Eliciting Bid Taker Non-price Preferences in (Combinatorial) Auctions

Craig Boutilier
Dept. of Computer Science
University of Toronto
Toronto, ON, M5S 3H5, Canada
cebly@cs.toronto.edu

Tuomas Sandholm
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA 15213, USA
sandholm@cs.cmu.edu

Rob Shields
CombineNet Inc.
Fifteen 27th Street
Pittsburgh, PA 15222, USA
rshields@combinenet.com

Abstract

Recent algorithms provide powerful solutions to the problem of determining cost-minimizing (or revenue-maximizing) allocations of items in combinatorial auctions. However, in many settings, criteria other than cost (e.g., the number of winners, the delivery date of items, etc.) are also relevant in judging the quality of an allocation. Furthermore, the bid taker is usually uncertain about her preferences regarding tradeoffs between cost and non-price features. We describe new methods that allow the bid taker to determine (approximately) optimal allocations despite this. These methods rely on the notion of *minimax regret* to guide the elicitation of preferences from the bid taker and to measure the quality of an allocation in the presence of utility function uncertainty. Computational experiments demonstrate the practicality of minimax computation and the efficacy of our elicitation techniques.

1 Introduction

Combinatorial auctions (CAs) generalize traditional market mechanisms to allow the direct specification of bids over *bundles* of items [13; 14] together with various types of side constraints [17]. This form of *expressive bidding* is extremely useful when a bidder's valuation for collections of items—or bidder's costs in reverse auctions—exhibit complex structure. The problem of *winner determination (WD)*, namely, determining a (cost or revenue) optimal allocation of items given a collection of expressive bids, is generally NP-complete [14]. However, algorithms have been designed for WD that work very well in practice [9; 16; 18]. Indeed, recent advances have impelled the application of CAs to large-scale, real-world markets, such as procurement.

Most of the recent literature on CAs has focused on algorithms that find optimal allocations w.r.t. cost or revenue. However, in many settings, features other than cost also play a role in assessing the quality of an allocation. For example, in a procurement auction (or reverse auction), the bid taker may be concerned with the number of suppliers awarded business, the percentage of business awarded to a specific supplier, the average delivery-on-time rating of awarded business, and any of a host of other factors that can be traded off against cost. Most algorithms can be adapted to deal with such features *if*, say, tradeoff weights are made explicit and incorporated into the objective function. However, most bid takers are unable or unwilling to articulate such precise tradeoffs (for reasons we elaborate below).

One way to deal with this issue is to allow users to generate a collection of allocations by imposing various constraints on these feature values (e.g., limiting the number of suppliers to five) and examining the implications on the optimal allocation (e.g., how cost changes) by rerunning WD. When this process ends, the user chooses one of the generated allocations as making the right tradeoffs between cost and the relevant non-price attributes. Unfortunately, this manual process of *scenario navigation* does nothing to ensure sufficient or efficient exploration of allocation or tradeoff space.[1]

In this paper, we explicitly view the process of choosing an optimal allocation as requiring a form of *preference elicitation* with the bid taker (i.e., the party determining the final allocation).[2] We assume that the way in which non-price features influence the choice of allocation can be modeled using a utility function, albeit one that is imprecisely specified. Using this observation, we develop a collection of techniques that can support, or even automate, the preference elicitation required to determine an (approximately) optimal allocation. As such, it can obviate the need for manual creation and exploration of different scenarios. Underlying these methods is the notion of *minimax regret*; this allows us to examine the worst-case error associated with optimizing an allocation in the presence of utility function uncertainty, thus giving *robust* allocations. It also provides valuable information to guide elicitation: it can suggest appropriate refinements of utility information, and when to terminate preference refinement.

We begin in Section 2 with a discussion of CAs and linear utility functions over non-price features. Section 3 addresses the issue of utility functon uncertainty and defines minimax

[1] In procurement settings, the bid taker generally maintains such flexibility in deciding the allocation. This means that there is no predetermined allocation rule (i.e., mapping from bids to allocations), unlike canonical auctions. Therefore, this is sometimes referred to as *procurement optimization* rather than an auction. These observations of typical use of WD software are based on our experiences with large-scale, real-world procurement problems.

[2] We distinguish the preference elicitation we study from forms of elicitation for CAs in which one attempts to minimize the amount of bid information needed from the *bidders* (e.g., [6; 7]).

regret as a suitable criterion for robust decision making. In Section 4 we define several techniques for computing minimax optimal allocations in CAs under different forms of utility function uncertainty. Section 5 describes two methods for preference elicitation based on minimax regret, and empirically evaluates these strategies on random problems with realistic structure and size, demonstrating the feasibility of our methods. We conclude in Section 6 with a discussion of future research.

2 CAs and Non-price-based Utility

We begin with a discussion of relevant concepts relating to CAs and utility for non-price features.

2.1 Combinatorial Auctions

CAs have found recent popularity as market mechanisms due to their ability to allow expressive bids reflecting the complex utility structure of market participants. As a result, economically efficient allocations of goods and services can be found using combinatorial bids in cases where noncombinatorial bids would not suffice. Recent research has focused on the problem of *winner determination* in various classes of CAs: finding an optimal allocation w.r.t. cost or revenue. While a computationally difficult problem [14], recent advances in optimization methods for CAs have made them a practical market technology [9; 16; 18].

The WD problem for a generic CA can be formulated as an integer program (IP). In this paper, we focus on *reverse CAs* where the goal is to obtain a collection of items at minimal cost, since procurement is a main motivation for our work. However, this is for concreteness only—all of our techniques apply *mutatis mutandis* to most variants of CAs (see [19] for a discussion of a number of such generalizations).

In a reverse CA, a *buyer* (bid taker) wishes to obtain a specific set of items G. *Sellers* submit bids of the form $\langle b_i, p_i \rangle$, where $b_i \subseteq G$ is a subset of items, and p_i is the price at which it is offered. Given a set of bids $B = \{\langle b_i, p_i \rangle : i \leq m\}$, we let an *allocation* be any subset of B. A *feasible allocation* is an allocation whose bids cover all items in G. The WD problem asks for the feasible allocation that supplies G as cheaply as possible, and can be formulated as a (linear) IP. Letting q_i be a boolean variable denoting that bid $\langle b_i, p_i \rangle$ was accepted, we wish to solve:

$$\min p_i q_i \;\; \textit{s.t.} \sum_{i: g \in b_i} q_i \geq 1, \; \forall g \in G$$

Note that we assume free disposal (i.e., obtaining multiple copies of any g incurs no penalty). This problem can be generalized to include multiple units of each item, no free disposal, various side constraints on the part of the sellers or the buyer, etc. Notice, however, that existing methods determine the optimal solution by minimizing only the (possibly constrained) *cost* of obtaining G.

2.2 Linear Utility for Non-price Features

While cost minimization is a primary objective in procurement, other features of the ultimate allocation will often play a role in assessing the quality of that allocation. For example, the buyer may prefer to deal with fewer suppliers, all else being equal, in order to minimize overhead. While a hard constraint might be imposed on the number of suppliers (e.g., less than five), this approach is undesirable, since rarely is a buyer unwilling to consider a larger number of suppliers if that reduces cost significantly. Instead there is a tradeoff—one is willing to pay more for a collection of items if the number of suppliers is less. There are a number of properties that are often used in procurement—to use just one domain of application—to assess the quality of allocations. Among these are: number of winners; (possibly aggregate) quality of certain items; percentage of volume given to a specific supplier; geographical diversity of winners; and many others.

Tradeoffs between price and non-price features can be captured by the buyer's *utility function*. We let X denote the set of feasible allocations given bids B. Let $F = \{f_1, \ldots, f_k\}$ be a set of *features* over which the buyer has preferences. We assume that the buyer's preferences for allocations $x \in X$ can be expressed by a *quasi-linear utility function* of the form

$$u(x) = \sum_{i=1}^{k} w_i f_i(x) - c(x), \tag{1}$$

where $c(x)$ is the cost of the allocation. Thus the utility function is completely characterized by a set of tradeoff weights w_i that express how much of each feature the buyer is willing to sacrifice for a given amount of money. We assume local value functions v_i that convert any non-numeric $f_i(x)$ into a numeric value $v_i(f_i(x))$; but we suppress mention of v_i for notational clarity.

Let $\mathbf{w} = \langle w_1, \ldots, w_k \rangle$ denote a *weight vector*. We assume all weights are non-negative (i.e., $\mathbf{w} \geq 0$), which implies that higher levels of each feature are preferred to lower levels.[3] We will use the notation $u(x; \mathbf{w})$ when we wish to emphasize the fact that utility is parameterized by a weight vector.

The independence and linearity of the local utility functions inherent in our formulation are, admittedly, fairly strong assumptions, but nonetheless often (roughly) hold in practice. Our techniques can be generalized to nonlinear local utility functions and *conditionally independent* features through various encoding tricks. We also assume that each feature $f_i(x)$ can be defined linearly in terms of attributes of allocation x. Again, if this is not so, certain approximations are possible. We briefly discuss these generalizations in Section 4.4.

3 Utility Function Uncertainty

In this section, we develop techniques for computing *robust* allocations when the parameters of the utility function are not precisely specified.

Given a known weight vector $\mathbf{w}$, our objective would be clear—find the feasible allocation $x^*_{\mathbf{w}}$ with maximum utility: $\max_x u(x)$. A number of WD algorithms can be adapted to this linear optimization problem. Unfortunately, buyers are usually uncertain about their tradeoff weights. This should not be surprising; the decision analysis literature is rife with evidence that decision makers have trouble constructing precise tradeoff weights [11]. And numerous methods for decision making and elicitation have been developed

[3] For instance, if fewer winners are preferred to more, the relevant feature is *negative* number of winners.

that account for this fact, both within decision analysis [23; 15] and AI [10; 5; 2; 4]. Furthermore, in procurement, the buyer is often faced with the difficult task of aggregating the preferences of multiple stakeholders via internal negotiation within the buying organization (the CFO preferring low cost, marketing people preferring high average delivery-on-time rating, plant managers preferring small numbers of suppliers, etc.). Thus, buyers are often unwilling to commit to specific tradeoff weights, often adopting the approach of "preference aggregation through least commitment."

In practice, we have found that buyers are often willing to provide loose bounds on these tradeoff weights, but unwilling to make things more precise without aid. As a result, buyers tend to ignore tradeoff weights altogether. Instead, they frequently use WD software to explore different *scenarios* by imposing various constraints on feature values and generating the corresponding optimal allocations. With a number of such allocations in hand, manual inspection is used to choose the most preferred allocation from this set. Unfortunately, while users find such manual *scenario navigation* appealing, it generally admits neither *sufficient* nor *efficient* exploration of allocation space.

In the next section, we develop techniques that will support, or even automate, this process. However, we first need methods for making allocational decisions in the presence of utility function uncertainty. Assume that through some interaction with the buyer, we have a set of linear constraints on the buyer's weight vector, with W denoting the set of feasible vectors.[4] Without full knowledge of $\mathbf{w}$, we cannot maximize utility (Eq. 1): whatever x we chose, there may be some feasible utility function for which x is far from optimal. Our goal instead is to find a *robust allocation* that, in the worst case, is as close to optimal as possible. In other words, even if an adversary were allowed to choose the utility function, we want x to be as close to optimal as possible.

Defn The *regret* of allocation x w.r.t. utility function $\mathbf{w}$ is

$$R(x, \mathbf{w}) = \max_{x' \in X} u(x'; \mathbf{w}) - u(x; \mathbf{w}) \qquad (2)$$

The *maximum regret* of x w.r.t. feasible utility set W is

$$MR(x, W) = \max_{\mathbf{w} \in W} R(x, \mathbf{w}) \qquad (3)$$

The *pairwise max regret* of x w.r.t. x' (with W understood) is

$$MR(x, x') = \max_{\mathbf{w} \in W} u(x'; \mathbf{w}) - u(x; \mathbf{w}) \qquad (4)$$

The *minimax optimal (or robust) allocation* w.r.t. W (i.e., the allocation with *minimax regret*) is

$$x^*_W = \arg\min_{x \in X} \max_{x' \in X} MR(x, x') \qquad (5)$$

Minimax regret is one of the more appealing criteria for decision making under *strict uncertainty* (i.e., where distributional information is lacking) [20; 8]. It has found (in modified form) application to analysis of online algorithms [1]. Only recently has it been used in the explicit modeling of utility function uncertainty and elicitation [3; 15]. The minimax optimal allocation is attractive due to its *robustness* against utility uncertainty. The key difficulty in the application of minimax regret in practical settings is the computation of minimax optimal decisions and its use in elicitation, though recently computationally effective techniques have begun to emerge [22; 4].

[4]The elicitation methods below will make it clear exactly how these constraints arise. Linearity is important computationally, but is not critical to the definition of max regret that follows.

4 Computation of Minimax Regret

Most WD methods cannot be applied directly to computation of the minimax optimal allocation. Specifically, Eq. 5 requires solution of a minimax program (with a min over two maxes), preventing the application of very effective linear optimization techniques. In this section we describe an algorithm to compute the minimax optimal allocation in a way that uses standard integer programming or WD methods iteratively. We distinguish two settings. In the first, W is defined by upper and lower bounds for each weight w_i, these being quite natural to specify directly. In the second, arbitrary linear constraints on the w_i define W; these naturally arise when a buyer compares or ranks two or more alternatives.

4.1 Upper and Lower Weight Bounds

Computing Max Regret We begin with computation of max regret (Eq. 3). Suppose W is a hyperrectangle in weight space defined by an upper bound $w_i\Uparrow$ and lower bound $w_i\Downarrow$ on each parameter w_i. The max regret of a specific allocation x w.r.t. W is given by the obvious optimization program:

$$MR(x) = \max_{\mathbf{w} \in W} \max_{x' \in X} \sum_{i=1}^{k} w_i[f_i(x') - f_i(x)] - c(x') + c(x) \quad (6)$$

This poses the difficulty that the objective is quadratic, since both the w_i and the adversarial allocation x' are variables. Fortunately, since W is specified by upper and lower weight bounds, we can rewrite this as follows:

$$\max_{x' \in X} c(x) - c(x') + \sum_{i=1}^{k} \left\{ \begin{matrix} w_i\Uparrow(f_i(x') - f_i(x)) & \text{if} & f_i(x') > f_i(x)) \\ w_i\Downarrow(f_i(x') - f_i(x)) & \text{if} & f_i(x') \leq f_i(x)) \end{matrix} \right\}$$

With some clever encoding, we can make this objective linear, and rewrite this optimization as an IP. Specifically, we can linearly define variable A_i^+ to indicate that $f_i(x') > f_i(x)$ (and A_i^- the opposite), as well as variable F_i^+ (resp., F_i^-) that takes value $f_i(x')$ if A_i^+ is true and 0 otherwise (resp., if A_i^- is true). We then solve the following objective:

$$\max \sum_{i=1}^{k} \{ w_i\Uparrow[F_i^+ - f_i(x)A_i^+] + w_i\Downarrow[F_i^- - f_i(x)A_i^-] \} + c(X) - c(x)$$

Thus with a linear IP we can compute the max regret of any given allocation x.

Computing Minimax Regret Computing the allocation with minimax regret requires a minimax program:

$$\arg\min_{x \in X} \max_{\mathbf{w} \in W} \max_{x' \in X} \sum_{i=1}^{k} w_i[f_i(x') - f_i(x)] + c(x') - c(x)$$

We can convert this to a standard minimization using the following conceptual trick:

$$\min_{x \in X} \ \delta$$

$$\text{s.t. } \delta \geq \sum_{i=1}^{k} w_i[f_i(x') - f_i(x)] + c(x') - c(x), \ \ \forall x', \mathbf{w} \quad (7)$$

The difficulty lies in the fact that these constraints over all $x' \in X$ and $\mathbf{w} \in W$ cannot be enumerated explicitly.

Fortunately, for any specific x', we need not enumerate the (continuous) set of possible weight vectors W; rather we can identify the unique weight vector $\mathbf{w}$ (hence the only active constraint for x') that maximizes the regret of x w.r.t. x' by setting each w_i to either its upper or lower bound (depending on the relative values of this feature in x and x'). This can be accomplished using a strategy much like the one outlined above, using a specific indicator variable for each x'. This still leaves the problem of enumerating all allocations x' to form the constraint set above. To circumvent this, we use an iterative constraint generation procedure, whereby we solve relaxed versions of the problem, gradually adding constraints for various x' until we are assured that all active constraints in the full program are present.[5]

Note that in defining this program, we use tricks similar to those used to compute max regret. However, now allocation variables refer to "our" allocation (we are minimizing over these variables), not the adversary's allocation. The adversary's allocations will be fixed within specific constraints. The formulation is similar to the max regret computation except that we require variables $A_i^+[x']$, $A_i^-[x']$, $F_i^+[x']$, and $F_i^-[x']$ for each adversarial x' present in the constraint set (similar in meaning to those above).

The constraint generation procedure works as follows:

1. Let *Gen* $= \{x'\}$ for some arbitrary feasible x'.
2. Solve the following IP:

   $$\min \delta$$

   $$\text{s.t. } \delta \geq \sum_{i=1}^{k}\{w_i\!\uparrow\!(f_i(x')A_i^+[x'] - F_i^+[x']) + w_i\!\downarrow\!(f_i(x')A_i^-[x'] - F_i^-[x'])\} + c(x') - c(X) \quad \forall x' \in \textit{Gen}$$

   $$\text{and definitional constraints} \quad \forall x' \in \textit{Gen}$$

   Let x^* be the IP solution with objective value δ^*.
3. Compute the minimax regret of allocation x^* using the IP above, producing a solution with regret level r^* and adversarial allocation x''. If $r^* > \delta^*$, then add x'' to *Gen* and repeat from Step 2; otherwise (if $r^* = \delta^*$), terminate with minimax optimal solution x^* (with regret level δ^*).

The correctness of this procedure is easy to verify. The constraints ensure that δ is greater or equal to the pairwise regret level $MR(x^*, x')$ of the solution x^* and every $x' \in$ *Gen*, and the minimization ensures that x^* has the minimum such regret. Given the solution x^* to this relaxed IP, we can easily determine the adversarial allocation x'' that maximizes regret w.r.t. x^*. If the max regret of x^* is no greater than the δ^*, we are assured that all unexpressed constraints (for those x' outside *Gen*) are satisfied. If the computed max regret level is greater than δ^*, then the corresponding constraint is added, thus making "maximal" progress towards a true solution.

4.2 Arbitrary Weight Constraints

When W is given by a set of arbitrary linear constraints, we can't exploit weight bounds in the manner above. However, we can use an alternating optimization technique to compute max regret when W is so defined using only a sequence of IPs and LPs. Note that if we fix a weight vector $\mathbf{w}$, the quadratic program in Eq. 6 becomes a linear IP; and if we fix the allocation x', it becomes an LP. So to solve Eq. 6 we start by fixing $\mathbf{w}$ to some random feasible weight vector $\mathbf{w}_1$. We then iterate over the following steps:

(a) Given a fixed $\mathbf{w}_i$, solve the IP above, obtaining solution x'_i, with max regret m'_i.

(b) Given the fixed x'_i, solve the LP above, obtaining solution $\mathbf{w}_{i+1}$ with max regret m_{i+1}.

This alternation stops whenever the max regret computed at two successive levels is identical. Since the computation of a new $\mathbf{w}$ is an LP, the algorithm will terminate with a solution $\mathbf{w}$ whose value is equal to that of some $\mathbf{w}'$ lies at a vertex of the polytope W. This ensures convergence though perhaps only at a local maximum. We can integrate this scheme with a constraint generation procedure to compute an approximately minimax optimal allocation with some minor modifications in a straightforward way.

When the dimensionality of W is small, we can improve the procedure. First, we run a standard vertex enumeration algorithm once to compute the vertices of W—this will be a small set if only a few features are relevant to utility. At each iteration of the procedure, rather than repeatedly solving IPs and LPs to generate a new constraint, we can find the adversarial allocation x' that maximizes the regret of x^* as follows: we compute the $x_{\mathbf{w}}$ that maximizes the regret of x^* w.r.t. each weight vector $\mathbf{w}$ in the vertex set; the maximum of these gives the x' and $\mathbf{w}$ at which the regret of x^* is maximized. This x' (with suitable setting of weights) is then added to the constraint set.[6] This removes the LP from each iteration and ensures convergence to a global maximum. Indeed, one can avoid constraint generation altogether and simply impose the constraints for each $\mathbf{w}, x_{\mathbf{w}}$ pair and solve a single IP.

4.3 Empirical Evaluation

We tested the effectiveness of our procedures on some large scale, real-world reverse CAs. The goal in this section is to demonstrate the computational feasibility of our constraint generation procedure for practical problems. Here we focus on the computation of minimax regret for problems with weight bounds. (We discuss arbitrary linear constraints in Section 5.2.)

For these experiments, we generated random reverse CAs which have a similar structure to a class of transportation procurement problems encountered frequently in the real world. In these instances, the items to be procured are divided into

[5] This related to Bender's reformulation for mixed IPs [12].

[6] This is much like Bender's decomposition [12].

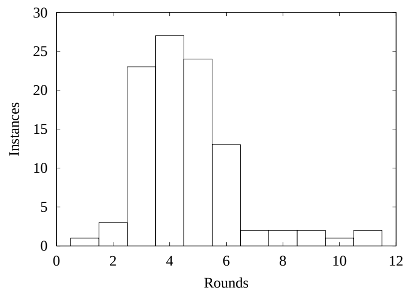

Figure 1: Number of constraints generated.

five regions; the relevant features are the number of winning suppliers in each region, and the number of winning suppliers overall (yielding six features in total). Each instance has ten suppliers, and ten times as many bids as there are items. The bids are on individual items only.[7]

All experiments were conducted for the case where w is defined by upper and lower bounds for each weight w_i. The initial bounds were chosen to reflect plausible tradeoff weights for each feature, and are reasonably (but plausibly) loose.

To measure the computational feasibility of the minimax optimization, we tracked how many rounds of constraint generation were required to determine the minimax optimal allocation—since each iteration requires the solution of two WD problems, this gives a sense of how the problem will scale with one's favorite WD algorithm. We first test how the number of generated constraints scales with problem size. Testing problems sizes of 10, 20, 30, 40, and 50 items (at each size, we generated 10 instances), we found the average number of constraints generated to be 3.3., 3.8, 3.6, 3.8, and 3.1, respectively. Figure 1 shows the distribution of constraint generation rounds for a larger set of instances (100 instances with 500 bids and 50 items). The vast majority of instances required only 3 or 4 rounds of constraint generation.[8] Interestingly, the average number generated constraints stays nearly constant as problem size increases. This is very promising for the scalability of this technique to even larger instances (at least from problems in this class), since it seems to depend only on how well the underlying WD method itself scales.

Should the number of required constraints become infeasible, approximate solutions can be obtained in an anytime fashion by early termination of constraint generation (we do not test this behavior here since the number of required constraints is so small). Faster convergence can also be achieved by "seeding" the constraint generation procedure with the constraints used in related versions of the problem; for instance, in the elicitation process, once new utility information is received, the new minimax problem can be seeded with the constraints from the prior, closely related problem. Finally, the solution time per generated constraint can be shortened by using approximate WD.

[7] Here the combinatorial complexity of WD stems solely from consideration of the number of winners.

[8] Average WD solution time for these instances was 19.37s.

4.4 Generalizations

In this section, we briefly hint at how the simple linear utility model can be generalized and minimax regret computation adjusted to reflect these changes.

First, we note that independence of features is not critical. If the utilities of some features are conditionally dependent on those of others, but this dependence is sparse (e.g., as captured in a graphical model [3]), the number of utility parameters remains small, and typically the utility of an allocation is a linear function of these parameters.

Uncertainty in a nonlinear local utility function for a specific feature f_i cannot be represented using bounds (or constraints) on a single weight w_i. However, even if the utility function has an unknown, or nonparametric form, we can still compute minimax regret if we have upper and lower bounds on the utility of *sampled points* from the domain of f_i. We defer details to a longer version of this paper. From the point of view of elicitation, we can address the question of which sampled points offer the greatest (potential) improvement in decision quality. Finally, dealing with certain types of features that cannot be defined by linear constraints is also possible. For certain features, we can impose bounds on their values with linear constraints. This permits the discretization of feature values, allowing one to define utility as a function of the specific discrete level taken by that feature.

5 Preference Elicitation

A critical component of WD when non-price features play a role is *preference elicitation* from the bid taker. While the minimax regret methods described above allow robust decisions to be made in the face of utility function uncertainty, these must be coupled with tools that support a buyer in refining her preferences. Uncertainty must be reduced if regret is to reach an acceptable level. Regret methods have a key advantage in this respect—when regret reaches a satisfactory level, further refinement of the utility function can be *proven* to be unnecessary. Thus, in a practical sense, minimax regret can help focus elicitation on the *relevant* parts of the buyer's utility function.[9]

In this section, we describe two classes of techniques in which minimax regret is used to drive the process of elicitation. While a number of methods can be used for elicitation, we focus here on two especially simple techniques, one involving direct manipulation of weight bounds, the second involving comparisons of automatically chosen outcomes.[10] We provide detailed empirical results are presented for the first method, and preliminary results for the second.

5.1 Direct Weight-bound Manipulation

In our first method the buyer directly manipulates the upper and lower bounds (via a graphical interface) on the weights w_i, using regret-based feedback. We assume the buyer has provided some initial (possibly crude) bounds on each

[9] Also, different stakeholders in the buying organization can decide to end their internal negotiation when the benefit from further negotiation (i.e., max regret) becomes provably small.

[10] We are also exploring various hybrids of these methods, including their combination with manual navigation.

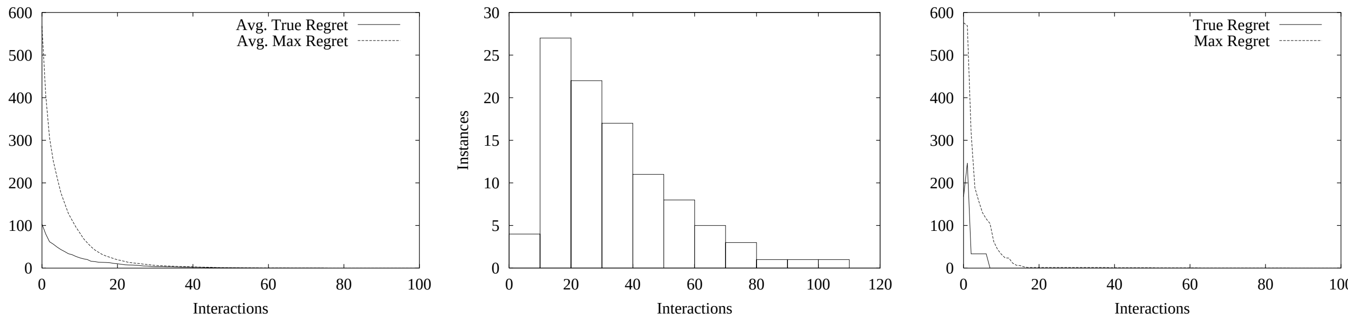

Figure 2: (a) Avg. max regret and avg. true regret as a function of number of slider *interactions* (each interaction is one tightening of one bound of one feature weight); (b) Number of interactions required to reach zero max regret; (c) Reduction of max regret on a specific instance (true regret is zero immediately).

weight. At each round k of interaction, we display the following information: a slider demonstrating the range of values for each w_i, with bounds $w_i^k\!\uparrow$ and $w_i^k\!\downarrow$ defining the feasible weight set W^k at round k; the minimax optimal allocation x^k; the adversarial allocation a^k; the weights w_i^a chosen by the adversary that maximize regret of x^k (these will lie at either an upper or lower bound, and are highlighted on each slider); and the (adversarial) utility of both x^k and a^k, as well as their difference (the minimax regret).

The buyer is asked to manipulate the bounds. If the buyer prefers x^k to a^k (or feels that regret overstates the amount by which a^k is preferred to x^k), we ask the buyer to adjust one of the bounds w_i^a at which the adversary's utility function lies. As this bound is adjusted up (in the case of a lower bound) or down (upper bound), the pairwise max regret $MR(x^k, a^k)$ must be reduced, and can be updated in real-time without reoptimization, providing the user with real-time feedback. (The only exception is when a^k and x^k agree on feature f_i.) However, if a^k is preferred to x^k, the user must adjust one of the w_i bounds opposite from that chosen by the adversary. In this case, pairwise regret is unaffected, and no immediate feedback is provided. After these manipulations, the new (tighter) bounds are used to recompute the minimax optimal allocation and related quantities. The process terminates when minimax regret is reduced to an acceptable level.

For experimental purposes, we simulate a user's behavior with a simple model which favors moving bounds that give immediate feedback. A *true weight* is chosen randomly for each feature using a truncated Gaussian distribution centered at the midpoint of the unknown weight interval $[w_i\!\uparrow, w_i\!\downarrow]$, with variance $(0.25(w_i\!\uparrow - w_i\!\downarrow))^2$. This reflects the fact that users are more likely to have true utility closer to the middle of their assessed intervals.[11] Each move of a bound by the simulated user removes half of the distance between the current bound and the true weight. At any stage, a user will move an "adversarial bound" (i.e., a bound at which the adversarial utility w_i^a is located) if some such bound is more than $0.1(w_i\!\uparrow - w_i\!\downarrow)$ from the true weight w_i; such moves provide immediate feedback, and if more than one such feature exists the bound that provides the largest reduction in $MR(x^k, a^k)$ is moved. Otherwise, if the adversarial bound is very close to the true utility weight (within 10% of the total slider range), the user moves an opposite bound.

Despite the rather cautious nature of the "simulated user," we find that the number of interactions required to find robust allocations is typically quite small. The anytime nature of utility refinement is evident in Figure 2(a), which shows max regret of the minimax optimal allocation as a function of the number of interactions (averaged over 100 50-item, 500-bid instances).[12] Max regret reduces very rapidly with the initial interactions. Reducing max regret to zero is more difficult, but only for some instances. On many instances, max regret is driven to zero very quickly, as shown in Figure 2(b). One instance did not reach zero max regret within 100 interactions, but it did reach very low max regret quickly, as exemplified in Figure 2(c). Note that *true regret*—the error in the minimax optimal allocation w.r.t. the true utility function—is considerably less than max regret (see Figure 2(a)). The anytime nature of the approach is critical: it allows one to terminate when max regret reaches an acceptable level, relative to the cost of further interaction.

5.2 Comparison Queries

The manipulation of bounds provides the user with direct control of utility function refinement. Alternatively, an elicitation scheme in which we directly *query* the user about specific allocations gives much more control to the system.

We have engaged in preliminary investigation of simple *comparison queries*. This involves asking the user to compare two allocations: "Do you prefer x' to x?"[13] A (yes/no) answer to such a query induces a linear constraint on possible weight vectors; for instance, if x is preferred to x', we impose the linear constraint

$$w_i f_i(x) - c(x) > w_i f_i(x') - c(x')$$

[11] We do not exploit this information in the querying process. Experiments with uniformly drawn utilities have qualitatively similar results, but require *slightly* more interaction.

[12] Optimal allocations have average utility of 23,960.

[13] Such comparisons need only involve the utility-bearing features of the allocations in question and the cost of each; the entire allocations need not be presented.

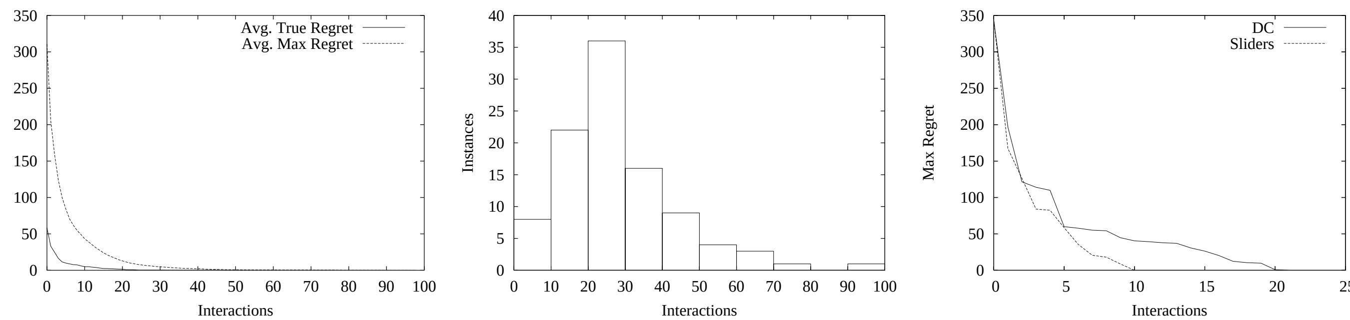

Figure 3: (a) Avg. max regret and avg. true regret as a function of number of *comparison queries*; (b) Number of queries required to reach zero max regret; (c) Reduction of max regret on a specific instance (vs. slider interactions).

on the set of weight vectors. We can also allow the user to express rough indifference by responding "I'm not sure," treating this as meaning $|u(x) - u(x')| \leq \epsilon$ for some small ϵ, and imposing the constraint the these values are ϵ-close.

The goal then is to construct a query plan that reduces the feasible region W in a fashion that reduces the max regret of the minimax optimal allocation quickly. A number of alternative query policies can be considered; to date we have focused on a very simple, yet promising strategy, in which the user is repeatedly asked to compare her minimax optimal allocation with the adversary's allocation. Any response to that query will generally provide valuable information. Should the user prefer the minimax optimal allocation, this immediately rules out the adversary's chosen utility function, and will generally reduce regret. Conversely, should the user prefer the adversary's allocation, this rules out the current minimax optimal allocation, thus imposing a constraint that forces a new (minimax optimal) decision to be made.

The ability to compute minimax regret for arbitrary polytopes W, as discussed above, is critical when dealing with comparison queries. We did some preliminary investigation of the comparison query strategy using the vertex enumeration scheme discussed in Section 4.2. After each query response, a new linear constraint is imposed on W. The optimal allocation $x_{\mathbf{w}}$ is computed for each vertex $\mathbf{w}$ of W and the minimax optimal allocation is then determined with a single IP. Note that since the vertices of W at each iteration are identical to the previous vertices except for those incident with the new constraint, we need only run a WD algorithm at the (small) set of *new* vertices.

Figure 3(a) shows average max-regret and true-regret reduction offered by the comparison query model (averaged over 100 30-item, 300-bid instances). The same desirable anytime nature as seen with slider interactions is evident, with regret reducing very quickly with just a few initial queries. Figure 3(b) shows the number of queries required to reach zero max regret (though as with sliders, this measure is less important than the anytime profile). Figure 3(c) illustrates performance on a typical instance, showing that regret reduces very quickly with the number of queries. For comparison, we show regret reduction with the slider mode of interaction on the same instance. While sliders generally require slightly fewer interactions, it is important to note that more information is generally being provided with each slider interaction than is contained in the response to a comparison query. A suitable measure of interaction cost is needed to draw definitive conclusions regarding the relative merits of the two approaches.

Computationally, the comparison query approach is quite fast for problems of this size, and is certainly able to support online interaction. Using the simple, non-iterative, vertex-enumeration approach described in Section 4.2 on the 100 30-item, 300-bid instances, we found that the W polytope had on average 253 vertices when querying terminates at the optimal (zero-regret) solution. The WD problem for each vertex is solved in 0.13s on average, but recall that after each interaction, only new vertices need to have their WD problems solved. On average, 54 vertices are added per query (each initial computation is on a hyperrectangle with 64 vertices, reflecting initial bounds). Finally, computation of the minimax optimal solution after each query (using the vertex solutions as constraints) takes an average of 3.5s. Thus average computation time per query is under 11 seconds. There are considerable opportunities for improving this performance as well.

6 Concluding Remarks

We have described new methods that allow a bid taker to find an (approximately) optimal allocation despite uncertainty in utility for non-price criteria. These methods rely on the notion of *minimax regret* to guide the elicitation of preferences from the bid taker on an as-needed basis, and to measure the quality of an allocation in the presence of utility function uncertainty. This provides a basis for deciding when to terminate utility refinement and for determining a robust allocation. Our computational experiments demonstrate the practicality of minimax computation and the efficacy of our elicitation techniques.

While we presented our techniques for allocation-level features, they can directly handle item-level (and bid-level) features as well (e.g., color, delivery date, etc.). The bid taker can leave certain item features unspecified, allowing bidders to specify various alternative (e.g., red on Monday for $10, or blue on Wednesday for $7).[14] The bid taker's utility function

[14]Preference elicitation *from the bidders* (using techniques analogous to ascending-price auctions), has been studied in single-item

can depend on these features, and our methods can be used if the bid taker is uncertain about that function. Our techniques and notation stay unchanged, but now each allocation (i.e., vector x) not only includes a decision variable (accept or reject) for each bid, but also variables describing how the additional features are set (e.g., what delivery date and color the bid taker chooses).

We plan to conduct more systematic experiments with various forms of comparison queries. Apart from the "current solution" strategy for choosing queries, we intend to investigate other methods for choosing the allocations that the user is asked to compare. We will also explore some of the iterative approaches to minimax optimization in the presense of arbitrary weight contraints for problems where vertex enumeration is infeasible.

Other important directions include the study of additional query types. In situations where computational bottlenecks arise, investigation of anytime constraint generation and anytime WD would prove interesting. We also plan to field these techniques in procurement applications.

Acknowledgements

This research was conducted at and supported by CombineNet, Inc.

References

[1] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, 1998.

[2] Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 239–246, Edmonton, 2002.

[3] 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.

[4] 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.

[5] 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.

[6] Wolfram Conen and Tuomas Sandholm. Preference elicitation in combinatorial auctions: Extended abstract. In *Third ACM Conference on Electronic Commerce (EC'01)*, pages 256–259, Tampa, FL, 2001. A more detailed version appeared in the IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms, pp. 71–80.

[7] Wolfram Conen and Tuomas Sandholm. Partial-revelation VCG mechanism for combinatorial auctions. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 367–372, Edmonton, 2002.

[8] Simon French. *Decision Theory*. Halsted Press, New York, 1986.

[9] Yuzo Fujisima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 548–553, Stockholm, 1999.

[10] Vu Ha and Peter Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 215–222, Providence, RI, 1997.

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

[12] George L. Nemhauser and Laurence A. Wolsey. *Integer Programming and Combinatorial Optimization*. Wiley, New York, 1988.

[13] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[14] Michael H. Rothkopf, Aleksander Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[15] Ahti Salo and Raimo P. Hämäläinen. 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.

[16] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 153:1–154, 2002.

[17] Tuomas Sandholm and Subash Suri. Side constraints and non-price attributes in markets. In *Proceedings of the IJCAI-01 Workshop on Distributed Constraint Reasoning*, Seattle, 2001.

[18] Tuomas Sandholm, Subash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 1102–1108, Seattle, 2001.

[19] Tuomas Sandholm, Subash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 69–76, Bologna, 2002.

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

[21] Aditya Sunderam and David C. Parkes. Preference elicitation in proxied multiattribute auctions. In *Fourth ACM Conference on Electronic Commerce (EC'03)*, pages 214–215, San Diego, 2003.

[22] 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*, Acapulco, 2003. to appear.

[23] 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.

auctions with item-level features (e.g., [21]).