Preference Elicitation in Combinatorial Auctions [Extended Abstract]

Wolfram Conen
XONAR GmbH
Wodanstr. 7
42555 Velbert, Germany
E-mail: conen@gmx.de

Tuomas Sandholm [*]
Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213
E-mail: sandholm@cs.cmu.edu

ABSTRACT

Combinatorial auctions (CAs) where bidders can bid on bundles of items can be very desirable market mechanisms when the items sold exhibit complementarity and/or substitutability, so the bidder's valuations for bundles are not additive. However, in a basic CA, the bidders may need to bid on exponentially many bundles, leading to difficulties in determining those valuations, undesirable information revelation, and unnecessary communication. In this paper we present a design of an auctioneer agent that uses topological structure inherent in the problem to reduce the amount of information that it needs from the bidders. An analysis tool is presented as well as data structures for storing and optimally assimilating the information received from the bidders. Using this information, the agent then narrows down the set of desirable (welfare-maximizing or Pareto-efficient) allocations, and decides which questions to ask next. Several algorithms are presented that ask the bidders for value, order, and rank information. A method is presented for making the elicitor incentive compatible.

1. INTRODUCTION

Combinatorial auctions where bidders can bid on bundles of items can be desirable market mechanisms when the items exhibit complementarity or substitutability, so the bidder's valuations for bundles are not additive. One of the problems with these otherwise desirable mechanisms is that determining the winners is computationally complex. There has been a recent surge of interest in winner determination algorithms for such markets [11, 13, 3, 15, 16].

Another problem, which has received less attention, is that combinatorial auctions require potentially every bundle to be bid on, and there are exponentially many bundles. This is complex for the bidder because she may need to invest effort or computation into determining each of her valuations [12, 14, 5, 6, 8]. It can also be undesirable from the perspective of revealing unnecessary private information and from the perspective of unnecessary communication.

*This work was funded by, and conducted at, CombineNet, Inc., 311 S. Craig St., Pittsburgh, PA 15213.

In this paper we present a blueprint for a software agent (an *elicitor*) for the auctioneer that will intelligently ask the bidders the right questions for determining good allocations without asking unnecessary questions. Each of our algorithms is incremental in that it requests information, optimally propagates the implications of the answer, and does this again until it has received enough information.

The key observation of this paper is that topological structure that is inherent in the problem can be used to intelligently ask only relevant questions about the bidders' preferences while still finding the optimal (welfare-maximizing and/or Pareto-efficient) solution(s). Based on the information, the auctioneer agent can narrow down the set of potentially desirable allocations, and decide which questions to ask the bidders next.

We first present our topological observations. We then lay out two algorithms that capitalize on those observations and query the bidders selectively using restricted query policies. Then, we introduce data structures that are used to store and propagate richer information received from the bidders, and we discuss algorithms that support completely general policies for querying the bidders. Finally we discuss the efficiency of elicitation, and how to make it incentive compatible.

2. COMBINATORIAL AUCTION SETTING

In a combinatorial auction, the seller has a set $\Omega = \{\omega_1, \dots, \omega_m\}$ of indivisible, distinguishable items that she can sell. Any subset of the items is called a *bundle*. The set of bidders (buyers) is called $N = \{1, \dots, n\}$.[1] Each bidder has a value function $v_i : 2^\Omega \to \mathbf{R}$ that states the value that the bidder is willing to pay for any given bundle.

A *collection* $(X_1, \dots, X_n)$ states which bundle $X_i \subseteq \Omega$ each bidder i receives. In a collection, some bidders' bundles may overlap in items, which would make the collection infeasible. We call a collection an *allocation* if it is feasible, i.e., each item is allocated to at most one bidder. The possibility that $X_i = \emptyset$ is allowed.

An allocation X is *welfare maximizing* if it maximizes $\sum_{i=1}^n v_i(X_i)$ among all allocations (feasible collections). We call an allocation X *Pareto efficient* if there is no other allocation Y such that $v_i(X_i) \geq v_i(Y_i)$ for each bidder i and strictly for at least some bidder i.

[1]In our model, the seller has zero reservation prices on all bundles, i.e., she gets no value from keeping them. If in reality she has reservation prices on bundles, that can be modeled by treating the seller as one of the bidders who submits bids that correspond to the reservation values.

3. TOPOLOGICAL STRUCTURE IN COMBINATORIAL AUCTIONS

We observed significant topological structure in the combinatorial auction setting. We use that to avoid asking the bidders unnecessary questions about their valuations.

3.1 Rank Lattice and Associated Algorithms

Conceptually, the bundles can be ranked for each agent from most preferred to least preferred. This gives a unique rank for each bundle for each agent. The key observation behind the *rank lattice* is the following. Without referring to the values of the bundles, each collection can be mapped to a unique rank vector $[R_1(X_1), R_2(X_2), \ldots, R_n(X_n)]$. The set of rank vectors, and a *dominates*[2] relation between them define a lattice. Now, the important fact is that if a collection (resp. its rank vector) is feasible (i.e., is an allocation), then no collection (resp. its rank vector) "below" it can be a better solution to the allocation problem.

EXAMPLE 1. *Let there be two goods, A and B, and two agents, a_1, and a_2. The agents rank the bundles as follows.*
Agent a_1: $(1: AB, 2: A, 3: B, 4: \emptyset)$
Agent a_2: $(1: AB, 2: B, 3: A, 4: \emptyset)$

This implies the rank lattice of Fig. 1. Only a subset of the collections is feasible and, thus, corresponds to allocations.

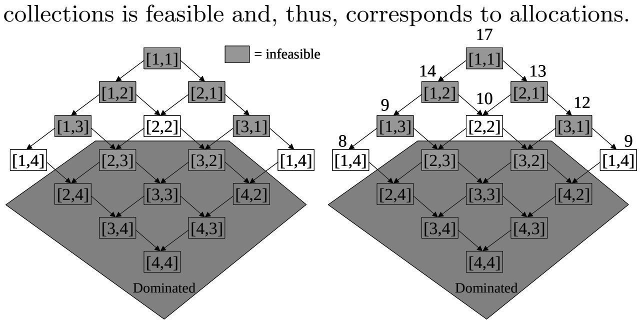

Figure 1: Left: Rank lattice for Example 1. The nodes are collections. Some of them are dominated, some are infeasible, some are both, and some are neither. Right: Rank lattice, augmented with sums of values for Example 2.

If a feasible collection is not dominated by another feasible collection, it is Pareto-efficient. In Figure 1, the set of Pareto-efficient solutions is $\{[2,2], [1,4], [4,1]\}$.

The following search algorithm that operates top-down along the implicitly given rank lattice finds all Pareto-efficient allocations. To generate successors in the lattice, it asks the agents what their next most preferred bundles are.

```
$s = (1, \ldots, 1)$ /* start node */
PAR = []; /* Pareto-optimal nodes */
OPEN = [$s$]; /* Unexpanded nodes */
while OPEN $\neq$ [] do
    $Remove(c, OPEN)$;   SUC = suc($c$)
    if Feasible($c$) then
        $PAR = PAR \cup \{c\}$;   $Remove(SUC, OPEN)$
    else foreach $n \in$ SUC do
        if $n \notin$ OPEN and $Undominated(n, PAR)$
          then $Append(n, OPEN)$
```

If (monetary) valuations can be asked, then those queries can be combined with rank queries to guide the search that finds a welfare-maximizing allocation.

[2]Given two rank vectors a and b, a is said to dominate b if $a_i \geq b_i$ for all agents i and $a_j > b_j$ for at least one agent j.

EXAMPLE 2. *Let there be the two goods, A and B, and the two agents, a_1, and a_2 of the above example. The agents assign the following values to the bundles:*

	$\emptyset$	A	B	AB
a_1	*0*	*4*	*3*	*8*
a_2	*0*	*1*	*6*	*9*

The values imply the preference order that has been considered in Example 1. The value-augmented rank lattice is shown in Figure 1 Right. The welfare-maximizing allocation is given by rank vector $[2,2]$*, that is* $X^* = \{A, B\}$.

The following search algorithm uses rank and value information to determine a welfare-maximizing allocation.

```
$s = (1, \ldots, 1)$ /* start node */
OPEN = $\{s\}$; /* Unexpanded nodes */
CLOSED = $\emptyset$; /* Expanded nodes */
while OPEN $\neq \emptyset$ do
    $c = \arg\max_{c \in OPEN} \sum_{i \in N} v_i(c_i)$
    OPEN = OPEN $\setminus \{c\}$
    if Feasible($c$) then return $\{c\}$
    CLOSED = CLOSED $\cup \{c\}$; SUC = suc($c$)
    foreach $n \in$ SUC do
        if $n \notin$ OPEN and $n \notin$ CLOSED
        then OPEN = OPEN $\cup \{n\}$
```

In practice, the algorithm would ask questions to determine the best rank vector in OPEN (i.e., to solve the arg max). For a given rank, the algorithm needs to know which bundle the agent associates with that rank and which value she attributes to that bundle. The algorithm traverses the rank lattice in a way that leads to a natural sequence of questions for the bidder: asking for the highest ranking bundle first, then proceeding to the next best bundle, and so on.

3.2 Policy-Independent Elicitation Algorithms

The algorithms presented so far are based on search, and the search strategy imposes constraints on the query policy (which question is asked as a function of answers received so far). We now present algorithms that avoid this weakness. Questions can be asked in any order that the auctioneer considers (*ex ante*) most efficient, and still no unnecessary (from the perspective of all the information known and derivable at that time) questions are asked.

The auctioneer may be able to ask any bidder the following types of queries:

- **Order queries:** Which bundle do you prefer, A or B?
- **Value queries:** What is your valuation for bundle A? The bidder can answer with bounds or an exact value.
- **Rank queries:** In your preferences, what is the rank of bundle A? Which bundle has rank x in your preferences? (Later we also discuss the more natural question: If you cannot get the bundles that you have declared most desirable so far, what is your most desired bundle among the remaining ones?)

3.2.1 *Augmented Order Graph*

The elicitor algorithms use a data structure called an *augmented order graph G* to assimilate the answers. It includes a node for each (bidder, bundle) pair (i, X). It includes a directed arc from node (i, X) to node (i, Y) (always nodes of the same bidder) whenever $v_i(X) > v_i(Y)$. We call this a domination arc $\succ$. The graph G also includes an upper bound UB and a lower bound LB for each node. Finally, it includes a rank $R_i(X)$ for every node. Some of these variables may not have values. Initially, G includes no edges. The upper bounds are initialized to ∞, and the lower bounds to 0 (in the free-disposal case) or to $-\infty$ (in the general case). All of the rank information is initially missing. If there is free disposal, edges are added to the graph to represent this: $((i, X), (i, Y)) \in \succ$ whenever $Y \subset X$.

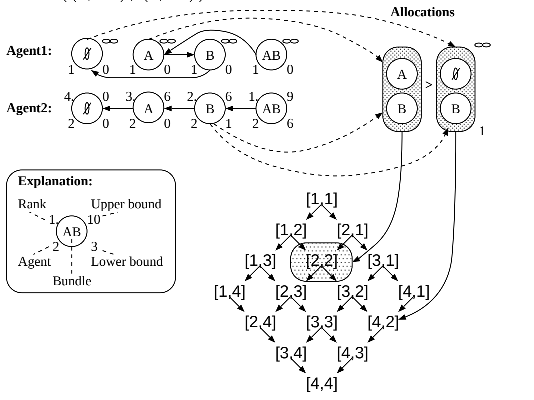

Figure 2: Order graph, feasible allocations, and how they relate to the rank lattice.

Figure 2 shows the augmented order graph for our 2-agent, 2-good example at a stage where some of the information from the bidders has not yet been asked. In the upper right corner, two allocations and their relation to the nodes in the graph are shown. These allocations are connected to the corresponding feasible collections (allocations) in the rank lattice. The lower bound LB of a collection is the sum of the lower bounds of the bundles in that collection. Similarly, the upper bound UB of a collection is the sum of the upper bounds of the bundles in that collection. In the example, the allocations can be ordered due to the available rank information. The allocation $(\{A\}, \{B\})$ dominates the other. The highlighted rank vector represents the welfare-maximizing allocation. This, however, cannot be determined yet due to lack of information.

Our algorithms use the augmented order graph as the basic analysis tool. As new information is obtained, it is incorporated into the augmented order graph. This may cause new arcs to be added, bounds to be updated, or rank information to be filled in. As a piece of information is obtained and incorporated, our algorithms fully propagate its implications. The process is monotonic in that new information allows us to make more specific inferences. Edges are never removed, upper bounds never increase, lower bounds never decrease, and rank information is never erased.

3.2.2 General Elicitation Algorithm Framework

In different settings, answering some of the query types can be more natural and easier than answering others. Therefore, we designed different algorithms that use different subsets of the query types listed above (and different query policies). The algorithms share the following general skeleton, but differ based on how the specific procedures in this skeleton work. Due to limited space, we do not present those algorithms here, but refer the reader to [2] for details. An augmented order graph G and an input set $\mathcal{Y}$ are expected as input to the algorithms. The type of input set $\mathcal{Y}$ depends on the specific algorithm.

```
Algorithm $Solve(\mathcal{Y}, G)$:
while not $Done(\mathcal{Y}, G)$ do
    $o = SelectOp(\mathcal{Y}, G)$ /* Choose question */
    $I = PerformOp(o, N)$ /* Ask bidder */
    $G = Propagate(I, G)$ /* Update graph */
    $\mathcal{Y} = Candidates(\mathcal{Y}, G)$ /* Curtail the set of
                                  candidate allocations */
```

In addition to this general structure, the algorithms share the procedures for efficiently comparing two collections for dominance and for optimally propagating value information, rank information, and order information in the augmented order graph. Due to limited space we will not present those procedures here. They are described in [2].

4. EFFICIENCY OF ELICITATION

The elicitor is economically efficient. Our algorithms that use value queries (possibly with other queries) are guaranteed to find the social welfare maximizing allocations. Our algorithms that only use order and/or rank queries are guaranteed to find the Pareto optimal allocations.

It is easy to show that the elicitor also saves revelation. Consider the following simple example. A bidder has revealed that she prefers bundle A over bundle B, and that her valuation for A is at most \$100. If, based on the bids from others, the elicitor already knows that it can obtain revenue higher than \$100 for bundle B, then the elicitor need not ask the bidder her valuation for B, because she would not win B anyway.

It is also easy to show that in the worst case, the number of queries the elicitor needs to ask to determine the optimal allocation is exponential in the number of items for sale (at least when it comes to value and order queries). Consider the following simple example with just one bidder. Say the bidder assigns a high value to some bundle, and zero value to all other bundles. The elicitor's goal of maximizing welfare amounts to trying to find the bundle that the bidder most prefers (and to prove that that bundle is better than any other). So, without any extra structure (such as knowledge of free disposal), the elicitor needs to get some information about the value of every bundle. Every value query provides information about only one bundle, and every order query provides information about only two bundles. Therefore, the number of value/order queries needed is at least half the number of bundles, which is exponential in items.

To improve the (average case) revelation efficiency, the elicitor can allow the bidders to also answer queries that were not asked (and our answer assimilation algorithms would not treat the answer any differently). This allows the bidder, who has some information (about his own valuations) that the elicitor does not have, to guide the revelation process. For example, this would solve the example of the previous paragraph. On the other hand, the elicitor also has information that the bidder does not have (about the others' valuations) so in some cases the query-directed revelation is effective—as shown in the paragraph before last.

We can also integrate the elicitation technique with open-cry ascending combinatorial auctions, where some unnecessary revelation is avoided via price feedback [10, 9, 7, 18, 8]. Namely, if the price of a bundle is already too high for an agent, the agent need not compute or communicate her *exact* valuation. On top of that, the elicitor can guide revelation, and bidders can answer queries that were not asked.

5. INCENTIVE COMPATIBLE ELICITATION

Motivating the bidders to answer queries truthfully is a key issue, and is exacerbated by the fact that the elicitor's queries leak information to the bidder about the answers that other bidders have given.

However, any of our elicitor designs can be made incentive compatible in the sense that every bidder answering the queries truthfully is a perfect Bayesian equilibrium. This is accomplished by organizing the mechanism so that if all the bidders answer truthfully, the final allocation and payments follow the *Vickrey-Clarke-Groves scheme (VCG)* [17, 1, 4]. In the VCG, the amount a bidder has to pay is the sum of others' revealed valuations for the bundles they get had the bidder not participated minus the sum of others' revealed valuations for the bundles they get in the actual allocation.

The elicitor can determine these payments by asking enough queries to be able to determine the welfare maximizing allocation overall, and *by asking extra queries to determine the welfare maximizing allocation for the auctions where each agent is ignored in turn.* Conceptually, one could think of $n+1$ "elicitors", each working to solve one of these problems. However, these "elicitors" can use the same data structure for assimilating the results, which leads to the advantage that queries answered for one "elicitor" can help another "elicitor" on its problem. Once all of the "elicitors" have found their welfare maximizing allocations respectively, the process can terminate. The notion of multiple "elicitors" is just for conceptual clarity of this presentation; in practice there would be only one elicitor asking all of the queries.

There is the risk of a lazy bidder who would not answer queries once enough have been answered to determine her allocation and payment. To deter this possibility, the mechanism could interleave, for that bidder, the questions pertinent to that bidder's allocation and VCG payment with queries pertinent to the other agents' allocations and VCG payments. This way the bidder would not know (at least not directly) which purpose the questions are for. Any order would motivate the bidder to reveal truthfully—the interleaving scheme is simply to avoid lazyness.

6. CONCLUSIONS

Combinatorial auctions require potentially every bundle to be bid on, and there are $2^m - 1$ bundles. This is complex for the bidder because she may need to invest effort or computation into determining each of her valuations. If the bidder evaluates bundles that she does not win, evaluation effort is wasted. Bidding on too many bundles can also be undesirable from the perspective of revealing unnecessary private information and from the perspective of causing unnecessary communication overhead. If the bidder omits evaluating (or bidding on) some bundles on which she would have been competitive, economic efficiency and revenue are generally compromised. A bidder could try to evaluate (more accurately) only those bundles on which she would be competitive. However, in general it is difficult for the bidder to know on which bundles she would be competitive before evaluating the bundles.

To address these problems, we presented a design of an elicitor that helps guide the revelation of information from the bidders to the auctioneer by asking relevant questions from the bidders and optimally assimilating the answers. It allows bidders to also answer queries that were not asked. It can be used in conjunction with price-feedback mechanisms to get the best of all of the (known) mechanisms for guiding revelation. We also presented a way to make the elicitor incentive compatible.

7. REFERENCES

[1] E H Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[2] Wolfram Conen and Tuomas Sandholm. Minimal preference elicitation in combinatorial auctions. In *IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms*, pages 71–80, Seattle, WA, 2001.

[3] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *IJCAI*, pages 548–553, 1999.

[4] Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[5] Kate Larson and Tuomas Sandholm. Computationally limited agents in auctions. In *AGENTS-01 Workshop of Agents for B2B*, pages 27–34, 2001.

[6] Kate Larson and Tuomas Sandholm. Costly valuation computation in auctions: Deliberation equilibrium. In *TARK*, pages 169–182, Siena, Italy, July 2001.

[7] David Parkes. iBundle: An efficient ascending price bundle auction. In *ACM-EC*, pages 148–157, 1999.

[8] David Parkes. Optimal auction design for agents with hard valuation problems. In *Agent-Mediated Electronic Commerce Workshop at the IJCAI*, 1999.

[9] David Parkes and Lyle Ungar. Iterative combinatorial auctions: Theory and practice. *AAAI*, p. 74–81, 2000.

[10] David Parkes and Lyle Ungar. Preventing strategic manipulation in iterative auctions: Proxy-agents and price-adjustment. In *AAAI*, pages 82–89, 2000.

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

[12] Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *AAAI*, pages 256–262, 1993.

[13] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *IJCAI*, pages 542–547, 1999.

[14] Tuomas Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000. Special Issue on Applying Intelligent Agents for Electronic Commerce. Short version at ICMAS, pages 299–306, 1996.

[15] Tuomas Sandholm and Subhash Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *AAAI*, pages 90–97, 2000.

[16] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. *IJCAI*, pages 1102–1108, 2001.

[17] W Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *J Finance*, 16:8–37, 1961.

[18] Peter R Wurman and Michael P Wellman. AkBA: A progressive, anonymous-price combinatorial auction. In *ACM-EC*, pages 21–29, 2000.