

Regret-based Utility Elicitation in Constraint-based Decision Problems

Craig Boutilier¹, Relu Patrascu¹, Pascal Poupart¹, and Dale Schuurmans²

¹ Dept. of Computer Science, University of Toronto, Toronto, ON, M5S 3H5, CANADA,
cebly,relu,ppoupart@cs.toronto.edu

² Department of Computing Science, University of Alberta, Edmonton, AB, T6G 2E8,
CANADA, dale@cs.ualberta.ca

Abstract. Constraint-based optimization requires the formulation of a precise objective function. However, in many circumstances, the objective is to maximize the utility of a specific user among the space of feasible configurations (e.g., of some system or product). Since elicitation of utility functions is known to be difficult, we consider the problem of incremental utility elicitation in constraint-based settings. Assuming graphical utility models, and adopting the *minimax regret decision criterion* for optimization in the presence of imprecise utility, we describe several elicitation strategies that require the user to answer only binary (bound) queries on the utility model parameters. While a theoretically motivated algorithm can provably reduce regret quickly (in terms of number of queries), we demonstrate that, in practice, heuristic strategies perform much better, and are able to find optimal (or near-optimal) configurations with far fewer queries, while leaving much of the utility function unspecified.

1 Introduction

The development of automated decision software is a key focus within decision analysis [11, 21, 18] and AI [8, 9, 4]. To deal with different users, some form of preference elicitation must be undertaken in order to capture specific user preferences to a sufficient degree to allow an (approximately) optimal decision to be taken. Different approaches to this problem have been proposed, including Bayesian methods that quantify uncertainty about preferences probabilistically [9, 4], and methods that simply pose constraints on the set of possible utility functions and refine these incrementally [21, 5, 18, 20, 6].

Constraint-based optimization (CBO) provides a natural framework for specifying and solving many decision problems. For example, configuration tasks [17] can naturally be viewed as reflecting a set of hard constraints (options available to a customer) and a utility function (reflecting customer preferences). While much work in the constraint-satisfaction literature has considered indirectly modeling preferences as hard constraints (with suitable relaxation techniques), more direct modeling of utility function has come to be recognized as both natural and computationally effective. Soft constraint frameworks [19, 3] that associate values with the satisfaction or violation of various constraints can be seen as implicitly reflecting a user utility function. Explicit separation of constraints from the utility model has also been proposed [7].

However, the requirement of complete utility information demanded by CBO is often problematic. For instance, users may have neither the ability nor the patience to provide full utility information to a system. Furthermore, in many if not most instances, an optimal decision (or some approximation thereof) can be determined with a very partial specification of the user’s utility function. As such, it is imperative that preference elicitation procedures be designed that focus on the relevant aspects of the problem. Preferences for unrealizable or infeasible outcomes are not (directly) relevant to decision making in a particular context; nor are precise preferences needed among outcomes that are provably dominated by others given the partial information at hand. Ultimately, it is the impact on decision quality that should guide elicitation effort [9, 4, 20, 6].

In recent work, we proposed a method for CBO in the presence of imprecise utility information [7]. Using the minimax regret decision criterion, we developed techniques to compute configurations that minimize the worst-case error given incomplete utility function information. While the imprecisely specified utility functions assumed there were of the type one would expect to arise during the course of incremental preference elicitation, the *actual process* of preference elicitation was not addressed. Indeed, very little attention has been paid to the problem of preference elicitation in the constraint-satisfaction literature. Only recently has the problem of elicitation of objective functions been given due attention [15].³

In this paper we address the problem of preference elicitation in CBO. We assume a set of (hard) constraints together with a graphical utility model [1, 5] capturing user preferences. While the structure of the utility model is known, the parameters of this utility model are imprecise, given by upper and lower bounds. Adopting the minimax regret model of [7], a robust decision can be made with respect to this utility uncertainty, by choosing the *minimax optimal configuration*. This is the solution the user would regret the least should an adversary choose a utility function consistent with our knowledge of the user’s preferences. If the parameters are loose enough, this regret may be very high, in which case, we would like to query the user for additional information about their utility function. In this work, we consider only *bound queries* (a local form of *standard gamble queries* [12])—that provide tighter upper or lower bounds on the utility parameters—since these are reasonably easy for users to assess and have been studied extensively in the decision analysis literature. We develop several new strategies for eliciting such information, strategies whose aim is to reduce the worst-case error (i.e., get guaranteed improvement in decision quality) with as few queries as possible.

Our first strategy, *halve largest gap* (HLG), provides the best theoretical guarantees—it works by providing uniform uncertainty reduction over the entire utility function. Our second strategy, *current solution* (CS), is more heuristic in nature, and focuses attention on *relevant* aspects of the utility function. Our empirical results show that this strategy works much better in practice, and does indeed distinguish relevant from irrelevant queries. Furthermore, its ability to discern good queries is also largely unaffected by approximation: the anytime nature of minimax computation allows time bounds to be used to ensure real-time response, yet elicitation effort remains about the same. We also introduce several additional strategies which capture some of the same intuitions

³ Related, but of a decidedly different character is work on constraint acquisition [14]; more closely tied is work on learning soft constraints [16].

as HLG and CS, but with different computational procedures (and complexity). Among these, the *optimistic-pessimistic (OP)* method works almost as well as CS, but with much lower computational demands.

The remainder of the paper is organized as follows. In Sec. 2 we briefly review constraint-based optimization with graphical utility models, define minimax regret, and describe our method of computing minimax regret for CBO problems. We also suggest several computational shortcuts well-suited to the interactive elicitation context. We describe our elicitation strategies in Sec. 3 and provide empirical comparisons of these strategies in Sec. 4. We conclude in Sec. 5 with a discussion of future research directions.

2 Constraint-based Optimization and Minimax Regret

We begin by describing the basic framework, graphical utility models, and the minimax regret decision criterion. We then provide a brief recap of the algorithm we use to compute the decision with minimax regret [7] and suggest some computational shortcuts motivated by interactive elicitation.

2.1 Optimization with Graphical Utility Models

We assume a finite set of attributes $\mathbf{X} = \{X_1, X_2, \dots, X_N\}$ with finite domains. An assignment $\mathbf{x} \in \text{Dom}(\mathbf{X})$ is often referred to as a *state*. For simplicity of presentation, we assume these attributes are boolean, but nothing important depends on this. We also have a set of hard constraints \mathcal{C} over these attributes. Each constraint \mathcal{C}_ℓ , $\ell = 1, \dots, L$, is defined over a set $\mathbf{X}[\ell] \subset \mathbf{X}$, and thus induces a set of legal configurations of attributes in $\mathbf{X}[\ell]$. We assume that the constraints \mathcal{C}_ℓ are represented in some logical form and can be expressed compactly: for example, we might write $X_1 \wedge X_2 \supset \neg X_3$ to denote the legal configurations of X_1, X_2, X_3 . We let $\text{Feas}(\mathbf{X})$ denote the subset of *feasible states* (i.e., assignments satisfying \mathcal{C}).

Suppose we have a known utility function $u : \text{Dom}(\mathbf{X}) \rightarrow \mathbf{R}$. Our aim is to find an optimal feasible state \mathbf{x}^* ; i.e., any

$$\mathbf{x}^* \in \arg \max_{\mathbf{x} \in \text{Feas}(\mathbf{X})} u(\mathbf{x}).$$

For this reason, we sometimes call feasible states *decisions*. While this problem can be formulated as an integer program (IP) and solved in any of a number of different ways, the problem specification is not compact: it requires exponentially many utility parameters $u(\mathbf{x})$ to be specified (one per state). In such *flat* utility functions, it is not generally possible to formulate the optimization concisely.

If some structure on the utility function is imposed, say, in the form of a *factored* graphical model, we are then generally able to reduce the number of variables to be linear in the number of parameters of the graphical model. We consider here the GAI (generalized additive independence) model [1] because of its generality (encompassing both linear models [13] and UCP-nets [5] as special cases).

Specifically, assume that our utility function can be written as the sum of K local utility functions, or *factors*, over small sets of variables:

$$u(\mathbf{x}) = \sum_{k \leq K} f^k(\mathbf{x}[k]). \quad (1)$$

Here each function f^k depends only on a local family of attributes $\mathbf{X}[k] \subset \mathbf{X}$. We denote by $\mathbf{x}[k]$ the restriction of state \mathbf{x} to the attributes in $\mathbf{X}[k]$. The problem of finding an optimal configuration can again be formulated as an IP, and can often be solved very effectively. For example, variable elimination [10], a well-known form of non-serial dynamic programming [2], can be applied to solve the problem in time exponential in the induced tree-width of the combined constraint/utility graph (which is often small in many structured problems).

2.2 Minimax Regret

Suppose the utility function for a CBO problem is unknown, but constraints on its parameters (e.g., in the form of bounds) are available. A very natural decision criterion in such a case is *minimax regret* [12, 5, 18, 20]: prefer the (feasible) assignment \mathbf{x} that obtains minimum max-regret, where max-regret is the largest quantity by which one could “regret” choosing action \mathbf{x} (while allowing the utility function to vary within the bounds).

More formally, let \mathcal{U} denote the set of feasible utility functions, reflecting our partial knowledge of the user’s preferences. The set \mathcal{U} may be finite; but more commonly it will be continuous, defined by bounds (or constraints) on (sets of) utility values $u(\mathbf{x})$ for various states. We refer to a pair $\langle \mathcal{C}, \mathcal{U} \rangle$, where \mathcal{C} is a set of configuration constraints, as an *imprecise CBO problem*.

The *pairwise regret* of state \mathbf{x} with respect to state \mathbf{x}' over feasible utility set \mathcal{U} is defined as

$$R(\mathbf{x}, \mathbf{x}', \mathcal{U}) = \max_{u \in \mathcal{U}} u(\mathbf{x}') - u(\mathbf{x}), \quad (2)$$

which is the most one could regret choosing \mathbf{x} instead of \mathbf{x}' (e.g., if an adversary could impose any utility function in \mathcal{U}). The *maximum regret* of decision \mathbf{x} is:

$$MR(\mathbf{x}, \mathcal{U}) = \max_{\mathbf{x}' \in \text{Feas}(\mathbf{X})} R(\mathbf{x}, \mathbf{x}', \mathcal{U}) \quad (3)$$

The *minimax regret* of feasible utility set \mathcal{U} is:

$$MMR(\mathcal{U}) = \min_{\mathbf{x} \in \text{Feas}(\mathbf{X})} MR(\mathbf{x}, \mathcal{U}) \quad (4)$$

If the only information we have about a user’s utility function is that it lies in the set \mathcal{U} , then a decision \mathbf{x}^* that minimizes max-regret—that is, an \mathbf{x}^* such that $MR(\mathbf{x}^*, \mathcal{U}) = MMR(\mathcal{U})$ —seems reasonable. Specifically, without distributional information over the set of possible utility functions, choosing (or recommending) a *minimax-optimal* decision \mathbf{x}^* minimizes the worst case loss with respect to possible realizations of the utility function $u \in \mathcal{U}$.

2.3 Computing Minimax Regret

Computation of minimax regret must be tackled with care if it is to be practical in problems with large numbers of variables and high-dimensional utility functions. Clearly, imprecise CBO problems pose potential difficulty for just these reasons. Fortunately, one can formulate the minimax regret optimization (Eq. 4) in a manner that exploits the graphical structure of the utility model, thereby admitting (in practice) computationally tractable solution [7].

Assume an imprecise CBO problem with factors f^k , $k \leq K$, defined over local families $\mathbf{X}[k]$, as described in Sec. 2.1. The parameters of this utility function are denoted by $u_{\mathbf{x}[k]} = f^k(\mathbf{x}[k])$, where $\mathbf{x}[k]$ ranges over $Dom(\mathbf{X}[k])$. We assume upper and lower bounds on each of these parameters, which we denote by $u_{\mathbf{x}[k]}^\uparrow$ and $u_{\mathbf{x}[k]}^\downarrow$, respectively. By defining $u(\mathbf{x})$ as in Eq. 1, pairwise regret, max regret and minimax regret are all defined in the same manner outlined in Sec. 2.2. Each of these quantities can be computed effectively by exploiting the graphical structure of the utility model [7], as we now detail.

It is straightforward to compute the pairwise regret of any pair of states \mathbf{x} and \mathbf{x}' . For each factor f^k and assignment pair $\mathbf{x}[k], \mathbf{x}'[k]$, we define the *local pairwise regret*: $r_{\mathbf{x}[k], \mathbf{x}'[k]} = u_{\mathbf{x}'[k]}^\uparrow - u_{\mathbf{x}[k]}^\downarrow$ when $\mathbf{x}[k] \neq \mathbf{x}'[k]$, and $r_{\mathbf{x}[k], \mathbf{x}'[k]} = 0$ when $\mathbf{x}[k] = \mathbf{x}'[k]$. With factored models, $R(\mathbf{x}, \mathbf{x}', \mathcal{U})$ is the sum of local pairwise regrets:

$$R(\mathbf{x}, \mathbf{x}', \mathcal{U}) = \sum_k r_{\mathbf{x}[k], \mathbf{x}'[k]}. \quad (5)$$

We can compute max regret $MR(\mathbf{x}, \mathcal{U})$ by substituting Eq. 5 into Eq. 3:

$$MR(\mathbf{x}, \mathcal{U}) = \max_{\mathbf{x}' \in Feas(\mathbf{X}')} \sum_k r_{\mathbf{x}[k], \mathbf{x}'[k]} \quad (6)$$

This can be recast as an IP as follows:

$$MR(\mathbf{x}, \mathcal{U}) = \max_{\{I_{\mathbf{x}'[k]}, X'_i\}} \sum_k \sum_{\mathbf{x}'[k]} r_{\mathbf{x}[k], \mathbf{x}'[k]} I_{\mathbf{x}'[k]} \quad \text{subject to } \mathcal{A}^4 \text{ and } \mathcal{C} \quad (7)$$

Here each X_i is a problem variable (in the original CBO problem). Boolean indicator variable $I_{\mathbf{x}'[k]}$ denotes that the adversary has chosen a state \mathbf{x}' (the solution to the IP) in which instantiation $\mathbf{x}'[k]$ holds in factor f^k (hence there is one such variable per utility parameter). \mathcal{A} is a set of constraints that ensures consistency among the $I_{\mathbf{x}'[k]}$ and X'_i variables (see [7] for further details).

With the ability to compute the max regret of a state \mathbf{x} effectively, we now turn to minimax regret computation. We can reformulate minimax regret $MMR(\mathcal{U})$ by substituting Eq. 6 into Eq. 4:

$$MMR(\mathcal{U}) = \min_{\mathbf{x} \in Feas(\mathbf{X})} \max_{\mathbf{x}' \in Feas(\mathbf{X}')} \sum_k r_{\mathbf{x}[k], \mathbf{x}'[k]} \quad (8)$$

which leads to the following mixed integer program (MIP) formulation:

$$\begin{aligned}
MMR(\mathcal{U}) &= \min_{\{I_{\mathbf{x}^{[k]}}, X_i\}} \max_{\mathbf{x}' \in Feas(\mathbf{X}')} \sum_k \sum_{\mathbf{x}^{[k]}} r_{\mathbf{x}^{[k]}, \mathbf{x}'^{[k]}} I_{\mathbf{x}^{[k]}} \quad \text{subject to } \mathcal{A} \text{ and } \mathcal{C} \quad (9) \\
&= \min_{\{I_{\mathbf{x}^{[k]}}, X_i, M\}} M \\
&\quad \text{subject to } \begin{cases} M \geq \sum_k \sum_{\mathbf{x}^{[k]}} r_{\mathbf{x}^{[k]}, \mathbf{x}'^{[k]}} I_{\mathbf{x}^{[k]}} & \forall \mathbf{x}' \in Feas(\mathbf{X}') \\ \mathcal{A} \text{ and } \mathcal{C} \end{cases} \quad (10)
\end{aligned}$$

In Eq. 9, we introduce the variables for the minimization, while in Eq. 10 we transform the minimax program into a min program. The new continuous variable M corresponds to the max regret of any state. This MIP has a number of $I_{\mathbf{x}^{[k]}}$ variables that is linear in the number of utility parameters. However, this MIP is not generally compact because Eq. 10 has one constraint per feasible state \mathbf{x}' . Nevertheless, we can get around the large number of constraints by *constraint generation*. This approach proceeds by repeatedly solving the MIP in Eq. 10, but using only a subset of the constraints on M associated with the feasible states \mathbf{x}' . At the first iteration, all constraints on M are ignored. At each iteration, we obtain a solution indicating some decision \mathbf{x} with purported minimax regret; however, since certain unexpressed constraints may be violated, we cannot be content with this solution. Thus, we look for the unexpressed constraint on M that is maximally violated by the current solution. This involves finding a *witness* \mathbf{x}' that maximizes regret w.r.t. the current solution \mathbf{x} ; that is, a decision \mathbf{x}' (and, implicitly, a utility function) that an adversary would choose to cause a user to regret \mathbf{x} the most.

Recall that finding the feasible \mathbf{x}' that maximizes $R(\mathbf{x}, \mathbf{x}', \mathcal{U})$ involves solving a single IP given by Eq. 7. We then impose the specific constraint associated with witness \mathbf{x}' and re-solve the MIP in Eq. 10 at the next iteration with this additional constraint. Clearly, if no constraint is violated at the current solution \mathbf{x} , then \mathbf{x} is the minimax-optimal configuration. The procedure is finite and guaranteed to arrive at the optimal solution. The constraint generation routine is not guaranteed to finish before it has the full set of constraints, but is relatively simple and in practice (as we will see) tends to generate a very small number of constraints. Thus we solve this very large MIP using a series of small MIPs, each with a linear number of variables and a set of active constraints that is also, typically, very small.

The above minimax regret procedure was tested on a real-estate problem of 47,775,744 configurations, a car-rental problem of 61,917,316,000 configurations and some synthetic problems of various sizes [7]. Despite the potentially large number of constraints (one per configuration), the optimal minimax configuration was found by generating only 7 constraints for both real-estate and car-rental problems in respectively 2 and 40 seconds. A number of other experiments verify the effectiveness of these techniques on problems of practical size [7].

In practice, since minimax regret will be computed between elicitation queries, it is critical that minimax regret be estimated in a relatively short period of time (say 5 seconds). Several improvements can be made to speed up minimax regret computation. For instance, it is often sufficient to find a feasible (instead of optimal) configuration \mathbf{x} for the MIP in Eq. 10 for each newly generated constraint. Intuitively, as long as the feasible \mathbf{x} allows us to find a violated constraint, the constraint generation continues to

Input: imprecise CBO problem, worst-case error tolerance τ .

1. Compute minimax regret mmr
2. Repeat until $mmr < \tau$
 - (a) Ask bound query q about some utility parameter $u(\mathbf{x}[k])$.
 - (b) If $u(\mathbf{x}[k]) \leq q$ then lower $u_{\mathbf{x}[k]}^{\uparrow}$ to q .
 - (c) Otherwise raise $u_{\mathbf{x}[k]}^{\downarrow}$ to q
 - (d) recompute mmr

Table 1. General form of the interactive elicitation procedure.

progress. Hence, instead of waiting a long time for an optimal \mathbf{x} , we can stop the MIP solver as soon as we find a feasible solution for which a violated constraint exists. Of course, at the last iteration, when there are no violated constraints, we have no choice but to wait for the optimal \mathbf{x} .

Minimax regret can be also be estimated more quickly—thus ensuring the real-time response needed for interactive optimization—by exploiting the anytime nature of the computation to simply stop early. Since minimax regret is computed incrementally by generating constraints, early stopping has the effect that some violated constraints may not have been generated. As a result the solution is a lower bound on minimax regret. We can also quickly compute an upper bound on minimax regret by computing the max regret of the \mathbf{x} found for the last MIP solved. These lower and upper bounds are often tight enough to provide elicitation guidance of similar quality to that obtained from computing minimax regret exactly.

Finally we observe that the minimax regret problem solved after receiving a response to one query is very similar to that solved before posing the query. As such, one can “seed” the minimax procedure invoked after a query with the constraints generated at the previous step. In this way, typically, only a few extra constraints are generated during each minimax computation. Given that the running time of minimax regret is dominated by constraint generation, this effectively amortizes the cost of minimax computation over a number of queries.

While we focus in this paper on the use of upper and lower bounds on utility parameters, the minimax procedures described here can be applied, with some modification, to problems in which arbitrary linear constraints are posed over utility parameters. The generalizations are more computationally difficult, but can be approximated feasibly. The ability to deal with such general constraints is important when dealing with specific types of queries (as we discuss below).

3 Elicitation Strategies

While the use of minimax regret provides a useful way of handling imprecise utility information, the initial bounds on utility parameters provided by users are unlikely to be tight enough to admit configurations with provably low regret. Instead, we imagine an interactive process in which the decision software queries the user for further information about her utility function—refining bounds on the parameters—until minimax

regret, given the current constraints, reaches an acceptable level τ .⁵ Table 1 summarizes the general form of the interactive elicitation procedure that we consider.

We first discuss the type of queries that we consider in this work, then describe a number of elicitation strategies. Throughout this section we assume some imprecise CBO problem $\langle \mathcal{C}, \mathcal{U} \rangle$ where \mathcal{U} is specified by graphical utility model $\{f^k : k \leq K\}$ with upper and lower bounds on its parameters.

3.1 Bound Queries

The types of queries we consider are *bound queries* in which we ask the user whether one of her utility parameters lies above a certain value. A positive response raises the lower bound on that parameter, while a negative response lowers the upper bound: in both cases, uncertainty is reduced.

While users often have difficulty assessing numerical parameters, they are typically better at comparing outcomes [13, 12]. However, a bound query can be viewed as a local form of a *standard gamble query (SGQ)*, commonly used in decision analysis; these in fact ask for comparisons. An SGQ for a specific state \mathbf{x} asks the user if she prefers \mathbf{x} to a gamble in which the best outcome \mathbf{x}_\top occurs with probability l and the worst \mathbf{x}_\perp occurs with probability $1 - l$ [13]. A positive response puts a lower bound on the utility of \mathbf{x} , and a negative response puts an upper bound. Calibration is attained by the use of common best and worst outcomes across all queries (and numerical assessment is restricted to evaluating probabilities).⁶

While we focus on bound queries, other forms of queries are quite natural. For example, comparison queries ask if one state \mathbf{x} is preferred to another \mathbf{x}' . Either response imposes a linear constraint on the parameters of the utility model that refines our assessment of possible utility functions. Since these constraints “tie” the assessment of various parameters (and do not just modify bounds), minimax regret computation would have to take the generalized form alluded to above.

Several of our query strategies rely on the following definitions.

Defn 1 Let $\langle \mathcal{C}, \mathcal{U} \rangle$ be an imprecise CBO problem. An *optimistic state* \mathbf{x}^o , a *pessimistic state* \mathbf{x}^p , and a *most uncertain state* \mathbf{x}^{mu} are any states satisfying:

$$\begin{aligned} \mathbf{x}^o &\in \arg \max_{\mathbf{x} \in \text{Feas}(\mathbf{X})} \max_{u \in \mathcal{U}} u(\mathbf{x}) \\ \mathbf{x}^p &\in \arg \max_{\mathbf{x} \in \text{Feas}(\mathbf{X})} \min_{u \in \mathcal{U}} u(\mathbf{x}) \\ \mathbf{x}^{mu} &\in \arg \max_{\mathbf{x} \in \text{Feas}(\mathbf{X})} \max_{u, u' \in \mathcal{U}} u(\mathbf{x}) - u'(\mathbf{x}) \end{aligned}$$

⁵ We could insist that regret reaches zero (i.e., that we have a provably optimal solution), or stop when regret reaches a point where further improvement is outweighed by the cost of additional interaction.

⁶ If the user is nearly indifferent to the two alternatives, they may be tempted to respond “I don’t know.” This can be handled by imposing a quantitative interpretation on “near indifference” and imposing a constraint that makes these two utilities “close.”

An optimistic state is a feasible state with the greatest upper bound on utility. A pessimistic state has the greatest lower bound on utility. A most uncertain state has the greatest difference between its upper and lower bounds. Each of these states can be computed in a single optimization by setting the parameters of the utility model to their upper bounds, their lower bounds, or their difference, and solving the corresponding (precise) CBO problem.

3.2 The Halve Largest Gap Strategy

The first query strategy we consider is the *halve largest gap (HLG)* strategy. It asks a query at the midpoint of the interval of the parameter $\mathbf{x}[k]$ with the largest gap between its upper and lower bounds. This is motivated by theoretical considerations, based on simple worst-case bounds on minimax regret. Define the *gap* of a utility parameter $u(\mathbf{x}[k])$, the *span* of factor f^k and *maxspan* of our utility model as follows:

$$\text{gap}(\mathbf{x}[k]) = u_{\mathbf{x}[k]\uparrow} - u_{\mathbf{x}[k]\downarrow} \quad (11)$$

$$\text{span}(f^k) = \max_{\mathbf{x}[k] \in \text{Dom}(\mathbf{X}[k])} \text{gap}(\mathbf{x}[k]) \quad (12)$$

$$\text{maxspan}(\mathcal{U}) = \sum_k \text{span}(f^k) \quad (13)$$

The quantity *maxspan* measures the largest gap between the upper and lower utility bound, regardless of feasibility. We can show that this quantity bounds minimax regret:

Proposition 1 *For any $\langle \mathcal{C}, \mathcal{U} \rangle$, $\text{MMR}(U) \leq \text{maxspan}(\mathcal{U})$.*

Since $\text{MMR}(U) \leq \text{MR}(\mathbf{x}^o, \mathcal{U})$ by definition and for any optimistic state \mathbf{x}^o we have $\text{MR}(\mathbf{x}^o, \mathcal{U}) \leq \text{maxspan}(\mathcal{U})$, the result follows.⁷

This suggests an obvious query strategy, the HLG method, in which a bound query is asked of the parameter p with the largest gap, at the midway point of its interval, $(p\uparrow - p\downarrow)/2$. This method ensures rapid reduction in max regret:

Proposition 2 *Let \mathcal{U} be an uncertain utility model with n parameters and let $m = \text{maxspan}(\mathcal{U})$. After $n \log(m/\varepsilon)$ queries in the HLG strategy, minimax regret is no greater than ε .*

In the worst case, no query strategy can reduce regret more quickly than HLG. Furthermore, there are classes of utility functions for which the bound is tight, so worst-case \mathcal{U} and configuration constraints \mathcal{C} exist that ensure regret will never be reduced to zero in finitely many queries.⁸

⁷ The definition of *maxspan* can be tightened in two ways. (a) One could account for logical consistency across utility factors (e.g., if X occurs in two factors, we cannot have a maximum utility span for a single state that instantiates the span in one factor with X true, and the span in the other with X false). Computing this tighter definition of span requires some minor optimization to find the logically consistent state with max span. (b) One could make this tighter still by restricting attention to feasible states (w.r.t. \mathcal{C}). The result still holds with these tighter definitions. However, the current definition requires no optimization to assess.

⁸ The bound is not, generally, tight if there is overlap in factors. The bound is tight if *maxspan* is defined to account for logical consistency.

3.3 The Current Solution Strategy

While HLG allows one to provide strong worst-case guarantees on regret improvement, it is “undirected” in that considerations of feasibility play no role in determining which queries to ask. An alternative strategy is to focus attention on parameters that participate in defining the max regret, namely, the minimax optimal \mathbf{x}^* and the adversarial witness \mathbf{x}^w for the current \mathcal{U} (recall that the witness maximizes the regret of \mathbf{x}^*). The *current solution (CS) query strategy* asks about the utility parameter in the set $\{\mathbf{x}^*[k] : k \leq K\} \cup \{\mathbf{x}^w[k] : k \leq K\}$ with largest $gap(\mathbf{x}[k])$ and queries the midpoint of the corresponding utility interval. Intuitively, should the answer to a query raise the lower bound on some $u(\mathbf{x}^*[k])$ or lower the upper bound on some $u(\mathbf{x}^w[k])$, then the pairwise regret $R(\mathbf{x}^*, \mathbf{x}^w)$ will be reduced, and usually minimax regret will be reduced as well. Of course, if the answer lowers the upper bound on some $u(\mathbf{x}^*[k])$ or raises the lower bound on some $u(\mathbf{x}^w[k])$, then pairwise regret $R(\mathbf{x}^*, \mathbf{x}^w)$ remains unchanged and minimax regret is not guaranteed to be reduced.

We have also experimented with a variant of the CS strategy in which regret is computed approximately to ensure fast interactive response in the querying process. This can be done by imposing a time bound on the solution algorithm for computing minimax regret, exploiting the anytime nature of the method described in Sec. 2.3. While we can’t be sure we have the minimax optimal solution with early termination, the solution may be good enough to guide the querying process. Furthermore, since we can compute the max regret of the anytime solution, we have an upper bound on minimax regret which can be used as a natural termination criterion.

3.4 Alternative Strategies

Finally, we consider several other strategies, which we describe briefly. The *optimistic query strategy* computes an optimistic state \mathbf{x}^o and queries (at the midpoint of the interval) the utility parameter in \mathbf{x}^o with the largest gap. Intuitively, an optimistic \mathbf{x}^o is a useful adversarial choice, so refining information about it can help reduce regret. The *pessimistic query strategy* is analogous, relying on the intuition that a pessimistic choice is useful in preventing the adversary from making us regret our decision too much. The *optimistic-pessimistic (OP) strategy* combines the two intuitions: it chooses the parameter with largest gap among both states. These strategies are computationally appealing since they require only standard CBO, not minimax optimization.⁹

The *most uncertain state (MUS) strategy* is a variant of HLG that accounts for feasibility: we compute a most uncertain state \mathbf{x}^{mu} and query (at the midpoint) the parameter in \mathbf{x}^{mu} with the largest gap. Finally, the *second-best (SB) strategy* is based on the following intuition: suppose we have the optimistic state \mathbf{x}^o and the second-best optimistic state \mathbf{x}^{2o} (i.e., that state with the second-highest upper bound—this is computable with a single optimization). If we could ask a query which reduced the upper bound utility of \mathbf{x}^o to lower than that of \mathbf{x}^{2o} , we ensure that regret is reduced (since the adversary can no longer attain this most optimistic value); if the lower bound of \mathbf{x}^o were raised

⁹ Even termination can be determined heuristically, for example, by computing the max regret of the optimistic state after each query, or doing minimax optimization after every k queries.

to the level of x^{2o} 's upper bound, then we could terminate—knowing that x^o is *optimal*. Thus we would like to query x^o at x^{2o} 's upper bound: a negative response will reduce regret, a positive response ensures x^o is optimal. Unfortunately, this cannot be implemented directly, since we can only query local parameters, but the strategy can be approximated for factored models by “distributing” this query across the different parameters and asking a set of queries.

The *myopically optimal (MY) strategy* computes the average regret reduction of the midpoint query for *each* utility parameter by solving the minimax optimization problem for each response to each query; it then asks the query with the largest regret reduction averaged over both possible answers. For large problems, this approach is computationally infeasible, but we test it on small problems to see how the other methods compare.¹⁰

4 Empirical Results

To test the effectiveness of the various query strategies, we ran a series of elicitation experiments on a variety of problems. For each problem we tested the following elicitation strategies: halve largest gap (HLG), current solution (CS), current solution with a computation-time bound of five seconds per query (CS-5), optimistic-pessimistic (OP), and most uncertain state (MUS). In addition, on problems small enough to permit it, we also tested these strategies against the much more computationally demanding myopically optimal method (MY).

We implemented the constraint generation approach outlined in Sec. 2.3 and used CPLEX as the generic IP solver. Our experiments considered two realistic domains—car rentals and real estate—as well as randomly generated synthetic problems. In each case we imposed a factored graphical structure to reduce the required number of utility parameters (upper and lower bounds). The graphical structure of the problems considered was sufficient to admit practical solution.

First, we experimented with a set of small synthetic problems. We did this to allow a comparison of all our proposed heuristics with the MY strategy which is quite demanding computationally. Figure 1 reports the average minimax regret over 45 small synthetic problems constructed by randomly setting the utility bounds and the variables on which each utility factor depends. Each problem has 10 attributes that can take at most 4 values and 10 factors that depend on at most 3 attributes. We simulate user responses by drawing a random utility function consistent with the bounds and using this to answer queries. Results are shown for two cases: utility parameters drawn from a uniform distribution over each interval, and those drawn from a (truncated) Gaussian centered at the midpoint of the interval (reflecting that a user is somewhat more likely to have a true parameter value nearer the middle of the range).¹¹ In both cases, we observe that the OP, CS and CS-5 heuristics offer elicitation strategies that decrease minimax-regret at a rate very close to MY. This suggests that OP, CS and CS-5 are computationally feasible, yet promising alternatives to the computationally prohibitive MY strategy.

¹⁰ By doing lookahead of k stages of this type, we could in fact compute the optimal query plan of k -steps; however, doing so is infeasible.

¹¹ All experiments show a reasonably small variance so we exclude error bars for legibility.

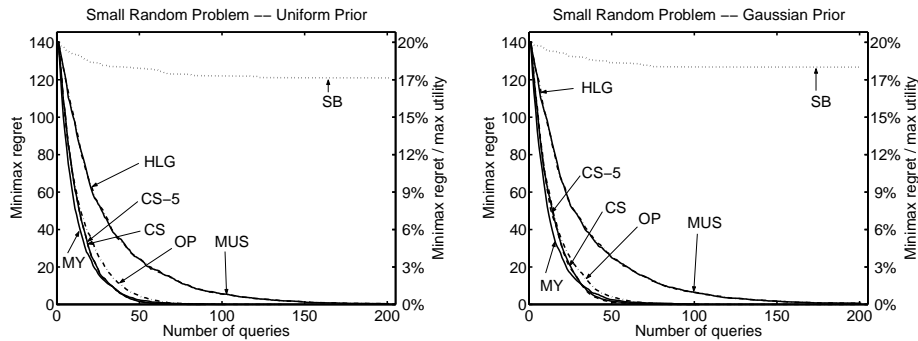


Fig. 1. Avg. max regret on small random problems (45 instances) as a function of number of queries given (a) uniform and (b) Gaussian distributed utilities.

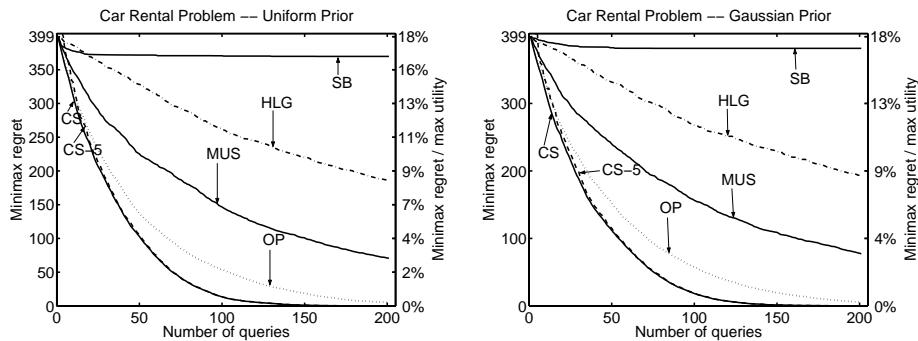


Fig. 2. Avg. max regret on car problem (45 instances) as a function of number of queries given (a) uniform and (b) Gaussian distributed utilities.

We report on further experiments using all strategies except MY with larger synthetic problems, the real-estate problem and the car rental problem. As above all results are averaged over 45 trials. The car-rental problem is modeled with 26 (multivalued) variables that specify various attributes of a car relevant to typical rental decisions, resulting in 61,917,360,000 possible configurations. The factored utility model consists of 36 local factors, each defined on at most five variables, with 150 utility parameters.¹² Performance of the various query strategies is depicted in Fig. 2, showing average minimax regret as a function of the number of queries. Initial utility bounds are set to give minimax regret of roughly 18% of the optimal solution. Both CS and CS-5 perform extremely well: regret is reduced to almost zero within 160 queries on average. Though this may seem like a lot of queries, recall that the problem is itself large and the utility model has 150 parameters. More importantly, these methods show excellent anytime performance: after only 80 queries, average minimax regret has dropped from 18% to under 2%. Interestingly, the time bound of 5 seconds imposed by CS-5, while leading to approximately minimax optimal solutions, does not affect query quality: the approximate solutions give rise to queries that are virtually as effective as the optimal solutions.

¹² See [7] for further details on all problem instances discussed here.

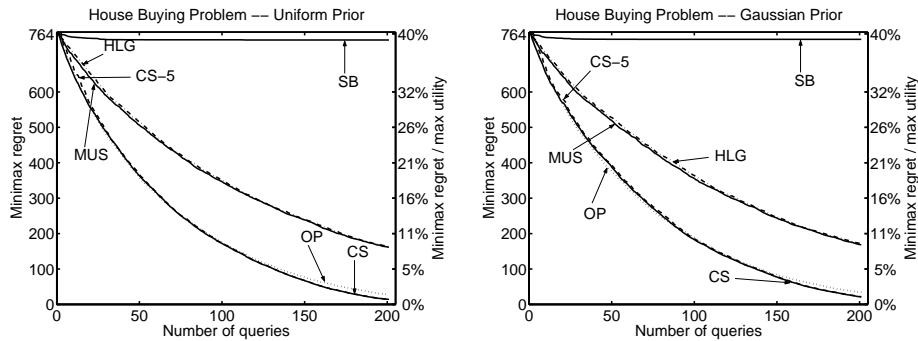


Fig. 3. Avg. max regret on real estate problem (45 instances) as a function of number of queries given (a) uniform and (b) Gaussian distributed utilities.

The CS strategy requires on average at most 83s per query. The OP strategy works very well too, and requires less computation time (0.1s per query) since it does not need to solve minimax problems (except to verify termination “periodically”, which is not reflected in query computation time). However, both OP and CS-5 are fast enough to be used interactively on problems of this size. MUS, HLG, and SB do not work nearly as well, with SB essentially stalling because of the slow progress made in reducing the upper bounds of the optimistic state.

The real-estate problem is modeled with 20 (multivalued) variables, with 47,775,744 possible configurations. The factored utility model consists of 29 local factors, giving rise to 100 utility parameters. Query performance is shown in Fig. 3, using the same regime as above. Again, both CS and CS-5 perform best, and the time bound of CS-5 has no effect on the quality of the CS strategy. Interestingly, OP performs almost identically to these, with somewhat lower computational cost (CS takes 14s/query, CS-5 5s, and OP 0.1s). Each of these methods reduces minimax regret from 40% of optimal to under 5% in about 120 queries. As above, SB fails to make progress, while HLG and MUS provide reasonable performance. Note that HLG requires no optimization nor any significant computation (except to verify termination). We also tested the query strategies on larger randomly generated problems (with 25 variables of domain size no more than four, and 20 utility factors with no more than three variables each). The same performance patterns as in the real-estate problem emerge, with CS, CS-5 and OP all performing much better than the others. Although OP performs slightly better than CS/CS-5, we do not believe that this is statistically significant since intervals of one standard deviation on their performance overlap.

5 Concluding Remarks

We have developed a number of query strategies for eliciting bounds on the parameters of graphical utility models for the purpose of solving imprecise constraint-based optimization problems. The most promising of these strategies, CS and OP, perform extremely well, requiring very few queries (relative to the model size) to provide dramatic reductions in regret. We have shown that using approximation of minimax regret

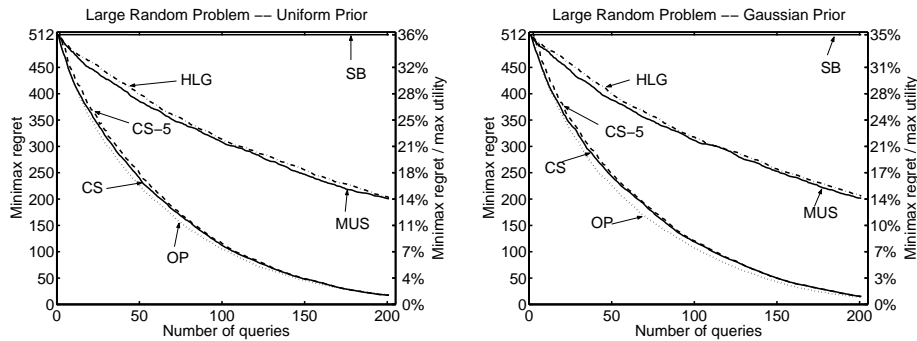


Fig. 4. Avg. max regret on large random problems (45 instances) as a function of number of queries given (a) uniform and (b) Gaussian distributed utilities.

reduces interactive computation time to levels required for real-time response without a noticeable effect on the performance of CS. OP also can be executed in real-time, since it does not require the same intensive minimax computation.

We hope to extend this research a number of directions. Naturally, we would like to consider additional query types, such as comparisons of outcomes and tackle the associated computational problems. New query strategies could also be devised. In practice, we often can assume or have access to distributional information over utilities. Rather than asking queries at midpoints of intervals, we could optimize the query point using probabilistic (value of information) computation, while using (distribution-free) regret to make decisions [20]. Optimal (non-myopic) strategies could be found by solving prohibitively large continuous MDPs. Nevertheless it would be interesting to explore non-myopic heuristics and to what extent lookahead information can improve the reduction in minimax regret.

References

1. Fahiem Bacchus and Adam Grove. Graphical models for preference and utility. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 3–10, Montreal, 1995.
2. Umberto Bertele and Francesco Brioschi. *Nonserial Dynamic Programming*. Academic Press, Orlando, 1972.
3. Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, 1997.
4. Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 239–246, Edmonton, 2002.
5. Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. UCP-Networks: A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 56–64, Seattle, 2001.
6. Craig Boutilier, Rajarshi Das, Jeffrey O. Kephart, Gerald Tesauro, and William E. Walsh. Cooperative negotiation in autonomic systems using incremental utility elicitation. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 89–97, Acapulco, 2003.

7. Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization with the minimax decision criterion. In *Ninth International Conference on Principles and Practice of Constraint Programming*, pages 168–182, Kinsale, Ireland, 2003.
8. U. Chajewska, L. Getoor, J. Norman, and Y. Shahar. Utility elicitation as a classification problem. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 79–88, Madison, WI, 1998.
9. Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 363–369, Austin, TX, 2000.
10. Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 211–219, Portland, OR, 1996.
11. J. S. Dyer. Interactive goal programming. *Management Science*, 19:62–70, 1972.
12. Simon French. *Decision Theory*. Halsted Press, New York, 1986.
13. Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, New York, 1976.
14. Barry O’Sullivan, Eugene Freuder, and Sarah O’Connell. Interactive constraint acquisition. In *CP-2001 Workshop on User Interaction in Constraint Processing*, Paphos, Cyprus, 2001.
15. Pearl Pu, Boi Faltings, and Marc Torrens. User-involved preference elicitation. In *IJCAI-03 Workshop on Configuration*, Acapulco, 2003.
16. F. Rossi, A. Sperduti, K. B. Venable, L. Khatib, P. Morris, and R. Morris. Learning and solving soft temporal constraints: An experimental study. In *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming*, pages ???–???, Ithaca, NY, 2002.
17. D. Sabin and R. Weigel. Product configuration frameworks—a survey. *IEEE Intelligent Systems and their Applications*, 13(4):42–49, 1998.
18. Ahti Salo and Raimo P. Hämmäläinen. Preference ratios in multiattribute evaluation (PRIME)—elicitation and decision procedures under incomplete information. *IEEE Trans. on Systems, Man and Cybernetics*, 31(6):533–545, 2001.
19. Thomas Schiex, Helene Fargie, and Gérard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 631–637, Montreal, 1995.
20. Tianhan Wang and Craig Boutilier. Incremental utility elicitation with the minimax regret decision criterion. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 309–316, Acapulco, 2003.
21. Chelsea C. White, III, Andrew P. Sage, and Shigeru Dozono. A model of multiattribute decisionmaking and trade-off weight determination under uncertainty. *IEEE Transactions on Systems, Man and Cybernetics*, 14(2):223–229, 1984.