\boldsymbol{A}^i$ is ranked before $\boldsymbol{A}^j$ in the DM preference. Constraints enforce pairwise rankings to match DM preferences. A quadratic penalty $\xi_{i,j}^2$ is added to the objective function when a less preferred solution gets a utility score which is not sufficiently smaller than the more preferred one. The regularization parameter C trades-off matching

DM preferences with sparsity of the weight vector, and is optimized during the learning process as discussed further down.

Optimization phase. The ultimate goal of the algorithm is returning the best possible instance given the DM utility function. However, since the utility function is unknown, a preference elicitation phase is needed to gather information on DM preference and use it to refine the current approximation $\hat{f}$ of her utility. CLEO asks the DM for pairwise comparisons of configurations. The two configurations to be compared by the DM are generated by optimizing the learned utility function $\hat{f}(\boldsymbol{A})$ twice. Since the learned utility function is a weighted combination of soft constraints involving Boolean variables and first-order logic predicates defined over discrete and continuous variables, it is optimized by using an off-the-shelf Max-SMT solver, which can efficiently reason in these hybrid domains. The two optimization runs are performed based on the following principles:

1. the generation of top-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 focusing 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 [3]. In addition, the goal of CLEO is the identification of the solution preferred by the user (*learning to optimize*) rather than an accurate global approximation of the DM utility function (*learning per se*). This requires a shift of paradigm with respect to standard machine learning strategies, in order to model the relevant areas of the optimization fitness surface rather than reconstruct it entirely.

The second principle advocates the introduction of some diversification in the search, by exploring the neighbourhood of the best solution 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 [24].

16

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 accounts for the second principle, by enforcing a new solution $\boldsymbol{A}^{**}$ 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 solutions.

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 solutions. On the other hand, if some of them are needed to explain the DM preferences, driving their elicitation can allow to identify the deficiencies of the current approximation $\hat{f}$ and recover previously discarded relevant decisional items.

Overall algorithm. The pseudocode of the full CLEO algorithm is shown in Algorithm 1. It takes as input the set of catalog attributes, the set of atomic constraints, the set of hard constraints defining the feasible configurations, and returns the solution which is most preferred by the DM. In the initialization phase, the DM is asked for two pairwise comparisons of configurations selected by CLEO independently and uniformly at random in the feasible search space. Then a refinement loop begins, where at each iteration first an approximation of the DM utility function is learned using the current feedback. The refinement amounts at solving the "learning to rank" problem in Eq. (2), where $\mathcal{D}$ is the dataset of all pairwise preferences collected so far. The regularization parameter C is set to one in the first iteration, and fine-tuned by an internal cross validation on the training set in the following ones. With a slight abuse of notation, we write $\hat{f} \leftarrow$ argmax to indicate that $\hat{f}$ is the function whose weights $\boldsymbol{w}$ are the result of the maximization. The configuration $\boldsymbol{A}^*$ maximizing the learned utility function $\hat{f}$ is recommended to the DM. If she is not satisfied with the suggested solution, an additional optimizer $\boldsymbol{A}^{**}$ of $\hat{f}$ is generated, favouring diversity between $\boldsymbol{A}^*$ and $\boldsymbol{A}^{**}$ based on the diversification strategy defined above. The dataset $\mathcal{D}$ is then updated by including the comparison between $\boldsymbol{A}^*$ and $\boldsymbol{A}^{**}$ performed by the DM.

17

Data: Set of catalog attributes, set of atomic constraints, set of hard
       constraints

Result: Most preferred solution $\boldsymbol{A}^*$

    /* Initialization   */

1 Select three configurations uniformly at random

2 $\mathcal{D} \leftarrow$ ranking of configurations by DM

    /* Refinement   */

3 **while** *true* **do**

      /* learning   */

4

$$\hat{f} \quad \leftarrow \quad \underset{\boldsymbol{w},\boldsymbol{\xi} \geq 0}{\operatorname{argmin}} \|\boldsymbol{w}\|_1 + C \sum_{\boldsymbol{A}^i \succ \boldsymbol{A}^j} \xi_{i,j}^2$$

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

$$\forall\, \boldsymbol{A}^i \succ \boldsymbol{A}^j \in \mathcal{D}$$

      /* optimization   */

5   Recommend configuration $\boldsymbol{A}^* = \operatorname{argmax} \hat{f}$ to the DM

6   **if** *termination criterion is not satisfied* **then**

        /* preference elicitation   */

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

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

9   **end**

10 **end**

Algorithm 1: Pseudocode for CLEO.

Being an interactive process involving a human DM, the most obvious 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 conclusions.

5. CLEO properties

The CLEO algorithm has *no free parameters* to be manually tuned. The number of iterations does not need to be fixed at the beginning. The DM may ask for an additional iteration by comparing the recommended configuration

$\boldsymbol{A}^*$ with her own preferences. The termination criterion is thus represented by the satisfaction of the DM with $\boldsymbol{A}^*$. The regularization parameter C in Eq. (2) is set to one in the first iteration, and fine-tuned by internal cross-validation on the training set in the following ones. In the first iteration, two pairwise comparisons are asked to the DM, while in the following iterations a single pairwise comparison is asked. The configurations to be compared at the first iteration are generated by sampling independently and uniformly 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 [24]. In particular, the diversity of the proposed solutions helps the user to reveal the hidden preferences: in many cases the decision maker is not aware of all preferences until she sees them violated. For example, a user does not usually think about the preference for an intermediate airport until a solution suggests an airplane change in a place she dislikes [24].

The human cognitive capabilities bound the number of catalog attributes and the size d of soft constraints. The limited size of the Max-SMT instances 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. As a matter of fact, the learning task in Eq. (2) is convex, and thus guaranteed to converge to its global optimum, but the consistency of the learning algorithm with the true underlying user utility is only guaranteed asymptotically (i.e., provided that enough training data is available). On the other hand, CLEO does not need to learn the exact form of the DM utility function. The goal of our approach is indeed to elicit as few preference information from the DM as possible in order to identify her favourite solution (*learning to optimize*). For example, consider the toy DM utility function represented by the negation of a single ternary term: $\neg(\varphi_1 \wedge \varphi_2 \wedge \varphi_3)$. The approximation of the DM utility function consisting of the formula $\neg\varphi_1$ is sufficient to find one of the favourite DM solutions. More in general, only the shape of the utility function locally guiding the search to the correct direction is actually needed. Indeed the experimental results reported in Sec. 7 show the ability of CLEO in identifying the optimal solution and the improvements in the quality of the candidate solutions when increasing the number of refinement iterations

(anytime property).

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

1. *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;

2. *real-time* interaction with the DM. Due to the limited number n of catalog attributes and to the bounded size d of soft constraints, the learning phase (problem (2)) is accomplished in a negligible amount of time (w.r.t. the user response time). An analogous observation holds for the computational effort required by the optimization phase. 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;

3. *robustness* to inconsistent and contradictory human feedback. The adoption of regularized machine learning strategies in CLEO enables a robust approach that can handle inaccurate (pairwise) comparisons of solutions from the DM. Assuming that a user always provides accurate and consistent preference information is not realistic. Different factors may generate uncertain and inconsistent feedback from the DM, including occasional inattention, embarrassment when comparing very similar solutions or solutions which are very different from her favourite one, DM fatigue increasing with the number of queries answered;

4. *user cognitive load.* CLEO asks the user just for pairwise comparisons of candidate solutions. Most users are typically more confident in comparing solutions, providing qualitative judgments like "I prefer solution $\boldsymbol{A}^*$ to solution $\boldsymbol{A}^{**}$ ", rather than in specifying how much they prefer $\boldsymbol{A}^*$ over $\boldsymbol{A}^{**}$;

5. *scalability.* At each preference elicitation stage, just one candidate

query is considered by CLEO, independently of the cardinality of the configuration space. The adoption of 1-norm regularization for the formulation of the learning problem requires that the input catalog attributes are explicitly projected in the feature space, i.e., the space of all possible soft constraints. Dealing with the explicit projection Φ in Eq. (2) is tractable only for a rather limited number of catalog attributes and size of constraints d. However, this will typically be the case when interacting with a human DM. Research in psychology has indeed shown that humans cannot handle simultaneously more than few (7 ± 2) factors [13].

6. Related work

The problem of automatically learning utility functions and eliciting preferences is widely studied within the Artificial Intelligence community [25, 26]. Different approaches have been proposed to take decisions with partial preference information during the elicitation process. The uncertainty in the utility function is usually represented by a set of feasible utility functions (reasoning under strict uncertainty) [4, 5, 6], which is narrowed down when additional preference information is elicited, or by a probability distribution over possible utility functions (Bayesian approach) [7, 3, 8, 9], refined when additional knowledge of the DM preferences is obtained. Finally, a recent line of research [10] developed within the Constraint Programming community shares with CLEO the combinatorial formulation of the DM utility function (constraint-based preference elicitation). In the following, these approaches to preference elicitation are reviewed and compared with CLEO. We also motivate the choice of the Bayesian method introduced by Guo and Sanner in [3] as benchmarking algorithm in our experiments and summarize its main features. A more detailed description and discussion about the state-of-the-art methods for preference elicitation can be found in Appendix A.

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 about the DM preferences is decreased by restricting the feasible hypothesis

set, when relevant preference information is received during the elicitation process. This approach is often referred to as *reasoning under strict uncertainty* [25].

The *minimax regret* criterion [27] 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 most preferred solution $\boldsymbol{A}^*$ 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 therefore suggests a robust decision w.r.t. the worst possible case. The recent work in [4, 5, 6] 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 solution quality and convergence to provably-optimal results. On the other hand, these approaches assume perfect feedback from the DM and cannot handle the imprecise and contradictory information which is 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 (Bayesian approaches) consists of defining a *probability distribution* (or belief) over the candidate utility functions [7, 3, 8, 9]. The probabilistic framework offers a flexible approach to preference elicitation, handling the uncertainty in both utility and DM feedback. The *expected utility* of a configuration is defined as the average utility computed with respect to the probability distribution over the utility functions. The configuration maximizing the expected utility is usually recommended to the user. Therefore, under the Bayesian paradigm, robust decisions are taken to minimize risk in expectation. Queries are asked to the DM in order to increase the posterior probability of her utility. The probabilistic framework enables to estimate the informativeness of the candidate queries. At each stage of the preference elicitation process the maximally informative query is asked. The maximum expected loss (MEL) of taking a decision $\boldsymbol{A}$ is the maximum expected reduction in utility when choosing $\boldsymbol{A}$ instead of the DM most preferred solution $\boldsymbol{A}^*$, where expectation is taken over the probability distribution of the utility functions. The *value of information* (VOI) criterion

22

suggests the query generating the largest expected reduction in MEL. Exact computation of VOI, as well as exact computation of the posterior distribution over utility functions given the feedback, are extremely expensive. The state-of-the-art approaches [7, 3] resort to approximate solutions.

The closest approach to CLEO is the Bayesian method introduced by Guo and Sanner in [3] (referred to as GSM). Indeed, unlike the techniques based on minimax regret and on the constraint satisfaction formalism, CLEO and GSM satisfy all the main principles [3] needed for practical applicability of preference elicitation (see Sec. 1).

The GSM algorithm [3] searches for the configuration preferred by the DM within a given set of candidates. The configurations are described by n *discrete* attributes $x_1, \ldots, x_n$, where the k-*th* attribute is assigned values from a finite set X_k with cardinality $|X_k|$. The user utility functions are represented by a weight vector $\mathbf{w}$ with dimension $\sum_{k=1}^{n} |X_k|$ specifying the utility of *each* attribute value in X_k for *each* attribute k. This modelling choice assumes *preferential independence* among the set of attributes.

The uncertainty about the user preferences is represented by considering the weight vector $\mathbf{w}$ as a multivariate continuous random variable and by maintaining a probability distribution $\Pr(\mathbf{w})$, which is incrementally refined. Different strategies are defined to select the query to be asked at each refinement stage. Since GSM asks pairwise comparisons to the DM, in principle the VOI of each possible pairwise comparison has to be estimated. This query strategy, termed *informed VOI*, thus scales quadratically with the number of configurations and its computational cost is affordable for small search spaces only. In the experiments reported in [3], already 20 configurations prevent its application, even if the probabilities of the two possible answers to a pairwise comparison are assigned fixed arbitrary values (*uninformed VOI* strategy) rather than the values estimated from the elicited preference information. 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). For scalability purposes, the authors also suggest an alternative query strategy which does not use the VOI criterion to rank a set of candidate comparisons. At each preference elicitation just one query is considered, namely the comparison between the configuration $\mathbf{x}^*$ with greatest expected utility and the solution $\mathbf{x}^{EL}$ maximizing the expected loss of recommending $\mathbf{x}^*$ instead of $\mathbf{x}^{EL}$ (*simplified VOI* strategy).

Unlike CLEO, GSM is conceived for instances characterized by purely

discrete attributes, and cannot tackle preference elicitation tasks over hybrid domains. 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 [10] articulates the user preferences in terms of *soft* constraints and introduces constraint optimization problems where the DM preferences are not completely known before the solution process starts. In soft constraints each assignment to the variables of one constraint is associated with a preference value taken from a preference set. The preference value represents the level of desirability of the assignment to the variables of the constraint. As the preference score is associated to a partial assignment to the problem variables, it represents a *local* preference value. The desirability of a complete assignment is defined by a *global* preference score, computed by applying a combination operator to the local preference values. A set of soft constraints generates an order (partial or total) over the complete assignments of the variables of the problem. Given two solutions of the problem, the preferred one is selected by computing their global preference levels. Preference elicitation strategies have been introduced [10] to deal with scenarios where preference information is partially unknown. Some of the local preference values attached to soft constraints are assumed to be missing, and the DM is asked for an explicit feedback on specific assignments for these constraints, in terms of score values quantifying her preference for a certain assignment. In comparison to this approach based on the Constraint Programming formalism, CLEO assumes a much more limited amount of initial knowledge about the problem at hand. In [10], decision variables, soft constraint topology and structure are assumed to be known in advance and the incomplete initial information consists of missing local preference values only. CLEO assumes complete ignorance about the structure of the constraints over the decisional variables of the user. The initial problem knowledge is limited to a set of catalog attributes. CLEO extracts the decisional items of the DM from the set of catalog attributes and learns the weighted constraints constructed from them modeling the DM preferences.

Furthermore, the technique in [10] is based on local elicitation queries, with the final user asked to reveal her preferences about assignments for specific soft constraints. Global preferences or bounds for global preferences associated to complete solutions of the problem are derived from the local

preference information. CLEO goes in the opposite direction: it asks the user to compare complete solutions and learns local utilities (i.e., the weights of the soft constraints of the logic formula) from global preference values. In many cases, recognizing appealing or unsatisfactory global solutions may be much easier than defining local utility functions, associated to partial solutions. For example, while scheduling a set of activities, the evaluation of complete schedules may be more affordable than assessing how specific ordering choices between couples of activities contribute to the global preference value. Furthermore the algorithm in [10] asks the DM for quantitative evaluations of partial solutions: she does not just rank couples of activities, she provides score values quantifying her preference for the partial activity rankings, a much more demanding task. Finally, the approach in [10] assumes consistent and accurate quantitative feedback from the DM. Under this assumption, the optimality of the recommended solution is guaranteed. However, this approach cannot be applied in our realistic experimental setting characterized by the noisy human feedback.

7. Experimental results

The following empirical evaluation demonstrates that CLEO can handle realistic preference elicitation tasks defined over hybrid domains and with uncertain human feedback. No alternative algorithm capable of tackling these preference elicitation tasks is currently available (see Sec. 6). To overcome this limitation, our experimental work consists of two phases. First, CLEO is tested over a couple of realistic preference elicitation tasks with the above features. For this purpose, a benchmark of Max-SMT problems is defined, involving both discrete and continuous decisional variables. In a second step, a set of simplified synthetic problems with discrete decisional variables only is introduced, in order to compare CLEO with the existing preference elicitation algorithms. In particular, we consider Boolean decisional attributes only and generate a set of synthetic Maximum-Satisfiability (Max-SAT) benchmarks. In this simplified setting, the benchmarking preference elicitation algorithm is the method by Guo and Sanner [3].

For the experiments performed, the mapping function ψ in CLEO projects configurations into the space of all possible conjunctions of up to three atomic constraints (i.e., $d = 3$). The next section describes the well-known noisy response model used in both Max-SMT and Max-SAT experiments for simulating inaccurate and inconsistent feedback provided by the DM during the

preference elicitation process.

7.1. Noisy response model for human feedback

In the experiments the feedback from the user is assumed to be affected by the inaccuracies and inconsistencies. The user ranks configurations $\boldsymbol{A}$ based on a latent utility function $f(\boldsymbol{A})$. In particular, configuration $\boldsymbol{A}^i$ is preferred to configuration $\boldsymbol{A}^j$, i.e., $\boldsymbol{A}^i \succ \boldsymbol{A}^j$, 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 the assumption of independent and identically distributed Gaussian noise, the probability that the user prefers configuration $\boldsymbol{A}^i$ to configuration $\boldsymbol{A}^j$ is defined as follows:

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

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.

The above user response model, linking pairwise comparisons to a continuous latent utility function, has been widely used in the economic and psychological studies to describe the individual choice behaviour of humans [28, 29, 30]. It is known as the Thurstone-Mosteller or Probit model. In our experimental setting σ^2_{noise} is fixed to 10, to have noise values comparable with the latent utility values $f(\boldsymbol{A})$.

7.2. Realistic preference elicitation tasks over hybrid domains

CLEO is tested over a benchmark of Max-SMT problems, formulating realistic preference elicitation tasks. The Max-SMT tool used for the experiments is the "Yices" solver [31] (version 1.0), which is publicly available at `http://yices.csl.sri.com/` (as of August 2015). Each point of the curves

depicting our results is the median value over 400 runs with different random seeds.

Max-SMT is a recent research area. Even if existing results [11] indicate that Max-SMT solvers can efficiently address real-world problems, to the best of our knowledge no well-established publicly available Max-SMT benchmarks exist and preference elicitation tasks have not been encoded into Max-SMT instances yet.

In this work, we modelled a *scheduling* problem as a Max-SMT instance, where the DM expresses her preferences about the candidate schedules of a set of jobs. In the spirit of real-world recommendation tasks, we also design a *housing* problem aimed at selecting a location for building a house. The formulation consists of both unknown soft constraints representing the user preferences and known hard constraints defining the feasible search space. The housing problem is challenging, due to complex non-linear relationships among decision variables. For example, the variable encoding the cost of the location is defined as a function of the remaining decision variables. The results obtained by CLEO over both the preference elicitation tasks are discussed below.

7.2.1. Scheduling problem

A set of five jobs must be scheduled over a given period of time. Each job has a fixed known duration, the atomic constraints define the overlap of two jobs or their non-concurrent execution. The user unknown utility function is generated by selecting uniformly at random weighted conjunctions of atomic constraints. The solution of 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 of the user 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 i-*th* job, respectively. If s_i is scheduled before s_j, the constraint expressing the overlap of the two jobs is $s_j - s_i < d_i$, while their non-concurrent execution is encoded by $s_j - s_i \geq d_i$. Let us note that there are 40 possible constraints for a set of 5 jobs. The maximum size of the soft constraints is assumed to be three. The weights of soft constraints are distributed uniformly at random in the range $[1, 100]$.

CLEO is tested over a benchmark of randomly generated utility functions according to the couple (*number of decisional features, number of soft constraints*). The decisional features are the atomic constraints appear-
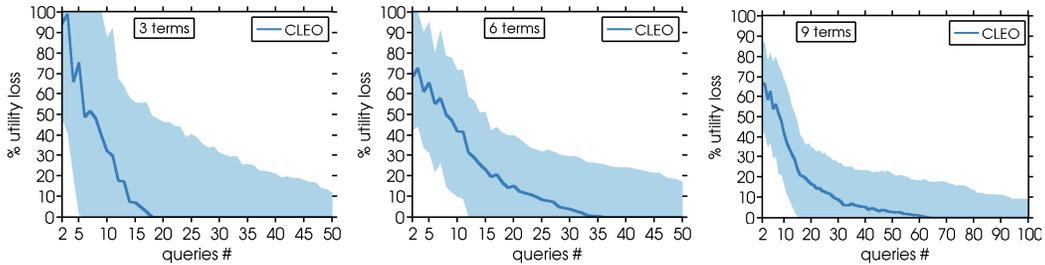
Figure 1: Performance of CLEO in solving the scheduling problem. The y-axis reports the percentage utility loss, while the x-axis contains the number of pairwise-comparison queries asked so far. The curve reports the median values observed over 400 runs of CLEO, while the shaded area denotes the range among the 25th and the 75th percentiles. Please note the different range of the x-axis in the case of nine soft constraints.

ing in the soft constraints. We generate functions for the following values: $\{(5, 3), (10, 6), (15, 9)\}$. Each DM utility has at least two soft constraints with a size of three. Let's underline once more that utility functions with more that few factors or factors with many terms are unrealistic when considering human DM [13].

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 solution, 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 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 substantial improvement in the quality of the recommended solution is achieved by CLEO when additional queries are asked to the DM (anytime property). Furthermore, CLEO identifies the DM preferred solution 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 solution. With 9 soft constraints, 64 pairwise comparisons are required on average to recommend the DM preferred solution. However, with 40 queries, a percentage utility loss within 5.5% is obtained. The shaded area shows that CLEO identifies the DM preferred solution quite consistently when increasing the number of queries (the interquartile range is within 25% after 35 queries even in the case of nine soft constraints).

7.2.2. Housing problem

We consider a customer planning to build her own house and judging potential housing locations provided by a real estate company (henceforth the *housing* problem). There are different locations available 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 have a family house with a huge garden and pool), etc. The customer may formulate her judgments by considering a description of the housing locations based on a predefined set of parameters, including, e.g., crime rate, distance from downtown, location-based taxes and fees, public transit service quality, walking and cycling facilities, proximity to commercial facilities or green areas, etc. Many of these parameters may be uninformative, as they do not represent any decisional criterion for the customer. Furthermore, hard constraints defining the feasible locations 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 4) defining feasible housing locations and known in advance 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 housing locations 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. Soft constraints are represented by weighted conjunctions of both predicates in the linear arithmetic theory and Boolean variables, in the case of attributes number $2, 3, \ldots, 6$ in Table 3. For example, one predicate may model the preference for a location with distance from nearest free parking smaller than a given threshold, while a Boolean variable encodes, e.g., the aspiration for houses with garage.

We generated a set of 40 predicates, i.e., atomic constraints. The user unknown utility function is composed of soft constraints with two or three predicates, with at least one soft constraint with three predicates. The maximum number of predicates in a soft constraint is assumed to be known. The

Table 3: Catalog attributes for the housing problem.

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

weights of soft constraints are integer values selected uniformly at random in the range $[1, 100]$.

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 decisional features, number of soft constraints*): $\{(5, 3), (10, 6), (15, 9)\}$, where the decisional features are the predicates appearing in the soft constraints. The promising results observed for the scheduling problem are confirmed, even though the housing problem is much harder, due to complex non-linear interactions among the decisional attributes. When increasing the number of queries asked, the quality of the solution 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 solution 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 increas-

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

ing the number of queries asked, showing that CLEO recommends better quality solution more consistently. However, in the case of three soft constraints, the interquartile range 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 solution 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 in order 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
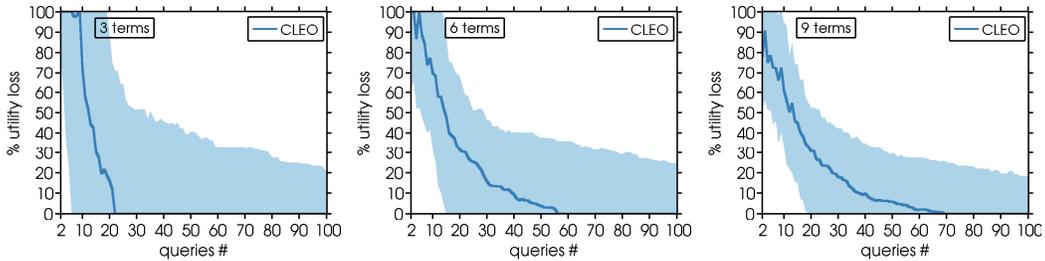
Figure 2: Performance of CLEO while solving the housing problem.

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 the unknown DM utility function is a *weighted Maximum Satisfiability* (Max-SAT) instance consisting of the weighted combination of the Boolean terms. The benchmarking algorithm is the GSM method [3] described in Sec. 6.2.

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 [3] 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 [3] are too slow, as pointed out also by the authors themselves 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.

Fig. 3 reports the percentage utility loss of the recommended configuration w.r.t the DM preferred solution for an increasing number of pairwise comparisons asked so far. The curves report the median values observed over 200 runs for CLEO (darker solid line or blue solid line if viewed in colour) and GSM (lighter dashed line or red dashed line if viewed in colour). The shaded areas depict the interquartile range measuring the dispersion around
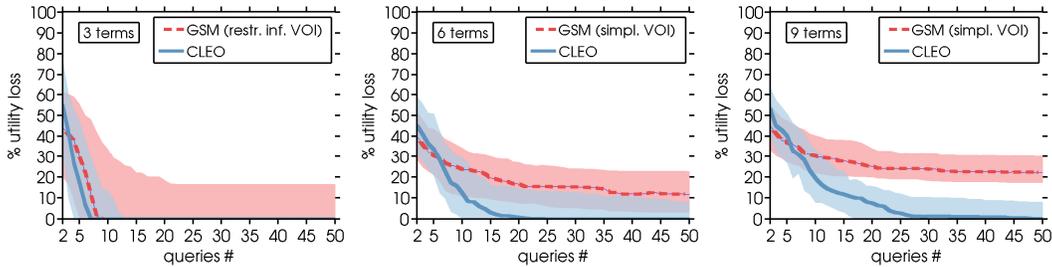
Figure 3: Performance of the CLEO (darker solid line or blue solid line if viewed in colour) and GSM (lighter dashed line or red dashed line if viewed in colour) algorithms over the Boolean problems. The y-axis reports the percentage utility loss, while the x-axis contains the number of pairwise comparisons asked so far. The curves report the median values observed over 200 runs, while the shaded area denotes the range among the 25th and the 75th percentiles of the observations. Best viewed in colour.

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 solution. 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 solution with a null interquartile range (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 solution, 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 solution 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 solutions 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.

8. Conclusions

This paper introduces CLEO, a preference elicitation algorithm that, unlike existing approaches, handles preference elicitation tasks defined over hybrid domains and with uncertain human feedback. A combinatorial formulation of the unknown DM utility function is adopted. CLEO consists of an incremental procedure, iteratively optimizing the learned approximation of DM utility function to generate candidate solutions and refining the approximation based on the human feedback received. Simple pairwise comparison queries are asked to the DM.

CLEO assumes *very limited initial knowledge*. In detail, since different decision makers 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, in order 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 towards the optimal solutions, its robustness, as well as its ability to recover from suboptimal initial choices.

Our experiments involve preference elicitation tasks over hybrid domains, with uncertain 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 increasing the complexity of the preference elicitation task.

CLEO can be generalized in a number of directions. The learning stage employs a ranking loss function based on pairwise preference evaluation. More complex ranking losses have been proposed in the literature (see for instance [32]), especially to increase the importance of correctly ranking the highest scoring solutions, and could be combined with 1-norm regularization.

Active learning is a hot research area and a broad range of different approaches has been proposed (see [33] for a review). The simplest and most common framework is that of *uncertainty sampling*: the learner queries the instances on which it is least certain. However, the ultimate goal of a recommendation or optimization system is selecting the best instance(s) rather than correctly modeling the underlying utility function. The query strategy should thus tend to suggest good candidate solutions and still learn as much as possible from the feedback received. Typical areas where research on this issue is quite popular are single- and multi-objective interactive optimization [34] and information retrieval [35]. The need to trade-off multiple requirements in this active learning setting is addressed in [36] where the authors consider relevance, diversity and density in selecting candidates. Our future research will consider the application of these active learning techniques. The performance of our method indeed depends on the trade-off between the identification of candidates solutions satisfying the DM (i.e., solutions optimizing the current learned preference model) and the generation of *informative* training examples for the following refinement of the learned model.

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 [3] 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). This is an interesting and challenging direction for future research.

Another research direction is the extension of our approach to handle feedback from multiple DMs [37]. In particular, an interesting case study is the exploitation of preferences of previous DMs to minimize the elicitation effort for a new user [7, 9]. We also plan to extend our algorithm to tackle preference drift [38], 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 [39]), 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 [15] when learning quantitative scores. As our previous results seem to indicate [15], this can produce a degradation in the quality of returned solutions when the utility function is very sparse. Kernelized versions of zero-norm regularization [40] could be tried in order to enforce sparsity in the projected space if needed. Let us however 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.

Acknowledgments

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] 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.

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

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

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

[7] 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.

[8] 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.

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

[10] 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.

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

[12] S. Teso, R. Sebastiani, A. Passerini, Structured learning modulo theories, Artificial Intelligence (2015).

[13] 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.

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

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

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

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

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

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

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

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

[29] 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).

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

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

[32] 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.

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

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

[35] 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.

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

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

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

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

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

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

Appendix A. Additional discussion about the state-of-the-art of preference elicitation

This section reviews two notable state-of-the-art approaches for preference elicitation: the body of work adopting the Minimax regret criterion [4, 5, 6] and the more recent line of research [10] developed within the Constraint Programming community. In particular, the latter method shares with CLEO a constraint-based approach to preference elicitation, resulting in a combinatorial formulation of the DM preferences. However, both state-of-the-art methods are not thus suitable for the realistic recommendation tasks considered in our experimental setting, characterized by inaccurate and inconsistent human feedback. In the following, we review these alternative approaches in detail and compare them with CLEO.

Appendix A.1. Minimax regret-based approaches

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

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

$$R(\mathbf{x}, \mathbf{x}', U) = \max_{u \in U} \ u(\mathbf{x}') - u(\mathbf{x}) \tag{A.1}$$

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

$$MR(\mathbf{x}, U) = \max_{\mathbf{x}'} \ R(\mathbf{x}, \mathbf{x}', U) \tag{A.2}$$

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

$$MMR(U) = \min_{\mathbf{x}} \ MR(\mathbf{x}, U) \tag{A.3}$$

and the configuration $\mathbf{x}^r = \mathrm{argmin}\ MR(\mathbf{x}, U)$ minimizing the maximum regret is the configuration recommended to the DM by the minimax regret

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

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

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

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

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

Appendix A.1.1. Comparison with CLEO

While the CLEO is a preference elicitation method approximately correct with high probability, the minimax regret-based approaches assume an adversarial entity that acts to maximize the DM regret and they aim at beating the adversary by recommending the best configuration with respect to the worst case loss. However, this adversarial model is not always strongly motivated by real-world applications, where users are typically interested in

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

Appendix A.2. Preference elicitation methods based on constraint satisfaction

Recent work in the field of constraint programming [10] shares with CLEO the combinatorial approach to model user preferences. It defines the user preferences in terms of *soft* constraints and introduces constraint optimization problems where the DM preferences are not completely known before the solving process starts. Let us first briefly describe the c-semiring formalism [41] adopted in paper [10] to model soft constraints.

In soft constraints, a generalization of hard constraints, each assignment to the variables of one constraint is associated with a preference value taken from a preference set. The preference value represents the level of desirability of the assignment to the variables of the constraint. As the preference score is associated to a partial assignment to the problem variables, it represents a *local* preference value. The desirability of a complete assignment is defined by a *global* preference score, computed by applying a combination operator to the local preference values. A set of soft constraints generates an

order (partial or total) over the complete assignments of the variables of the problem. Given two solutions of the problem, the preferred one is selected by computing their global preference levels. Soft constraints are represented by an algebraic structure, called *c-semiring* (where letter "c" stays for "constraint"), providing two operations for combining ($\times$) and comparing ($+$) preference values. In detail, the c-semiring is a tuple $(A, +, \times, \mathbf{0}, \mathbf{1})$ where:

- A is a set and $\mathbf{0}, \mathbf{1} \in A$;

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

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

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

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

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

The generality of the semiring-based soft constraint formalism permits to express several kinds of preferences, including partially ordered ones. For example, different instances of c-semirings encode weighted or probabilistic soft constraint satisfaction problems [42]. However, the c-semiring formalism can model just *negative* preferences. First, the best element in the ordering induced by $\leq_A$, denoted by $\mathbf{1}$, behaves as indifference, since $\forall a \in A, 1 \times a = a$. This result is consistent with intuition: when using only negative

preferences, indifference is the best level of desirability that can be expressed. Furthermore, the combination of desirability levels returns a lower overall preference, since $a \times b \leq_A a, b$, again consistently with the fact of dealing with negative preferences.

Preference elicitation strategies have been introduced [10] within this formalism in order to deal with scenarios where preference value information is partially unknown. Some of the local preference values attached to soft constraints are assumed to be missing, and the DM is asked for an explicit feedback on specific assignments for these constraints, in terms of score values quantifying her preference for a certain assignment. The elicitation strategy is aimed at minimizing the number of queries to the DM.

Appendix A.2.1. Comparison with CLEO

Concerning expressivity of the representation formalisms, the work in [43] shows how to encode semiring-based soft constraint satisfaction problem (SCSP) instances into equivalent weighted MAX-SAT formulations. Each solution of the latter instance corresponds to a solution of the former one. Details on the encoding algorithm can be found in Appendix A.2.2. The rationale for the MAX-SAT encoding is the exploitation of the efficient and widely studied techniques implemented in modern SAT solvers, which can efficiently handle large-size structured problems [44]. The encoding can in principle be applied also to SCSPs with continuous decision variables or discrete variables defined over large size finite domains, possibly however at the cost of a significant blow-up in the translation. In this case, one may cast the SCSP instance into a weighted MAX-SMT rather than a weighted MAX-SAT formulation.

Concerning the preference elicitation setting, our formulation assumes a much more limited amount of initial knowledge about the problem to be optimized. In the work on preference elicitation for SCSPs [10], decision variables, soft constraint topology and structure are assumed to be known in advance and the incomplete initial information consists only of missing local preference values. CLEO assumes complete ignorance about the structure of the constraints over the decisional variables of the user. The initial problem knowledge is limited to a set of catalog attributes. CLEO extracts the decisional items of the DM from the set of catalog attributes and learns the weighted constraints constructed from them modeling the DM preferences. If the MAX-SAT encoding is applied to the SCSP with missing preferences, it produces a Boolean formula where some of the weights of the terms are

not known. On the other hand, CLEO handles MAX-SAT instances where both the constraints and their associated weights are initially unknown and are learnt by interacting with the DM.

Furthermore, the technique in [10] is based on *local* elicitation queries, with the final user asked to reveal her preferences about assignments for specific soft constraints. *Global* preferences or bounds for global preferences associated to complete solutions of the problem are derived from the local preference information. CLEO goes in the opposite direction: it asks the user to compare complete solutions and learns local utilities (i.e., the weights of the constraints of the logic formula) from global preference values. In many cases, recognizing appealing or unsatisfactory global solutions may be much easier than defining local utility functions, associated to partial solutions. For example, while scheduling a set of activities, the evaluation of complete schedules may be more affordable than assessing how specific ordering choices between couples of activities contribute to the global preference value. Furthermore the preference elicitation technique in [10] asks the DM for quantitative evaluations of partial solutions: she does not just rank couples of activities, she provides score values quantifying her preference for the partial activity rankings, a much more demanding task.

In order to reduce the embarrassment of the decision maker when specifying precise preference scores, interval-valued constraints [45] allow users to state an interval of utility values for each instantiation of the variables of a constraint. As a matter of fact, the informal definitions of degrees of preference such as "quite high", "more or less", "low" or "undesirable" cannot be naturally mapped to precise preference scores. However, the technique described in [45] requires the user to provide all the information she has about the problem (in terms of preference intervals) *before* the solving phase, without seeing any optimization result.

Even if interval-valued constraints [45] have been introduced to handle uncertainty in the evaluations of the DM, inconsistent preference information is not addressed [10]. This is a requirement to retain the optimality guarantees provided by the preference elicitation strategy. Conversely, CLEO trades optimality for robustness and can effectively deal with imprecise information from the DM, modelled in terms of inaccurate ranking of the candidate solutions.

Finally, while the work in [10] considers *unipolar* preference problems, modeling just negative preferences, CLEO naturally accounts for bipolar preference problems, with the final user specifying what she likes and what

she dislikes. Bipolar preference problems provide a better representation of the typical human decision process, where the degree of preference for a solution reflects the compensation value obtained by comparing its advantages with the disadvantages. Let us note that the work in [42] extends the soft constraint formalism to account for bipolar preference problems.

Appendix A.2.2. Econding SCSP into weighted MAX-SAT instances

The work in [43] introduces a method to encode a semiring-based soft constraint satisfaction problem (SCSP) instance into a weighted MAX-SAT instance, with each solution of the generated MAX-SAT instance corresponding to a solution of the original SCSP. With no loss of generality, assume a soft constraint problem with n variables $v_1, \ldots v_n$ having domain $D_1, \ldots D_n$, and m constraints $c_1, \ldots c_m$. Each instantiation of the variables of a constraint c_j, $j = 1 \ldots m$, is associated with a value from the c-semiring $(A, +, \times, \mathbf{0}, \mathbf{1})$. For each variable v_i, $i = 1 \ldots n$, and each value $d \in D_i$, a Boolean variable $b_{i,d}$ is introduced. When $b_{i,d}$ is set to true then v_i is assigned the value $d \in D_i$. The variables $b_{i,d}$, $i = 1 \ldots n$, $d \in D_i$, represent the Boolean variables of the weighted MAX-SAT problem.

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

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

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

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

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

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

v_1	v_2	preference value		num	term	weight
1	1	10		1	$(b_{1,1} \wedge b_{2,1})$	10
1	2	40		2	$(b_{1,1} \wedge b_{2,2})$	40
1	3	50		3	$(b_{1,1} \wedge b_{2,3})$	50
2	1	5		4	$(b_{1,2} \wedge b_{2,1})$	5
2	2	10		5	$(b_{1,2} \wedge b_{2,2})$	10
2	3	30		6	$(b_{1,2} \wedge b_{2,3})$	30
3	1	5		7	$(b_{1,3} \wedge b_{2,1})$	5
3	2	5		8	$(b_{1,3} \wedge b_{2,2})$	5
3	3	10		9	$(b_{1,3} \wedge b_{2,3})$	10

Table A.5: (left) example of soft constraint. The DM prefers assignments with $v_1 < v_2$. (Right) weighted Boolean terms encoding the soft constraint defined in the left table. When the Boolean variable $b_{i,d} : d \in D_i$, is set to true then v_i is assigned the value d.

associated to the assignment. Given the six Boolean variables $b_{1,d}$ and $b_{2,d}$ with $d = 1, 2, 3$ defined as above, the soft constraint in Table A.5 (left) is encoded into the set of Boolean terms in Table A.5 (right).

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

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

with weight equal to 10. Furthermore, each *at-least-one-value* and *at-max-one-value* hard clause h can be cast into a soft clause represented by its negation $\neg h$ and with associated the semiring value $\mathbf{0}$ [43]. The value $\mathbf{0}$ is indeed both the minimum value in the partial order defined by the relation $\leq_A$ and the absorbing element for the operator $\times$ combining the semiring values. Therefore, a candidate solution $\mathbf{b}$ of the generated MAX-SAT instance that does not satisfy one of these soft clauses receives the minimum semiring value $\mathbf{0}$. However, this implementation of the hard clauses does not allow to discern infeasible solutions from feasible ones with lowest possible preference, i.e., feasible solutions getting the lowest semiring value.

Given the generated MAX-SAT formulation, the optimization task consists of finding the assignment $\mathbf{b}^*$ to the Boolean variables $b_{i,d}$, $i = 1 \ldots n$,

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