Minimal Preference Elicitation in Combinatorial Auctions

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.

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 [10, 11, 2, 13, 15, 16, 14].

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, 4, 5, 7]. It can also be undesirable from the perspective of revealing unnecessary private information and from the perspective of unnecessary communication.

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). We present building blocks for a design of an auctioneer agent that interrogates the bidder intelligently regarding the bidder's preferences, and optimally assimilates the returned information. 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, and then move to the data structures that are used to store and propagate the information received. After that we discuss the algorithms for interrogating.

2 The Combinatorial Auction Setting

In a combinatorial auction, the seller has a set $\Omega = \{\omega_1, \ldots, \omega_m\}$ of indivisible, distinguishable items that she can sell. Any subset of the items is called a *bundle*. The set of bidders (buyers) is called $N = \{1, \ldots, n\}$.[1] Each bidder has a value function $v_i : 2^\Omega \rightarrow \mathbf{R}$ that states the value that the bidder is willing to pay for any given bundle. We assume that the valuations are unique ($v_i(X) \neq v_j(Y)$ if $i \neq j$ or $X \neq Y$). This an innocuous assumption since if the valuations are drawn from reals, the probability of a tie is zero.

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

[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.

Our definitions of desirability of the solution (welfare maximization and Pareto efficiency, discussed later), are defined with respect to the *reported* valuations. There is a large literature in game theory that deals with the issue of how to design the incentives (rules of the game, who gets which bundles, who pays what) so as to motivate each bidder to reveal her preferences truthfully. Later in this paper we will discuss the incentives for truthful revelation within our elicitation method.

We make the standard quasilinearity assumption about the bidder's utilities, so the utility of any bidder i for bundle $A \subseteq \Omega$ is $u_i(A, p) = v_i(A) - p$, where p is the amount that the bidder has to pay.

A *collection* $(X_1, \ldots, X_n)$ 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 and *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). An allocation X is *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.[2]

3 Topological Structure in Combinatorial Auctions

We observed that there is significant topological structure in the combinatorial auction setting. In this paper we will use that structure to avoid asking the bidders for unnecessary information about their valuations. In this section, we first discuss the topological structure in the context of an implicit graph which we call the *rank lattice*. We then present an actual data structure called the *augmented order graph* where the auctioneer stores all of the known information about the bidders' valuations, and propagates it using topological structure in several ways.

3.1 Rank Lattice

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" relation between them define a lattice. Now, the important fact is that if a collection (rank vector) is feasible (i.e., is an allocation), then no collection (rank vector) "below" it can be a better solution to the allocation problem. Consider the following example.

Example: Let there be two goods, A and B, and two agents, a_1, and a_2. 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

These values imply a preference order over the bundles:
Agent a_1: $(1 : AB, 2 : A, 3 : B, 4 : \emptyset)$.
Agent a_2: $(1 : AB, 2 : B, 3 : A, 4 : \emptyset)$.

Only a subset of the collections is feasible and, thus, corresponds to allocations (see Figure 1, left).

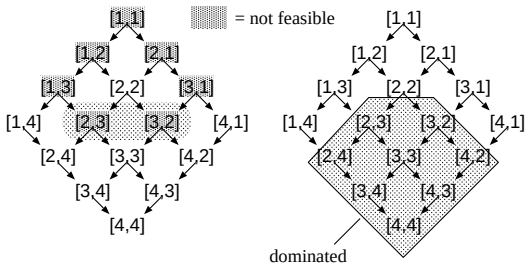

Figure 1: Rank lattice for our example. The nodes are collections. Some of them are infeasible, some are dominated, some are both, and some are neither.

[2] Our definition of Pareto efficiency is based on comparison of bundles within an agent. If payments are taken into account in the definition of Pareto efficiency, then the set of Pareto efficient solutions collapses to equal the set of welfare maximizing solution. We use the term *welfare maximizing* for the latter set, and reserve the term *Pareto efficient* for the former purpose.

If a feasible collection is not dominated by another feasible collection, it belongs to the set of Pareto-efficient solutions. In Figure 1, the set of Pareto efficient solutions can be obtained from overlaying the two graphs and picking the uncolored collections $[2,2]$, $[1,4]$, and $[4,1]$. Now, the welfare maximizing allocations can be determined by determining the value of each allocation in this set. In our example, the welfare maximizing allocation is given by rank vector $[2,2]$, that is $X^* = \{A, B\}$.

With additional knowledge about the valuation functions, the collections to be checked for feasibility can be further restricted. For example, in many auctions there is *free-disposal* ($v_i(X) \leq v_i(Y)$ if $X \subseteq Y$, i.e., the bidder can always throw away extra items for free). Free disposal combined with our assumption of unique valuations allows for the following type of pruning. It is never necessary to check the feasibility of collections that include one of the highest ranking bundles (a rank 1) and a non-empty bundle (the highest ranking bundle is always the bundle including all items, every such collection must be infeasible). Free disposal can be used to accomplish much more pruning in general. However, our techniques do not assume free disposal; they capitalize on it if it holds.

While our algorithms never explicitly construct the rank lattice, they capitalize on all of the information that can be deduced from it. Our algorithms also capitalize on other information and structure, as discussed in the next section.

3.2 Augmented Order Graph

The *augmented order graph* G 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 for each node 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. All of the rank information is initially missing. If there is free disposal, edges are added to the graph to represent this: $((i, X), (i, Y)) \in \succ$ iff $Y \subset X$ and $X \neq Y$.

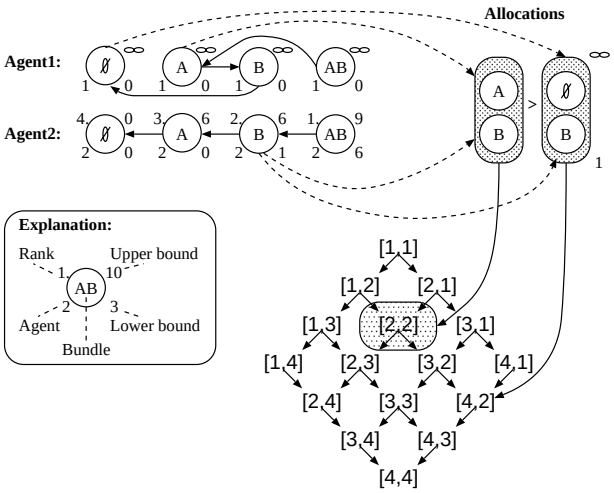

Figure 2: Order graph, feasible allocations, and how they relate to the rank lattice

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 an allocation is the sum of the lower bounds of the bundles in that allocation. Similarly, the upper bound UB of an allocation is the sum of the upper bounds of the bundles in that allocation. In the example, the allocations can be ordered due to the available rank information. The allocation $(\{A\}, \{B\})$ dominates the other. The highlighted rank combination 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 causes new arcs to be added, bounds to be updated, and rank information to be filled in. As a piece of information is obtained and incorporated, its implications are fully propagated, as will be discussed.

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.

4 Interactive Algorithms for Selecting an Allocation

In this section we present algorithms that find desirable allocations based on asking the bidders questions. The idea is to use this as a blueprint for implementing a software agent 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, propagates the implications of the answer, and does this again until it has received enough information. The auctioneer is allowed to ask any bidder i any of the following questions:

- **Order information:** Which bundle do you prefer, A or B?
- **Value information:** What is your valuation for bundle A? (the bidder can answer with bounds or an exact value).
- **Rank information:** 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?)

In different settings, answering some of these questions might be more natural and easier than answering others. Therefore, we present different algorithms that use only some of these types of questions.

We first present *policy independent algorithms* that make a clear separation between the policy of which questions to ask and the assimilation of the information obtained. The advantage is that our optimal assimilation techniques can then be used with any interrogation policy. The disadvantage is that the techniques are intense in terms of computation time and space if the number of (bidder, bundle) pairs is large. Later we present *policy driven algorithms* that tightly integrate the interrogation and assimilation for computational efficiency.

4.1 Policy Independent Algorithms

All of the policy independent algorithms utilize the same general structure. An augmented order graph G and an input set $\mathcal{Y}$ are expected as input to the algorithm. The type of the input set $\mathcal{Y}$ will depend on the specific algorithm. The general skeleton algorithm is as follows.

Algorithm *Solve*$(\mathcal{Y}, G)$:
while not *Done*$(\mathcal{Y}, G)$ **do**
  $o =$ *SelectOp*$(\mathcal{Y}, G)$ /* Choose questions */
  $I =$ *PerformOp*(o, N) /* Interrogate bidders */
  $G =$ *Propagate*(I, G) /* Update graph */
  $U =$ *Candidates*$(\mathcal{Y}, G)$ /* Select candidate allocations */

Given two allocations, a and b, and the augmented order graph G, the following procedure will be used in the policy independent algorithms to check whether a dominates b. This is determined using a combination of value information, order information, and rank information (queried and inferred).[3]

Algorithm *Dominates*(a, b, G):
$O_{ab} = FALSE$ /* Flag for order domination */
$C_{ab} = 0$ /* Amount of value domination */
foreach $i \in N$ **do**
  if $LB_i^a > UB_i^b$
    then $C_{ab} = C_{ab} + (LB_i^a - UB_i^b)$
    else if $a_i \succ b_i$
      then $O_{ab} = TRUE$
      else $C_{ab} = C_{ab} + (LB_i^a - UB_i^b)$
if $C_{ab} > 0$ or ($C_{ab} = 0$ and $O_{ab} = TRUE$)
  then return TRUE **else** return FALSE

[3] The *Dominates* procedure does not explicitly use rank information because the implications of the rank information will have already been propagated into the value information in the bounds and the order information.

Proposition 1. *Given a consistent order graph G, the* Dominates *algorithm returns TRUE if and only if enough information has been queried to determine that a dominates b.*

Next, propagating newly received information will be discussed. If value or order information is inserted into a previously consistent graph G, values of upper bounds are propagated in the direction of the edges and lower bounds in the opposite direction. This propagation is done via depth-first-search (that marks the nodes touched when they are visited) in G, so the propagation time is $O(v + e)$, where v is the number of bundles (number of nodes in G that correspond to the agent whose values are getting updated), and e is the number of edges between those nodes.

Case 1: Inserting a new lower bound at node k:
Procedure *PropLB*(k, G) /* G contains the set of edges, $\succ$ */
$Pre = \{l : (l, k) \in \succ\}$
foreach $l \in Pre$ **do**
  if $LB_k > LB_l$ **then** $LB_l = LB_k$; *PropLB*(l, G)

Case 2: Inserting a new upper bound at node k:
Procedure *PropUB*(k, G)
$Suc = \{l : (k, l) \in \succ\}$
foreach $l \in Suc$ **do**
  if $UB_k < UB_l$ **then** $UB_l = UB_k$; *PropUB*(l, G)

Case 3: Inserting a new edge $k \succ l$:
Procedure *InsertEdge*$((k, l), G)$
if $LB_k < LB_l$ **then** $LB_k = LB_l$; *PropLB*(k, G)
if $UB_k < UB_l$ **then** $UB_l = UB_k$; *PropUB*(l, G).

Case 4: Inserting an exact valuation for node k:
Procedure *InsertValue*$((k, v), G)$
$LB_k = v$; *PropLB*(k, G); $UB_k = v$; *PropUB*(k, G).

Case 5: Inserting a rank for node k:
Procedure *InsertRank*$((n, r), G)$
$(i, b) = n$; $K = \{(j, c) \in V : j = i\}$
if $\exists k \in K$ with $R_k < R_n$ and
  $R_k \geq R_l \; \forall l \in K$ with $R_l < R_n$
  then *InsertEdge*((k,n),G)
if $\exists k \in K$ with $R_k > R_n$ and
  $R_k \leq R_l \; \forall l \in K$ with $R_l > R_n$
  then *InsertEdge*((n,k),G)

Given a set of newly retrieved information, I, and the augmented order graph G, the following algorithm will insert the information and propagate it.
Algorithm *Propagate*(I, G):
foreach $i \in I$ **do**
  switch i /* Structural switch */
    (node k, UB b):
      if $UB_k > b$ **then** $UB_k = b$; *PropUB*(k,G)
    (node k, LB b):
      if $LB_k < b$ **then** $LB_k = b$; *PropLB*(k,G)
    (node k, node l): *InsertEdge*$((k, l), G)$
    (node k, value v): *InsertValue*$((k, v), G)$
    (node k, rank r): *InsertRank*$((k, r), G)$

The following subsections discuss the policy independent algorithms in detail. The algorithms differ based on the types of information that they request from the bidders. To obtain the complete algorithm, the following procedures have to be substituted for the corresponding procedures in the general structure above.

4.1.1 Algorithms that Use Order Information

Order information allows the Pareto-optimal allocations to be determined, but it cannot be used to determine welfare maximizing allocations because that would require quantitative tradeoffs across agents.

Given a non-empty set $\mathcal{Y}$ of feasible allocations and a graph G, the following algorithm will determine the set of allocations that are not dominated, given the information in hand.

Algorithm *Candidates*$_o(\mathcal{Y}, G)$:
$U = \mathcal{Y}$; $O = \emptyset$; $C = \emptyset$
while $U \neq \emptyset$ do
  pick $a \in U$; $U = U \backslash \{a\}$; $C = \emptyset$
  while $U \neq \emptyset$ do
    pick $b \in U$; $U = U \backslash \{b\}$
    if *Dominates*(b, a, G)
      then $a = b$
      else if not *Dominates*(a, b, G)
        then $C = C \cup \{b\}$
  $U = C$; $O = O \cup \{a\}$

Proposition 2. *The above algorithm determines the set $O \subset \mathcal{Y}$ such that for all $a \in O : \nexists b \in \mathcal{Y}$ with b dominates a.*

Given a set of allocations, $\mathcal{Y}$, which are all pairwise incomparable with respect to domination, and the graph G, the following function will return FALSE if a pair of allocations exists in $\mathcal{Y}$ which have been judged incomparable due to lack of information.

Function *Done*$_o(U, G)$
foreach $\{a, b\} \in U \times U, a \neq b$ **do**
  if not *DefinitelyIncomparable*[4] (a, b)
    then if $\exists i \in N : (i, a_i), (i, b_i) \notin \succ$
      and $((i, b_i), (i, a_i)) \notin \succ$
      then return FALSE
return TRUE

The remaining part of the algorithm is the interrogation policy. Our method can accommodate any policy here, but we suggest two intuitive ones:

(1) Arbitrarily pick a pair of distinct allocations $\{a, b\}$ that are incomparable due to a lack of information. Arbitrarily, choose one of the agents $i \in N$, for which no order information for the corresponding bundles a_i and b_i is available. Ask the bidder which one of them she prefers.[5]

(2) Determine the set of pairs of incomparable allocations, U. While doing so, determine a set P of pairs of unordered nodes $\{(i, a_i), (i, b_i)\} \in G$ for which $\exists a, b \in U, a \neq b$ so that neither $((i, a_i), (i, b_i)) \in \succ$ nor $((i, b_i), (i, a_i)) \in \succ$. Select from P a pair $p = \{(i, b_1), (i, b_2)\}$ of nodes so that the number of pairs in U which contain p is maximal.[6] Ask the bidder which bundle she prefers more, b_1 or b_2.

Proposition 3. *Given a set of allocations, $\mathcal{Y}$, and the graph G, for either interrogation policy,* Solve$_o(\mathcal{Y}, G)$ *will determine the set of Pareto-optimal allocations contained in $\mathcal{Y}$.*

4.1.2 Algorithms that Use Order and Value Information

Here we present an algorithm to determine welfare maximizing solutions. Given a non-empty set of feasible allocations, $\mathcal{Y}$, that are all pairwise incomparable with respect to domination, and an augmented order graph G, the following algorithm checks if $\mathcal{Y}$ contains only welfare maximizing allocations.

[4] A pair $\{a, b\}$ of allocations will be called *definitely incomparable*, iff there is a pair of agents, $\{i_1, i_2\}$ such that edges $(i_1, a_{i_1}) \succ (i_1, b_{i_1})$ and $(i_2, b_{i_2}) \succ (i_2, a_{i_2})$ and no edges $(i_1, b_{i_1}) \succ (i_1, a_{i_1})$ or $(i_2, a_{i_2}) \succ (i_2, b_{i_2})$ exist.

[5] The answer to this question alone might not be sufficient to order a and b since there may be other unordered bundles in those allocations. Also, this question might not be necessary: it can be possible to deduce the answer from answers to other alternative questions. On the positive side, the answer to this question may make asking some other questions unnecessary.

[6] Deciding this edge adds information to the largest number of decisions in the next stage.

Algorithm *Done*$_v(\mathcal{Y}, G)$:
if $|\mathcal{Y}| = 1$ **then return** TRUE
foreach $a \in \mathcal{Y}$ **do**
$lb = \sum_{n \in G_a} LB_n$
$ub = \sum_{n \in G_a} UB_n$
if $lb \neq ub$ **then return** FALSE
return TRUE

Candidates$_v$ is identical with *Candidates*$_o$.

Again, any interrogation policy could be used. We propose the following. Pick a node $n = (i, b) \in V$ with $LB_n \neq GB_n$ and (one of) the largest number(s) of relations to allocations in $\mathcal{Y}$ and $LB_n \neq GB_n$ with the greatest lower bound. Ask agent i for the value of bundle b.

Proposition 4. *Given a set of feasible allocations $\mathcal{Y}$ and a graph G, the algorithm* Solves$_v$ *will determine the set of welfare maximizing allocations contained in $\mathcal{Y}$.*

4.1.3 Algorithms that Use Rank Information

Algorithms that use rank information only cannot determine welfare maximizing solutions because that requires quantitative tradeoffs across agents. Instead, we present an algorithm for finding Pareto-efficient ones.

Given G and a set $\mathcal{C}$ of rank combinations, the following function answers TRUE if all elements of $\mathcal{C}$ are feasible.
Function *Done*$_r(\mathcal{C}, G)$
foreach $c \in \mathcal{C}$ **do**
if $\exists i \in N : \nexists (i, b) \in V$ with $R_{(i,b)} = c_i$
then return FALSE /* Information missing */
if not *Feasible*[7](c, G)
then return FALSE **else return** TRUE

Function *Candidates*$_r(\mathcal{C}, G)$:
foreach $c \in \mathcal{C}$ **do**
if *Infeasible*[8](c, G)
then $\mathcal{C} =$ *Expand*$(c, \mathcal{C}, G)$
/* Loops over newly inserted elements */
Function *Expand*(c, C, G)
$S = \mathrm{suc}^9(c)$; $C = C \backslash \{c\}$;
foreach $s \in S$ **do**
if not *IsDominated*(s, C, G)
then $C = C \cup \{s\}$

Function *IsDominated*(s, C, G): **foreach** $c \in C$ **do**
if $c \leq s$ /* lexicographic */
then if not *Infeasible*(c, G) **return** TRUE
return FALSE

We propose two interrogation policies for *SelectOp*$_r(\mathcal{C}, G)$: (1) Select from $\mathcal{C}$ a combination c with the least number of ranks without related nodes in G. For each such rank r at position i of c, ask bidder i which bundle she has at rank r. (2) As (1), but pick only one rank without a related node from c.

Proposition 5. *Given a set of combinations, $\mathcal{C}$, and the graph G, for both policies, algorithm* Solve$_r(\mathcal{C}, G)$ *will determine the set of feasible combinations in the (partial) lattice determined by $\mathcal{C}$ that are either in $\mathcal{C}$ or dominated only by infeasible combinations in $\mathcal{C}$. If $\mathcal{C}$ is initialized to $(1, \ldots, 1)$, the resulting set represents the set of Pareto-optimal allocations.*

[7] c is feasible if b^c, the corresponding set of bundles, is a partition of a subset of Ω. If not all bundles related to ranks are known yet, a FALSE will be returned.

[8] Infeasibility can often be determined without knowing all rank-bundle relations. If the partial information available is not sufficient to decide about infeasibility, FALSE will be returned. Thus, if both functions return FALSE, the information is insufficient.

[9] The successor function derives from a node $c = (r_1, \ldots, r_n)$ its set of successors as follows. For each $i, 1 \leq i \leq n$ with $r_i < 2^m$, generate $s_i \in \mathrm{suc}(c)$ as $(r_1, \ldots, r_i + 1, \ldots, r_n)$.

4.1.4 Algorithms that Use Rank and Value Information

In this section we discuss how rank and value information can be used to determine welfare maximizing solutions. The Candidates and Done functions are instantiated as follows to do this.
Function *Candidates*$_{rv}(\mathcal{C}, G)$:
$c = \arg\ \max_{d \in \mathcal{C}}$ *LB*[10](d, G)
if $\nexists d \in \mathcal{C} \backslash \{c\}$ with *UB*(d, G) >*LB*(c, G)
then $\mathcal{C}$ =*Expand*(c, C, G)

Function *Done*$_{rv}(\mathcal{C}, G)$
$c = \arg\ \max_{d \in \mathcal{C}}$ *LB*(d, G) /* Best valued node */
if $\exists d \in \mathcal{C} \backslash \{c\}$ with *UB*(d, G) >*LB*(c, G) and
not *Infeasible(d,G)*
then return FALSE **else return** TRUE
We propose the following interrogation policy for *SelectOp*$_{rv}(\mathcal{C}, G)$. Pick the undecided combination with the highest lower bound, say c. Pick from the remaining combinations one, say d, with $UP_d > LB_c$. If there is a rank i in d without a related node, ask agent i which bundles she ranks at rank r. Otherwise, if there is a rank i and a corresponding node (i, b) with $UB_(i, b) \neq LB_(i, b)$, ask agent i her value for bundle b. Otherwise, there is a rank j in c and a corresponding node (j, a) with $UB_(j, a) \neq LB_(j, a)$. Ask agent i her value of bundle a.

Proposition 6. *Given a set of combinations, $\mathcal{C}$, and the graph G, algorithm* Solve$_{rv}(\mathcal{C}, G)$ *will determine the set of feasible combinations in the (partial) lattice determined by $\mathcal{C}$ that are not dominated by other feasible combinations in the sublattice. If $\mathcal{C}$ is initialized to* $(1, \ldots, 1)$*, the resulting set represents the welfare maximizing allocations.*

4.2 Policy-Driven Algorithms

While the algorithms presented above are very flexible in terms of their interrogation policies, tying the policy more closely to the algorithm allows for improvements in computational efficiency, and also allows for the use of standard search strategies in these interactive algorithms.

4.2.1 Algorithm that Uses Rank and Value Information

The following search algorithm uses rank and value information to determine a welfare maximizing allocation.
Algorithm *BBF*:
$s = (1, \ldots, 1)$ /* start node */
OPEN $= \{s\}$; /* Unexpanded nodes, initialized to s */
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 $\backslash \{c\}$
if *Feasible(c)* **then return** $\{c\}$
CLOSED = CLOSED $\cup \{c\}$; SUC $= \mathrm{suc}(c)$
foreach $n \in$ SUC **do**
if $n \notin$ OPEN and $n \notin$ CLOSED
then OPEN = OPEN $\cup \{n\}$

The algorithm asks questions to determine the $\arg\max$. There are the obvious value questions first, and then the search proceeds one step down a path in the rank lattice. This entails asking all winning bidders the following question which is more natural than an unconstrained rank question: if you cannot get any one of the bundles that you have named desirable so far, what is your next preferred bundle?

Proposition 7. *The algorithm* BBF *determines a welfare maximizing allocation.*

[10] $LB_d = \sum_{i \in N} B(i, d_i)$, where $B(i, d_i)$ is $LB(i, b)$ if $\exists (i, b) \in V$ with $R_{(i,b)} = d_i$, and 0 otherwise. UB_d analogously, with ∞ instead of 0.

5 Conclusions and Future Research

Combinatorial auctions 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, they 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. 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 (and 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.

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' valuations while still finding the optimal (welfare maximizing and/or Pareto efficient) solution(s). We presented the rank lattice as an analysis tool. We then designed a data structure for storing and propagating all of the information that the auctioneer agent has received. Based on the information, the agent can narrow down the set of potentially desirable allocations, and intelligently decide which questions to ask the bidders next.

Three types of information queries were considered: value information (potentially with bounds only), order information, and rank information (arbitrarily or in order). Selective interrogation algorithms were presented that use different combinations of these to provably find the desired solution(s). Some of the algorithms focused on the propagation of information, and would support any interrogation policy. Finally, we presented a search-based algorithm that deeply integrates the interrogation policy into the algorithm in order to use a standard search strategy for interrogation, and in order to not have to use and store the elaborate data structure of the policy independent algorithms.

We believe that this line of research holds significant promise. We plan to integrate it with incentive compatible auction mechanisms such as the generalized Vickrey auction (GVA) [17, 1, 3]. The idea is that the our preference elicitation method would find a welfare maximizing solution, but would ask extra questions to be able to find the welfare maximizing solution under the assumption that each bidder in turn were not participating in the auction. These answers would suffice to compute the Vickrey payments, which would motivate the bidders to bid truthfully. Any of the algorithms of this paper could be used for this purpose by simply ignoring every bidder's bids in turn and asking the ensuing questions for determining the welfare maximizing allocation. If there are lazy bidders that would not participate once their bundles and prices have been determined, the mechanism could interleave the questions pertinent to GVA amidst questions for determining the overall allocation. This way the bidders would not know (at least not directly) which purpose the questions are for. The only open issue to deal with is the concern that the questions that the auctioneer asks a bidder leak information to the bidder about the answers that the other bidders have submitted so far. This makes the auction format not entirely sealed-bid. Since the GVA was originally designed for sealed-bid auctions, it is not totally obvious that it leads to an incentive compatible mechanism when used in conjunction with our preference elicitation method.

Another interesting avenue is to integrate this selective revelation technique with open-cry ascending combinatorial auctions, where some unnecessary revelation is avoided in another way [9, 8, 6, 18, 7]. Namely, if the price of a bundle is already too high for an agent for sure, the agent need not compute or communicate the exact value. Combining that idea with the topological ideas of this paper is likely to lead to a hybrid protocol that requires less information than either method alone.

References

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

[2] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 548–553, Stockholm, Sweden, August 1999.

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

[4] Kate Larson and Tuomas Sandholm. Computationally limited agents in auctions. In *AGENTS-01 Workshop of Agents for B2B*, Montreal, Canada, 2001.

[5] Kate Larson and Tuomas Sandholm. Costly valuation computation in auctions: Deliberation equilibrium. In *Theoretical Aspects of Reasoning about Knowledge (TARK)*, Siena, Italy, 2001.

[6] David C Parkes. iBundle: An efficient ascending price bundle auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 148–157, Denver, CO, November 1999.

[7] David C Parkes. Optimal auction design for agents with hard valuation problems. In *Agent-Mediated Electronic Commerce Workshop at the International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

[8] David C Parkes and Lyle Ungar. Iterative combinatorial auctions: Theory and practice. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 74–81, Austin, TX, August 2000.

[9] David C Parkes and Lyle Ungar. Preventing strategic manipulation in iterative auctions: Proxy-agents and price-adjustment. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 82–89, Austin, TX, August 2000.

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

[11] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 542–547, Stockholm, Sweden, 1999. Extended version first appeared as Washington Univ., Dept. of Computer Science, tech report WUCS-99-01, January 28th.

[12] 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. A short, early version appeared at the Second International Conference on Multi–Agent Systems, pages 299–306, 1996.

[13] Tuomas Sandholm and Subhash Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 90–97, Austin, TX, 2000.

[14] Tuomas Sandholm and Subhash Suri. Side constraints and non-price attributes in combinatorial markets. In *IJCAI-2001 Workshop on Distributed Constraint Reasoning*, Seattle, WA, 2001.

[15] Tuomas Sandholm, Subhash 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 (IJCAI)*, Seattle, WA, 2001.

[16] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *AGENTS-2001 Workshop on Agent-Based Approaches to B2B*, Montreal, Canada, 2001.

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

[18] Peter R Wurman and Michael P Wellman. AkBA: A progressive, anonymous-price combinatorial auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 21–29, Minneapolis, MN, October 2000.