

Minimax Regret Based Elicitation of Generalized Additive Utilities

Darius Braziunas

Department of Computer Science
University of Toronto
Toronto, ON M5S 3G4
darius@cs.toronto.edu

Craig Boutilier

Department of Computer Science
University of Toronto
Toronto, ON M5S 3G4
cebly@cs.toronto.edu

Abstract

We describe the semantic foundations for elicitation of generalized additively independent (GAI) utilities using the minimax regret criterion, and propose several new query types and strategies for this purpose. Computational feasibility is obtained by exploiting the local GAI structure in the model. Our results provide a practical approach for implementing preference-based constrained configuration optimization as well as effective search in multiattribute product databases.

Introduction

Representing, reasoning with, and eliciting the preferences of individual users is a fundamental problem in the design of decision support tools (and indeed, much of AI). A key issue in preference research is dealing with large, multiattribute problems: preference representation and elicitation techniques must cope with the exponential size of the outcome space. By exploiting intrinsic independencies in preference structure, *factored* utility models can provide tractable (although sometimes approximate) solutions to the problem. Utility function structure (such as additive, multilinear, generalized additive, etc.) can be used to represent large utility models very concisely (Keeney and Raiffa 1976). While additive models are the most widely used in practice, *generalized additive independence models* (GAI) have recently generated interest because of their greater flexibility and applicability (Bacchus and Grove 1995; Boutilier, Bacchus, and Brafman 2001; Gonzales and Perny 2004; Boutilier, et al. 2006; Braziunas and Boutilier 2005). Even though the semantic foundations of the GAI representation were described by Fishburn decades ago (Fishburn 1967), the design of effective elicitation techniques has gained attention only recently (Boutilier, et al. 2006; Gonzales and Perny 2004; Braziunas and Boutilier 2005).

In this paper, we develop a new model for utility elicitation in GAI models based on the minimax regret decision criterion (Savage 1954; Kouvelis and Yu 1997). Minimax regret provides a robust way of making decisions under utility function uncertainty, minimizing worst-case loss under all possible realizations of a user’s utility function (Boutilier, Bacchus, and Brafman 2001; Salo and Ramalainen 2001;

Boutilier, et al. 2006); as such it is applicable when distributional information over utility functions is not easily available. Regret has also proven to be an effective driver of preference elicitation (Wang and Boutilier 2003; Boutilier, et al. 2006; Boutilier, Sandholm, and Shields 2004). However, prior work on regret-based elicitation for GAI models has ignored key semantic issues, thus simplifying the approach to both elicitation and regret computation and optimization. By building on the semantic foundations of GAI elicitation laid out in (Braziunas and Boutilier 2005), we identify new classes of elicitation queries suitable for regret-based elicitation, and propose several new query strategies based on these classes.

Our approach emphasizes *local queries* over small sets of attributes; but *global queries* over full outcomes are required to calibrate certain terms across GAI factors (a problem ignored in previous work on regret-based elicitation). However, we will demonstrate that most of the “heavy lifting” can be achieved using local queries. Our new approach guarantees the semantic soundness of the utility representation in a way that techniques that ignore interactions across factors do not. In addition, these queries put much more intricate constraints on GAI model parameters than those considered in previous work. For this reason, we develop new formulations of the mixed integer programs required to handle these queries in regret-based optimization and show that the problem can be effectively solved despite the added complexity.

We begin in the Multiattribute Preferences section with relevant background on multiattribute utility. We discuss appropriate forms of both local and global queries for GAI elicitation in the Elicitation Queries section. We then describe effective integer programming formulations for minimax regret computation in the Minimax Regret Calculation section, followed by a discussion of regret computation in multiattribute product databases in the Multiattributed Product Databases section. The Elicitation strategies section presents several elicitation strategies based on the query types above, which are empirically evaluated in the Experimental Results section. Future directions are summarized in the Conclusions.

Multiattribute preferences

Assume a set of attributes X_1, X_2, \dots, X_n , each with finite domains, which define a set of *outcomes* $\mathbf{X} = X_1 \times \dots \times X_n$.

The preferences of a user, on whose behalf decisions are made, are captured by a *utility function* $u : \mathbf{X} \mapsto \mathbb{R}$. A utility function can be viewed as reflecting (qualitative) preferences over *lotteries* (distributions over outcomes) (Keeney and Raiffa 1976), with one lottery preferred to another iff its expected utility is greater. Let $\langle p, \mathbf{x}^\top; 1 - p, \mathbf{x}^\perp \rangle$ denote the lottery where the best outcome \mathbf{x}^\top is realized with probability p , and the worst outcome \mathbf{x}^\perp with probability $1 - p$; we refer to best and worst outcomes as *anchor* outcomes. Since utility functions are unique up to positive affine transformations, it is customary to set the utility of the best outcome \mathbf{x}^\top to 1, and the utility of the worst outcome \mathbf{x}^\perp to 0. If a user is indifferent between some outcome \mathbf{x} and the *standard gamble* $\langle p, \mathbf{x}^\top; 1 - p, \mathbf{x}^\perp \rangle$, then $u(\mathbf{x}) = p$.

Additive utilities

Since the size of outcome space is exponential in the number of attributes, specifying the utility of each outcome is infeasible in many practical applications. Most preferences, however, exhibit internal structure that can be used to express u concisely. *Additive independence* (Keeney and Raiffa 1976) is commonly assumed in practice, where u can be written as a sum of single-attribute *subutility functions*:¹

$$u(\mathbf{x}) = \sum_{i=1}^n u_i(x_i) = \sum_{i=1}^n \lambda_i v_i(x_i).$$

The subutility functions $u_i(x_i) = \lambda_i v_i(x_i)$ can be defined as a product of *local value functions* (LVFs) v_i and scaling constants λ_i . This simple factorization allows us to separate the representation of preferences into two components: “local” and “global.” Significantly, LVFs can be defined using “local” lotteries that involve only a single attribute: $v_i(x_i) = p$, where p is the probability at which the user is indifferent between two local outcomes x_i and $\langle p, x_i^\top; 1 - p, x_i^\perp \rangle$, *ceteris paribus*.² Since we can define value functions independently of other attributes, we can also *assess* them independently using queries only about values of attribute i . This focus on preferences over individual attributes has tremendous practical significance, because people have difficulty taking into account more than five or six attributes at a time (Green and Srinivasan 1978).

The scaling constants λ_i are “global” and are required to properly calibrate LVFs across attributes. To define the scaling constants, we first introduce a notion of a *reference* (or *default*) outcome, denoted by $\mathbf{x}^0 = (x_1^0, x_2^0, \dots, x_n^0)$. The reference outcome is an arbitrary outcome, though it is common to choose the worst outcome \mathbf{x}^\perp as \mathbf{x}^0 (more generally, any *salient* outcome, such as an “incumbent” will prove useful in this role). Let $\mathbf{x}^{\top j}$ be a full outcome where the j^{th} attribute is set to its best level whereas other attributes are fixed at their reference levels; $\mathbf{x}^{\perp j}$ is defined similarly. Then, $\lambda_j = u(\mathbf{x}^{\top j}) - u(\mathbf{x}^{\perp j})$. To assess scaling constants λ_j , one must ask queries involving global outcomes $\mathbf{x}^{\top j}$ and

¹This decomposition is possible if a user is indifferent between lotteries with the same marginals on each attribute.

² x_i^\top and x_i^\perp are the best and worst levels of attribute i . Without loss of generality, we assume $v_i(x_i^\top) = 1$, $v_i(x_i^\perp) = 0$.

$\mathbf{x}^{\perp j}$ for each attribute j . Only $2n$ such queries are needed, and they involve varying only one feature at a time from the reference outcome. This ease of assessment makes additive utility the model of choice in most practical applications.

Generalized additive utilities

Simple additive models, although very popular in practice, are quite restrictive in their assumptions of attribute independence. A more general utility decomposition, based on *generalized additive independence* (GAI), has recently gained more attention because of its additional flexibility (Bacchus and Grove 1995; Boutilier, Bacchus, and Brafman 2001; Gonzales and Perny 2004; Boutilier, et al. 2006; Braziunas and Boutilier 2005). It can model “flat” utility functions with no internal structure as well as linear additive models. Most realistic problems arguably fall somewhere between these two extremes.

GAI models (Fishburn 1967; Bacchus and Grove 1995) additively decompose a utility function over (possibly overlapping) *subsets* of attributes. Formally, assume a given collection $\{I_1, \dots, I_m\}$ of possibly intersecting attribute (index) sets, or *factors*. Given an index set $I \subseteq \{1, \dots, n\}$, we define $\mathbf{X}_I = \times_{i \in I} X_i$ to be the set of *partial outcomes* restricted to attributes in I . For a factor j , \mathbf{x}_{I_j} , or simply \mathbf{x}_j , is a particular instantiation of attributes in factor j . The factors are *generalized additively independent* if and only if the user is indifferent between any two lotteries with the same marginals on each set of attributes (Fishburn 1967). Furthermore, if GAI holds, the utility function can be written as a sum of *subutility* functions (Fishburn 1967):

$$u(\mathbf{x}) = u_1(\mathbf{x}_{I_1}) + \dots + u_m(\mathbf{x}_{I_m}).$$

As with additive utilities, in order to take full advantage of utility structure and facilitate elicitation, we would like to decompose GAI subutilities into “global” and “local” components. With GAI utilities, the matter is less straightforward, because the values of subutility functions u_j do not directly represent the local preference relation among the attributes in factor j . Intuitively, since utility can “flow” from one subutility factor to the next through the shared attributes, the subutility values do not have an independent semantic meaning.³ Nevertheless, it turns out that separation into local and global components is possible even with GAI utilities (Braziunas and Boutilier 2005).

First, we define a generalization of *local value function* (LVF) v_j . Let the *conditioning set* C_j of factor j be the set of all attributes that share GAI factors with attributes in j . Intuitively, fixing the attributes in the conditioning set to any value “blocks” the influence of other factors on factor j . In a manner similar to additive models, the local value $v_j(\mathbf{x}_j)$ of suboutcome \mathbf{x}_j is simply p , the probability that induces indifference in the *local* standard gamble,

³Indeed, a utility function can be faithfully decomposed in two different ways into the *same* set of factors such that the local (qualitative) preferences for two outcomes in an LVF are *reversed* in the two decompositions (Gonzales and Perny 2004; Braziunas and Boutilier 2005). This means purely local elicitation without calibration across factors is impossible without risking severe misrepresentation of user preferences.

$\mathbf{x}_j \sim \langle p, \mathbf{x}_j^\top; 1 - p, \mathbf{x}_j^\perp \rangle$, given that attributes in the conditioning set C_j are fixed at reference levels, *ceteris paribus*. We refer to the setting of attributes in C_j to their reference values as the *local value condition*. Here \mathbf{x}_j^\top and \mathbf{x}_j^\perp are the best and worst suboutcomes in factor j assuming the local value condition; as before, we also normalize the LVF by setting $v_j(\mathbf{x}_j^\top) = 1$ and $v_j(\mathbf{x}_j^\perp) = 0$. We see, then, that LVFs can be *elicited* using queries that involve only attributes in single factors and their (usually small) conditioning sets. We discuss specific elicitation methods below.

The canonical way to define subutilities (see (Fishburn 1967)) is:

$$u(\mathbf{x}) = \sum_{j=1}^m u_j(\mathbf{x}_j) = \sum_{j=1}^m \lambda_j \bar{u}_j(\mathbf{x}_j), \quad (1)$$

where λ_j is a scaling constant, and

$$\bar{u}_j(\mathbf{x}_j) = v_j(\mathbf{x}_j) + \sum_{k=1}^{j-1} (-1)^k \sum_{1 \leq i_1 < \dots < i_k < j} v_j(\mathbf{x}_j[\bigcap_{s=1}^k I_{i_s} \cap I_j]) \quad (2)$$

is an *unscaled* subutility function.⁴ For any \mathbf{x} , $\mathbf{x}[I]$ is an outcome where attributes not in I are set to the reference value (i.e., $X_i = x_i$ if $i \in I$, and $X_i = x_i^0$ if $i \notin I$).

To compute the unscaled subutility function $\bar{u}_j(\mathbf{x}_j)$, we must know which local suboutcomes are involved in the right-hand side of Eq. 2; this amounts to finding all nonempty sets $\bigcap_{s=1}^k I_{i_s} \cap I_j$ and recording the sign (+/-) for the corresponding LVFs. The structure of subutility functions depends only on the decomposition of GAI sets. Therefore, given a GAI graph, a graphical search procedure (Braziunas and Boutilier 2005) computes the relevant subsets for Eq. 2 efficiently. We do this once for each factor.

To simplify Eq. 2, we introduce the following notation. Let N_j be the number of local configurations in factor j (e.g., with 3 boolean attributes, $N_j = 8$). The local value function v_j can be expressed by N_j parameters $v_j^1, \dots, v_j^{N_j}$, with $v_j^i = v_j(\mathbf{x}_j)$ where \mathbf{x}_j is the i^{th} local configuration of factor j . Then, Eq. 2 can be rewritten as

$$\bar{u}_j(\mathbf{x}_j) = \sum_{i \in 1..N_j} C_{\mathbf{x}_j}^i v_j^i, \quad (3)$$

where the $C_{\mathbf{x}_j}^i$ are integer coefficients precomputed by the graphical search procedure (most of these are zero).

The scaling constants λ_j are defined in a way analogous to simple additive utility case. Let $\mathbf{x}^{\top j}$ and $\mathbf{x}^{\perp j}$ be the best and the worst (full) outcomes, given that attributes not in factor j are set to their reference levels. Then,

$$\lambda_j = u(\mathbf{x}^{\top j}) - u(\mathbf{x}^{\perp j}) = u_j^\top - u_j^\perp.$$

Thus, a GAI model is additively decomposed into factors which are the product of scaling constants, or *weights*, λ_j and a linear combination of LVF parameters:

$$u(\mathbf{x}) = \sum_j \lambda_j \bar{u}_j(\mathbf{x}_j) = \sum_j \left[(u_j^\top - u_j^\perp) \sum_{i \in 1..N_j} C_{\mathbf{x}_j}^i v_j^i \right] \quad (4)$$

⁴In Eq. 2, the sum is only over non-empty intersections.

As in the additive case, we need to elicit utilities of a few *full* outcomes to achieve the right calibration of the LVFs. For each factor we must know the utility of the best and the worst outcomes given that attributes in *other* factors are set to their reference levels.

Elicitation queries

In general, eliciting complete preference information is costly and, in most cases, unnecessary for making an optimal decision. Instead, elicitation and decision making can be viewed as a single process whose goal is to achieve the right tradeoff between the costs of interaction, potential improvements of decision quality due to additional preference information, and the value of a terminal decision (Boutlier 2002).

The types of queries one considers is an integral part of the preference elicitation problem. Some queries are easy to answer, but do not provide much information, while more informative queries are often costly. Computing or estimating the value of information can vary considerably for different query types. Finally, allowable queries define the nature of constraints on the feasible utility set. We broadly distinguish *global queries* over full outcomes from *local queries* over subsets of outcomes. In most multiattribute problems, people can meaningfully compare outcomes with no more than five or six attributes (Green and Srinivasan 1978). Therefore, most of the global queries have local counterparts that apply to a subset of attributes.

We will consider four types of queries for elicitation. The following queries are well-defined semantically, relatively simple, and easy to explain to non-expert users.

Local bound queries An LVF calibrates utilities of local outcomes with respect to local “anchor” outcomes \mathbf{x}_j^\top and \mathbf{x}_j^\perp , given that the attributes in conditioning set C_j are fixed at their reference levels. A *local bound query* on parameter v_j^x asks: “Assume that the attributes in C_j are fixed at reference levels. Would you prefer the local outcome \mathbf{x}_j to a lottery $\langle \mathbf{x}_j^\top, p; \mathbf{x}_j^\perp, 1 - p \rangle$, assuming that the remaining attributes are fixed at same levels (*ceteris paribus*)?” If the answer is “yes”, $v_j^x \geq p$; if “no”, then $v_j^x < p$. By definition, the local value parameters v_j^x lie in $[0, 1]$. This binary (yes/no) query differs from a (local) standard gamble since we do not ask the user to *choose* the indifference level p , only bound it.

Local comparison queries Local comparisons are natural and easy to answer. We ask a user to compare two local outcomes: “Assume that the attributes in C_j are fixed at reference levels. Would you prefer local outcome \mathbf{x}_j to local outcome \mathbf{x}_j' *ceteris paribus*?” If the answer is “yes”, $v_j^x \geq v_j^{x'}$; if “no”, then $v_j^x < v_j^{x'}$.

Anchor bound queries The scaling factor, or weight, for a subutility function \bar{u}_j is $(u_j^\top - u_j^\perp)$, where u_j^\top is the global utility of the outcome in which the j^{th} factor is set to its best

value, and all the other attributes are fixed at reference levels. Similarly, u_j^\perp is the utility of the “bottom anchor” of factor j . Utilities of anchor levels $u_1^\top, u_1^\perp, \dots, u_m^\top, u_m^\perp$ must be obtained using global queries. However, we need only ask $2m$ direct utility queries over full outcomes; this is the *same number of global queries* required for scaling in the additive case (considering each attribute as a factor).

Instead of eliciting exact anchor utilities directly, we propose global queries that are easier to answer. An *anchor bound query* asks: “Consider a full outcome $\mathbf{x}^{\top j}$, where attributes in factor j are set to their best values, and other attributes are fixed at reference levels. Do you prefer $\mathbf{x}^{\top j}$ to a lottery $\langle \mathbf{x}^\top, p; \mathbf{x}^\perp, 1-p \rangle$?” A “yes” response gives $u_j^\top \geq p$; and “no”, $u_j^\top < p$ (assuming, without loss of generality, that $u(\mathbf{x}^\top) = 1$ and $u(\mathbf{x}^\perp) = 0$). An analogous query exists for the “bottom” anchor $\mathbf{x}^{\perp j}$.

Anchor comparison queries We can also ask a user to compare anchor outcomes from different factors: “Do you prefer global outcome $\mathbf{x}^{\top k}$ to $\mathbf{x}^{\perp l}$ ”? If “yes”, then $u_k^\top \geq u_l^\perp$; if “no”, then $u_k^\top < u_l^\perp$. Such comparison queries are usually much easier to answer than bound queries.

Minimax regret calculation

Minimax regret is a natural decision criterion for making decisions with incompletely or imprecisely specified utility functions (Boutilier, Bacchus, and Brafman 2001; Salo and Ramalainen 2001; Boutilier, et al. 2006). It requires that our system recommend a (feasible) outcome \mathbf{x}^* that minimizes *maximum regret* with respect to all possible realizations of the user’s utility function. This guarantees worst-case bounds on the quality of the decision made under the type of strict uncertainty induced by the queries above (Wang and Boutilier 2003; Boutilier, et al. 2006; Boutilier, Sandholm, and Shields 2004).

Let \mathbf{U} be the set of feasible utility functions, defined by constraints on the values of factor anchors u_j^\top, u_j^\perp (for each factor j), and constraints on the LVF parameters v_j^i . Such constraints, denoted \mathcal{U} , are induced by responses to the queries described above. Let $Feas(\mathbf{X}) \subseteq \mathbf{X}$ be the set of *feasible* outcomes (e.g., defined by a set of hard constraints \mathcal{C}). We define minimax regret in three stages (following (Boutilier, et al. 2006)). The *pairwise regret* of choosing \mathbf{x} instead of \mathbf{x}' w.r.t. \mathbf{U} is $R(\mathbf{x}, \mathbf{x}', \mathbf{U}) = \max_{u \in \mathbf{U}} u(\mathbf{x}') - u(\mathbf{x})$. The *maximum regret* of choosing outcome \mathbf{x} is $MR(\mathbf{x}, \mathbf{U}) = \max_{\mathbf{x}' \in Feas(\mathbf{X})} R(\mathbf{x}, \mathbf{x}', \mathbf{U})$. Finally, the outcome that minimizes max regret is the *minimax optimal decision*: $MMR(\mathbf{U}) = \min_{\mathbf{x} \in Feas(\mathbf{X})} MR(\mathbf{x}, \mathbf{U})$. We develop tractable formulations of these definitions for GAI models.

Pairwise regret Given a GAI model, the *pairwise regret* of \mathbf{x} w.r.t. \mathbf{x}' over \mathbf{U} is:

$$\begin{aligned} R(\mathbf{x}, \mathbf{x}', \mathbf{U}) &= \max_{u \in \mathbf{U}} u(\mathbf{x}') - u(\mathbf{x}) \\ &= \max_{u \in \mathbf{U}} \sum_j [u_j(\mathbf{x}'_j) - u_j(\mathbf{x}_j)] \\ &= \max_{\{u_j^\top, u_j^\perp, v_j^i\}} \sum_j (u_j^\top - u_j^\perp) (\bar{u}_j(\mathbf{x}'_j) - \bar{u}_j(\mathbf{x}_j)) \\ &= \max_{\{u_j^\top, u_j^\perp, v_j^i\}} \sum_j \left[(u_j^\top - u_j^\perp) \sum_{i \in 1..N_j} (C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i) v_j^i \right]. \end{aligned} \quad (5)$$

In general, when constraints on utility space tie together parameters from different factors, regret computation has a quadratic objective. Such constraints might arise, for example, from global comparison queries. With linear constraints, this becomes a quadratic program.

Since factors reflect intrinsic independencies among attributes, it is natural to assume that utility constraints involve only parameters *within the same factor*. The constraints induced by *local* comparison or bound queries, for instance, have this form. We call constraints involving parameters within a single factor *local*. This allows modeling regret computation linearly as we discuss below.

If the constraints on local value parameters v_j^i are local then Eq. 5 can be simplified by pushing one “max” inward. This is made possible by the fact that the scaling factors $u_j^\top - u_j^\perp$ are *always positive*: $R(\mathbf{x}, \mathbf{x}', \mathbf{U}) =$

$$\begin{aligned} &= \max_{\{u_j^\top, u_j^\perp, v_j^i\}} \sum_j \left[(u_j^\top - u_j^\perp) \sum_{i \in 1..N_j} (C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i) v_j^i \right] \\ &= \max_{\{u_j^\top, u_j^\perp\}} \sum_j \left[(u_j^\top - u_j^\perp) \max_{\{v_j^i\}} \sum_{i \in 1..N_j} (C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i) v_j^i \right] \\ &= \max_{\{u_j^\top, u_j^\perp\}} \sum_j (u_j^\top - u_j^\perp) \bar{r}_j^{\mathbf{x}, \mathbf{x}'}, \end{aligned} \quad (6)$$

where (unscaled) “local regret”

$$\bar{r}_j^{\mathbf{x}, \mathbf{x}'} = \max_{\{v_j^i\}} \sum_{i \in 1..N_j} (C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i) v_j^i \quad (7)$$

can be precomputed by solving a small linear program (whose size is bounded by the factor size).

If constraints on LVF parameters are bound constraints, and therefore independent of each other, we can do without linear programming when computing the local regret $\bar{r}_j^{\mathbf{x}, \mathbf{x}'}$ (by pushing the max within the sum):

$$\bar{r}_j^{\mathbf{x}, \mathbf{x}'} = \sum_{i \in 1..N_j} \max_{\{v_j^i\}} (C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i) v_j^i,$$

where $\max_{\{v_j^i\}} (C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i) v_j^i =$

$$\begin{cases} (C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i) \max(v_j^i), & \text{if } C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i \geq 0, \\ (C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i) \min(v_j^i), & \text{if } C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i < 0. \end{cases}$$

Maximum regret The *max regret* of choosing \mathbf{x} is $MR(\mathbf{x}, \mathbf{U})$:

$$\begin{aligned} &= \max_{\mathbf{x}' \in Feas(\mathbf{X})} R(\mathbf{x}, \mathbf{x}', \mathbf{U}) \\ &= \max_{\mathbf{x}' \in Feas(\mathbf{X}), \mathbf{u} \in \mathbf{U}} u(\mathbf{x}') - u(\mathbf{x}) \\ &= \max_{\mathbf{x}' \in Feas(\mathbf{X}), \{u_j^\top, u_j^\perp, v_j^i\}} \sum_j \left[(u_j^\top - u_j^\perp) \sum_{i=1..N_j} (C_{\mathbf{x}'_j}^i - C_{\mathbf{x}_j}^i) v_j^i \right] \end{aligned} \quad (8)$$

If local value constraints involve only local value parameters within their own factors, the max regret expression above simplifies to:

$$MR(\mathbf{x}, \mathbf{U}) = \max_{\mathbf{x}' \in Feas(\mathbf{X}), \{u_j^\top, u_j^\perp\}} \sum_j (u_j^\top - u_j^\perp) \bar{r}_j^{\mathbf{x}, \mathbf{x}'}, \quad (9)$$

where local regrets $\bar{r}_j^{\mathbf{x}, \mathbf{x}'}$ can be precomputed beforehand and treated as constants. This optimization can be recast as a *mixed integer program* (MIP):

$$\begin{aligned} MR(\mathbf{x}, \mathbf{U}) &= \max_{\mathbf{x}' \in Feas(\mathbf{X})} \max_{\{u_j^\top, u_j^\perp\}} \sum_j (u_j^\top - u_j^\perp) \bar{r}_j^{\mathbf{x}, \mathbf{x}'} \\ &= \max_{\{I_j^{\mathbf{x}'}, u_j^\top, u_j^\perp\}} \sum_j \sum_{\mathbf{x}'_j} (u_j^\top - u_j^\perp) I_j^{\mathbf{x}'} \bar{r}_j^{\mathbf{x}, \mathbf{x}'}, \\ &\text{subject to constraints } \mathcal{A}, \mathcal{C} \text{ and } \mathcal{U}, \end{aligned}$$

where \mathcal{A} are state definition constraints tying indicators $I_j^{\mathbf{x}'}$ with consistent attribute assignments and \mathcal{C} are the domain constraints defining feasible configurations.

Using the ‘‘big-M’’ transformation, the quadratic optimization above can be linearized by introducing variables $Y_j^{\mathbf{x}'}$:

$$\begin{aligned} MR(\mathbf{x}, \mathbf{U}) &= \max_{\{Y_j^{\mathbf{x}'}, u_j^\top, u_j^\perp\}} \sum_j \sum_{\mathbf{x}'_j} \bar{r}_j^{\mathbf{x}, \mathbf{x}'} Y_j^{\mathbf{x}'}, \\ &\text{subject to } \begin{cases} 0 \leq Y_j^{\mathbf{x}'} \leq M_j I_j^{\mathbf{x}'}, \forall j, \mathbf{x}'_j \\ Y_j^{\mathbf{x}'} \leq (u_j^\top - u_j^\perp), \forall j, \mathbf{x}'_j \\ \mathcal{A}, \mathcal{C} \text{ and } \mathcal{U}. \end{cases} \end{aligned} \quad (10)$$

In the formulation above, the first constraint ensures that $Y_j^{\mathbf{x}'}$ is zero whenever $I_j^{\mathbf{x}'}$ is zero. If $I_j^{\mathbf{x}'}$ is one, $Y_j^{\mathbf{x}'}$ is bounded by some constant $M_j \geq (u_j^\top - u_j^\perp)$, and the second constraint ensures that $Y_j^{\mathbf{x}'}$ achieves the optimal value of $(u_j^\top - u_j^\perp)$. Since the difference $(u_j^\top - u_j^\perp)$ is always positive, it is assumed $Y_j^{\mathbf{x}'} \geq 0$. If all constraints in \mathcal{U} are linear, the problem becomes a MIP.

If uncertainty over factor anchor values is specified using bounds, and therefore anchor utilities are independent across different factors, the computation is simplified as follows:

$$\begin{aligned} MR(\mathbf{x}, \mathbf{U}) &= \max_{\mathbf{x}' \in Feas(\mathbf{X})} \max_{\{u_j^\top, u_j^\perp\}} \sum_j (u_j^\top - u_j^\perp) \bar{r}_j^{\mathbf{x}, \mathbf{x}'} \\ &= \max_{\mathbf{x}' \in Feas(\mathbf{X})} \sum_j \max_{\{u_j^\top, u_j^\perp\}} (u_j^\top - u_j^\perp) \bar{r}_j^{\mathbf{x}, \mathbf{x}'} \\ &= \max_{\mathbf{x}' \in Feas(\mathbf{X})} \sum_j r_j^{\mathbf{x}, \mathbf{x}'} = \max_{\{I_j^{\mathbf{x}'}\}} \sum_j \sum_{\mathbf{x}'_j} r_j^{\mathbf{x}, \mathbf{x}'} I_j^{\mathbf{x}'}, \end{aligned} \quad (11)$$

where *pairwise factor regret* $r_j^{\mathbf{x}, \mathbf{x}'}$ is

$$\begin{aligned} r_j^{\mathbf{x}, \mathbf{x}'} &= \max_{\{u_j^\top, u_j^\perp\}} (u_j^\top - u_j^\perp) \bar{r}_j^{\mathbf{x}, \mathbf{x}'} \\ &= \begin{cases} [\max(u_j^\top) - \min(u_j^\perp)] \bar{r}_j^{\mathbf{x}, \mathbf{x}'}, & \text{if } \bar{r}_j^{\mathbf{x}, \mathbf{x}'} \geq 0, \\ [\min(u_j^\top) - \max(u_j^\perp)] \bar{r}_j^{\mathbf{x}, \mathbf{x}'}, & \text{if } \bar{r}_j^{\mathbf{x}, \mathbf{x}'} < 0. \end{cases} \end{aligned}$$

Intuitively, the condition that both anchor and local value constraints are independent across factors allows us to compute *local regrets* for each factor separately, allowing formulation as a MIP.

Minimax regret Our goal is to find a feasible configuration \mathbf{x}^* that minimizes maximum regret

$$MMR(\mathbf{U}) = \min_{\mathbf{x} \in Feas(\mathbf{X})} MR(\mathbf{x}, \mathbf{U}).$$

We can express this optimization as a MIP:

$$\begin{aligned} MMR(\mathbf{U}) &= \min_{\{Y_j^{\mathbf{x}}, m\}} m, \text{ subject to} \\ &\begin{cases} m \geq \sum_j \sum_{\mathbf{x}_j} \bar{r}_j^{\mathbf{x}, \mathbf{x}'} Y_j^{\mathbf{x}}, \forall \mathbf{x}' \in Feas(\mathbf{X}) \\ 0 \leq Y_j^{\mathbf{x}} \leq M_j I_j^{\mathbf{x}}, \forall j, \mathbf{x}_j \\ Y_j^{\mathbf{x}} \leq (u_j^\top - u_j^\perp), \forall j, \mathbf{x}_j \\ \mathcal{A}, \mathcal{C} \text{ and } \mathcal{U}. \end{cases} \end{aligned} \quad (12)$$

In practice, we avoid the exponential number of constraints (one for each feasible adversary configuration \mathbf{x}') using an iterative constraint generation procedure that generates the (small) set of active constraints at the optimal solution. This requires solving the above MIP in Eq. 12 with only a subset of constraints, generating the maximally violated constraint at the solution of this relaxed MIP (by solving the max regret MIP Eq. 10 or Eq. 11), and repeating until no violated constraints are found (see (Boutillier, et al. 2006) for details).

Multiattribute product databases

The MIP formulations above assume that the space of feasible configurations is defined by a set of constraints \mathcal{C} specifying allowable combinations of attributes. Alternatively, the set of choices may be the elements of a *multiattribute product databases*, in which the set of feasible outcomes is specified explicitly, namely, as the set of all products in the database. Preference-based search of, and choice from, such a database can be effected using minimax regret as well, but can in fact be somewhat simpler computationally.

For any two items \mathbf{x} and \mathbf{x}' in the database, pairwise regret $R(\mathbf{x}, \mathbf{x}', \mathbf{U})$ can be computed using Eq. 6. The max regret $MR(\mathbf{x})$ of \mathbf{x} is determined by considering its pairwise regret with each other item. To determine the optimal product (i.e., that with minimax regret), we compute the max regrets of each item in the database, and choose the one with the least. This latter computation can be sped up considerably by iteratively generating minimax optimal candidate products against a current set of ‘‘adversary’’ items and testing their optimality. In practice, much like constraint generation, this speed up reduces the complexity of the algorithm from quadratic to linear in the size of the database.

Elicitation strategies

Minimax regret allows one to bound the loss associated with the recommended decision relative to the (unknown) optimal. If this bound on utility loss is too high, more utility information must be elicited. A decision support system can query the user until minimax regret reaches some acceptable level (possibly optimality), elicitation costs become too high, or some other termination criterion is met. We propose a generalization of the *current solution (CS)* elicitation strategy, first described in (Boutilier, et al. 2006). This strategy has been shown empirically to be very effective in reducing minimax regret with few queries in several domains (Boutilier, et al. 2006; Boutilier, Sandholm, and Shields 2004). The CS strategy considers only parameters involved in defining minimax regret (i.e., the current regret-minimizing solution \mathbf{x}^* and the adversary's witness \mathbf{x}^w), and asks a query about the parameter that offers the largest potential reduction in regret. We define below how we *score* various query types, and then define potential query strategies.

Local queries The pairwise regret of regret-minimizing outcome \mathbf{x}^* and witness \mathbf{x}^w (the current solution) is:

$$\begin{aligned} R(\mathbf{x}^*, \mathbf{x}^w, \mathbf{U}) &= \max_{\{u_j^\top, u_j^\perp\}} (u_j^\top - u_j^\perp) \sum_j \sum_{i \in 1..N_j} \max_{\{v_j^i\}} C_j^i v_j^i, \\ &= \sum_j (\dot{u}_j^\top - \dot{u}_j^\perp) \sum_{i \in 1..N_j} C_j^i \dot{v}_j^i, \end{aligned}$$

where $C_j^i = C_{\mathbf{x}_j^w}^i - C_{\mathbf{x}_j^*}^i$, and $\{\dot{u}_j^\top, \dot{u}_j^\perp, \dot{v}_j^i\}$ are utility parameter values that maximize regret. A local bound query adds a constraint on a local parameter v_j^i . We wish to find the parameter v_j^i that offers the largest *potential* reduction in the pairwise regret $R(\mathbf{x}^*, \mathbf{x}^w, \mathbf{U})$ at the current solution, hence in the overall minimax regret. The linear constraints on local parameters induce a polytope defining the feasible space for the parameters *for each factor*. Our elicitation strategies use the bounding hyperrectangle of this polytope as an approximation of this feasible region. This allows for quick computation of query quality. (The bounding hyperrectangle can be computed by solving two very small LPs, linear in factor size.) Let $gap_j^i = v_j^{i\uparrow} - v_j^{i\downarrow}$. If we ask a bound query about the midpoint of the gap, the response narrows the gap by half (either lowering the upper bound or raising the lower bound). The impact of constraining v_j^i on the pairwise regret $R(\mathbf{x}^*, \mathbf{x}^w, \mathbf{U})$ is mediated by the magnitude of a constant C_j^i and the current value of a scaling factor $(\dot{u}_j^\top - \dot{u}_j^\perp)$. We define the heuristic *score* for querying parameter v_j^i , a measure of its potential for reducing minimax regret, as:

$$S(v_j^i) = (\dot{u}_j^\top - \dot{u}_j^\perp) \text{abs}(C_j^i) gap_j^i / 2$$

The best bound query is that with the highest score. Determining this is linear in the number of GAI parameters.

Scoring local comparison queries is a more complicated, since it is more difficult to estimate the impact of adding a linear constraint on minimax regret. We again approximate the feasible local parameter space with a bounding hyperrectangle. Given the current solution, we consider a list of

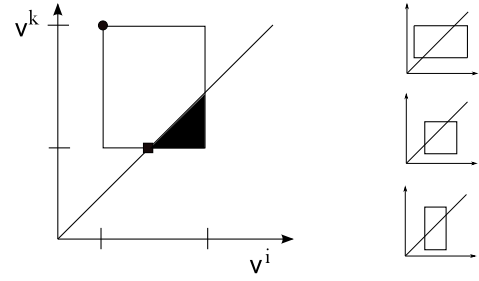


Figure 1: Four different ways to bisect a bounding rectangle. In all cases, if the response to a comparison query eliminates the part of the rectangle which contained the current solution point $(\dot{v}_j^i, \dot{v}_j^k)$ (marked with a circle), the new solution point (marked with a square) is one of the two intersections of the diagonal and the bounding rectangle. The shaded area approximates feasible parameter space after a response to a comparison query.

all pairs $\{(v_j^i, v_j^k)\}$ such that: (a) $C_j^i \neq 0$ and $C_j^k \neq 0$; (b) $v_j^{i\uparrow} \geq v_j^{k\downarrow}$ and $v_j^{k\uparrow} \geq v_j^{i\downarrow}$; and (c) the relationship between v_j^i and v_j^k is not known due to earlier queries. These conditions severely limit the number of pairs one must consider when determining the best local comparison query. The first condition eliminates many parameters from consideration because most coefficients C_j^i are zero. The second checks the bounds for implied relationships. Finally, the relationship between two parameters might already be known beforehand due to prior constraints or transitive closure of previous comparison constraints.

For each pair (v_j^i, v_j^k) considered, we compute a heuristic score as follows. First, we project the bounding hyperrectangle on the plane of the two parameters we are considering; the comparison constraint divides our 2-D rectangle along the 45-degree line. Fig. 1 shows all four cases and demonstrates that, after a response to a comparison query, the values of the parameters v_j^i, v_j^k (as well as the current level of regret) either remain the same, or they are pushed to lie at one of the two intersections of the diagonal with the bounding rectangle. In the latter case, the reduction in local regret can be approximated by

$$r_j^{i,k} = C_j^i \dot{v}_j^i + C_j^k \dot{v}_j^k - \max(C_j^i t_1 + C_j^k t_1, C_j^i t_2 + C_j^k t_2),$$

where (t_1, t_1) and (t_2, t_2) are the coordinates of the two intersections. The heuristic score for the query comparing \mathbf{x}_j^i to \mathbf{x}_j^k is:

$$S(\mathbf{x}_j^i, \mathbf{x}_j^k) = (\dot{u}_j^\top - \dot{u}_j^\perp) r_j^{i,k}.$$

The complexity of finding the best comparison query is linear in the number factors and quadratic in the number of local outcomes in each factor.

Global queries We use similar heuristic methods to compute the score of global anchor queries. In this case, we look at the impact of imposing constraints on anchor parameters u_j^\top, u_j^\perp , while keeping local regrets $\sum_i C_j^i \dot{v}_j^i$ constant. The resulting heuristic scores for both local and global queries are commensurable, allow comparison of different query types during elicitation.

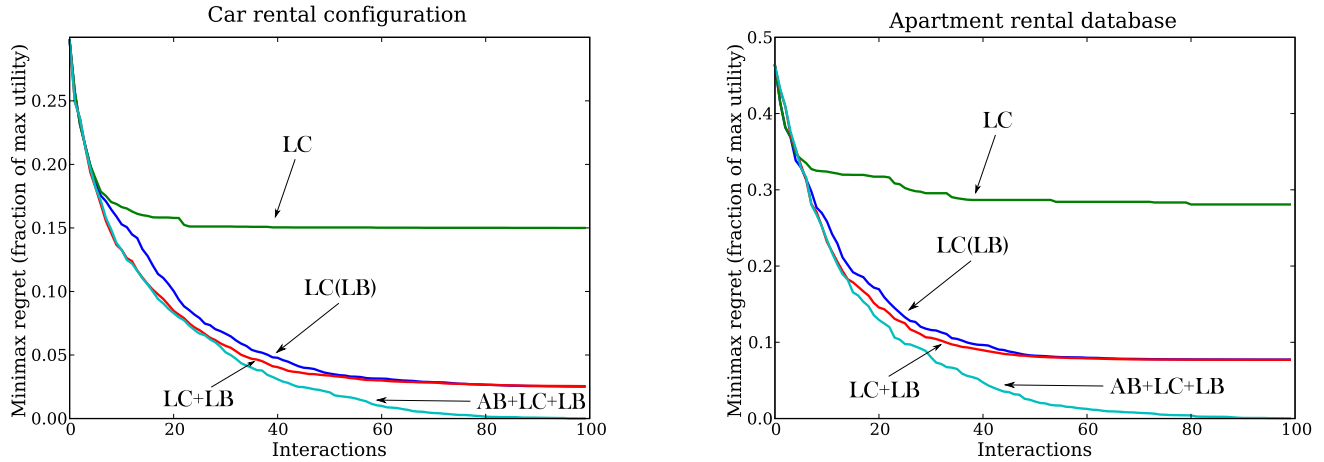


Figure 2: The performance of different query strategies on a) car rental configuration problem; b) apartment rental catalog problem. After averaging over 20 random instantiations of user utilities, the LB strategy curve was virtually indistinguishable from LC+LB; similarly, AB+LB was very close to AB+LC+LB. We omit these two curves for clarity.

Combining different queries If all types of queries are available, we can simply choose the next query to ask based on the heuristic score S described above. However, in general we want to consider not only the impact of a query in reducing regret, but also its cost to a user. Global queries are generally harder to answer than local queries; similarly, most users will find comparing two outcomes easier than dealing with bound queries (which require some numerical calibration w.r.t. anchors).

We consider several different strategies below that combine different query types. The *LC strategy* uses only local comparison queries; when our heuristic cannot recommend a comparison query, a comparison query is then picked at random. If instead of a random comparison query we select the best local bound query, we get the *LC(LB) strategy*. The *LB strategy* uses only local bound queries. The remaining strategies do not favor any query type, but simply recommends a query from the set of allowed types with the highest score: *LC+LB* combines local comparison and bound queries, and *AB+LC+LB* and *AB+LB* mix global anchor bound queries with local queries.

Experimental results

We tested our CS elicitation strategies on the car rental configuration problem from (Boutillier, et al. 2006; Braziunas and Boutillier 2005) and a small apartment rental database problem. The car-rental problem is modeled with 26 attributes that specify various attributes of a car relevant to typical rental decisions. The domain sizes of the attributes range from two to nine values, resulting in 6.1×10^{10} possible configurations. The GAI model consists of 13 local factors, each defined on at most five variables; the model has 378 utility parameters. There are ten hard constraints defining feasible configurations. The apartment rental problem comprises a database of 186 apartments, described by eight attributes, each having between two and 33 domain

values. The GAI model has five factors, and can be specified with 156 utility parameters. The code was implemented in Python, using CPLEX to solve the required MIPs in the car-rental problem (in the apartment database, regret computation does not require MIPs). Computing the current regret minimizing solution, which has to be updated after each query, takes about 1 second; determining the next query for any given strategy is even faster. Thus our approach admits real-time interaction.

We evaluated the six query strategies described above. Fig. 2 shows their performance on (a) the car rental configuration problem, and (b) the apartment rental database problem. The results are averaged over 20 random samples of the underlying user utilities as well as random prior bounds on utility parameters. The upper anchor bounds are drawn uniformly from $[1, 50]$, and lower bounds from $[-50, -1]$. The LVF bounds are drawn uniformly from $[0, 1]$.

With the exception of the LC strategy, all strategies (including those that use only local queries) exhibit a sharp initial reduction in minimax regret (from .30 to .05 with less than 40 interactions in the car-rental case). This means that in many cases we can either avoid costly global queries altogether or use them only in situations where very strict worst-case loss guarantees are required. Also, even though the LC strategy obviously does not perform as well as bound query strategies, we can notice that comparison queries (which are generally less costly in terms of user effort, time and accuracy than bound queries) are very effective during the first ten or so interactions, and do not hinder the performance of strategies in which they are used together with bound queries. Not surprisingly, only strategies that use anchor queries (*AB+LC+LB* and *AB+LB*) are able to reduce the regret level to zero; however, the performance of local-queries-only strategies, such as *LC(LB)*, *LC+LB* and *LB* is very encouraging.

Conclusions

We have provided a semantically justifiable approach to elicitation of utility functions in GAI models using the minimax regret decision criterion. The structure of a GAI model facilitates both elicitation and decision making via the semantically sound separation of local and global components. We described suitable forms of local and global queries and developed techniques for computing minimax optimal decisions under strict utility uncertainty, captured by linear constraints on the parameters of the GAI model. Unlike previous work that assumes the knowledge of either global scaling constants or local utility values, our elicitation strategies combine both local and global queries and provide a practical way to make good decisions while minimizing user interactions. The domains of application range from constrained configuration problems to preference-based search in multi-attribute product catalogs.

We are currently pursuing the extension of this work in several directions. We are investigating techniques for the effective elicitation of GAI utility structure (something we have taken as given in this work). We are also exploring the incorporation of probabilistic knowledge of utility parameters to help guide elicitation, while still considering regret in making final decisions (Wang and Boutilier 2003) and experimenting with additional query types. Finally, query strategies that take into account explicit query costs are of interest. Elicitation with human decision makers will allow us to consider the impact of psychological and behavioral issues—such as framing and ordering effects, sensitivity analysis, and the acceptability of different modes of interaction—on our normative model of elicitation.

References

- Fahiem Bacchus and Adam Grove. Graphical models for preference and utility. In *Proc. of UAI-95*, pages 3–10, Montreal, 1995.
- Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Proc. of AAAI-02*, pages 239–246, Edmonton, 2002.
- Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. UCP-Networks: A directed graphical representation of conditional utilities. In *Proc. of UAI-01*, pages 56–64, Seattle, 2001.
- Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170(8–9):686–713, 2006.
- Craig Boutilier, Tuomas Sandholm, and Rob Shields. Eliciting bid taker non-price preferences in (combinatorial) auctions. In *Proc. of AAAI-04*, pages 204–211, San Jose, CA, 2004.
- Darius Braziunas and Craig Boutilier. Local utility elicitation in GAI models. In *Proc. of UAI-05*, pages 42–49, Edinburgh, 2005.
- Peter C. Fishburn. Interdependence and additivity in multivariate, unidimensional expected utility theory. *International Economic Review*, 8:335–342, 1967.
- Christophe Gonzales and Patrice Perny. GAI networks for utility elicitation. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR2004)*, pages 224–234, Whistler, BC, 2004.
- Paul E. Green and V. Srinivasan. Conjoint analysis in consumer research: Issues and outlook. *Journal of Consumer Research*, 5(2):103–123, September 1978.

Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, New York, 1976.

Panos Kouvelis and Gang Yu. *Robust Discrete Optimization and Its Applications*. Kluwer, Dordrecht, 1997.

Ahti Salo and Raimo P. Hämäläinen. Preference ratios in multi-attribute evaluation (PRIME)—elicitation and decision procedures under incomplete information. *IEEE Trans. on Systems, Man and Cybernetics*, 31(6):533–545, 2001.

Leonard J. Savage. *The Foundations of Statistics*. Wiley, New York, 1954.

Tianhan Wang and Craig Boutilier. Incremental utility elicitation with the minimax regret decision criterion. In *Proc. of IJCAI-03*, pages 309–316, Acapulco, 2003.