Partial-revelation VCG mechanism for combinatorial auctions

Wolfram Conen
XONAR GmbH
Wodanstr. 7
42555 Velbert, Germany
conen@gmx.de

Tuomas Sandholm
Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213
sandholm@cs.cmu.edu

Abstract

Winner determination in combinatorial auctions has received significant interest in the AI community in the last 3 years. Another difficult problem in combinatorial auctions is that of eliciting the bidders' preferences. We introduce a progressive, partial-revelation mechanism that determines an efficient allocation and the Vickrey payments. The mechanism is based on a family of algorithms that explore the natural lattice structure of the bidders' combined preferences. The mechanism elicits utilities in a natural sequence, and aims at keeping the amount of elicited information and the effort to compute the information minimal. We present analytical results on the amount of elicitation. We show that no value-querying algorithm that is constrained to querying feasible bundles can save more elicitation than one of our algorithms. We also show that one of our algorithms can determine the Vickrey payments as a costless by-product of determining an optimal allocation.

Introduction

Combinatorial auctions, where agents can submit bids on *bundles* of items, are economically efficient mechanisms for selling m items to bidders, and are attractive when the bidders' valuations on bundles exhibit *complementarity* (a bundle of items is worth more than the sum of its parts) and/or *substitutability* (a bundle is worth less than the sum of its parts). Determining the winners in such auctions is a complex optimization problem that has recently received considerable attention (e.g., (Rothkopf, Pekeč, & Harstad 1998; Sandholm 2002; Fujishima, Leyton-Brown, & Shoham 1999; Sandholm & Suri 2000; Nisan 2000; Andersson, Tenhunen, & Ygge 2000; Sandholm *et al.* 2001)).

An equally important problem, which has received much less attention, is that of bidding. There are $2^m - 1$ bundles, and each agent may need to bid on all of them to fully express its preferences. This can be undesirable for any of several reasons: (1a) determining one's valuation for any given bundle can be computationally intractable (Sandholm 1993; 2000b; Parkes 1999; Larson & Sandholm 2001); (1b) there is a huge number of bundles to evaluate; (2) communicating the bids can incur prohibitive overhead (e.g., network traffic); and (3) agents may prefer not to reveal all of their val-

uation information due to reasons of privacy or long-term competitiveness. Appropriate bidding languages (Sandholm 2002; Fujishima, Leyton-Brown, & Shoham 1999; Sandholm 2000a; Nisan 2000; Hoos & Boutilier 2001) can potentially solve the communication overhead in some cases (when the bidder's utility function is compressible). However, they still require the agents to completely determine and transmit their valuation functions and as such do not solve all the issues. So in practice, when the number of items for sale is even moderate, the bidders will not bid on all bundles.

We study the situation in which a benevolent auctioneer (or arbitrator) wants to implement an efficient allocation of a set of heterogeneous, indivisible goods. The preferences of the participating agents (or consumers) are private information. The auctioneer tries to design a mechanism that gives no incentive for the bidders to misreport preferences.

It is well known that a Generalized Vickrey Auction (GVA), that is based on the elicitation of all utilities, is such an incentive-compatible mechanism. However, in that mechanism, each bidder evaluates each of the exponentially many bundles, and communicates a value for each one.[1] This is clearly impractical even for auctions with moderate numbers of goods.

Another important thread tries to identify iterative or progressive auction protocols that try to limit the space of preferences that are to be revealed in comparison to the fully revealing, naive GVA. Recently, auctions that follow certain solution procedures for primal/dual linear programs have been studied extensively and with respect to incentive-compatibility (see (Bikhchandani *et al.* 2001) for an overview and new suggestions). Another approach (AkBA) has been suggested (Wurman & Wellman 2000), though a detailed analysis of incentive-compatibility properties has yet to be performed. Iterative auctions are price-based, which requires that to guarantee that Vickrey payments are determined by the auction, prices must be computable that coincide with the Vickrey payments. Depending on the allowed price structures[2], equilibrium prices (that solve an underlying dual model) may or may not exist. The

---

[1]In general, preference communication in combinatorial auctions is provably exponential (even to find an approximately optimal solution) in the theoretical worst case (Nisan & Segal 2002).

[2]Price for a bundle additive in the prices of the contained goods,

existence depends on properties of preferences which will be considered either individually (e.g. submodularity) or, more general, with respect to the combination of agent types ("agents are substitutes", (Bikhchandani & Ostroy 2001)). The allowed price structure will also influence the applicability of the suggested mechanisms. For example, the unconstrained anonymous prices used in AkBA, see (Wurman & Wellman 2000), may require an enforcement of the condition that each agent is only allowed to purchase one bundle in one transaction (at the price quoted for that bundle and not, for example, in two transactions as may seem attractive if the sum of prices for sub-bundles is below the quoted price). Similar considerations are necessary for the unconstrained non-linear (and non-anonymous) prices used in auctions based on and extending the work of Bikhchandani et. al (see, for example, (Bikhchandani & Ostroy 2001; Parkes & Ungar 2000a; 2000b; Bikhchandani *et al.* 2001)).

We develop a partially-revealing, direct mechanism that does not exhibit the disadvantages of a GVA. It may save bidders from specifying or considering valuations for every bundle and, with respect to the existence of a protocol that establishes an incentive-compatible outcome and with respect to the burden put on necessary communication between consumers and auctioneer and computation of valuations on the side of the consumers, it exhibits certain limit properties that may allow to compare its attractiveness to those of iterative auctions in the general setting.

The basic idea is as follows: preferences can be elicited in a natural sequence, from most preferred towards least preferred. Combinations of individual preferences determine collections of preferred bundles. The individual preferences induce a partial *dominance* order on these collections. This relation can be exploited to guide a search for the best feasible collection through the space of infeasible collections with higher aggregated valuation.

We will present two algorithms to compute efficient allocations and an extension to determine Vickrey payments without requiring additional information. An efficient, incentive-compatible mechanism based on these algorithms will be described. We will consider aspects of the optimality of the algorithms and derive two results that bound the informational requirements for a certain class of algorithms.

The model

Our basic setting extends the concepts introduced in (Conen & Sandholm 2001). We consider a combinational economy consisting of n consumers, $N = \{1, \ldots, n\}$, a seller 0, and m goods, $\Omega = \{1, \ldots, m\}$. Each consumer $i \in N$ has utility over bundles, given by a function $u_i : 2^\Omega \to \mathbb{Q}_0^+$, where $u_i(\emptyset) = 0$. (For now, we will neither assume quasi-linearity nor monotony, because our first result does not rely on such assumptions.) The seller has all the goods; the consumers

---

unconstrained, non-linear prices for every bundle, coherent prices for bundles (the price for a bundle may not exceed the sum of prices of the bundles in any partition of the bundle, the prices for super bundles of the bundles in the supported allocation are additive); anonymous prices, different prices for the set of buyers and the set of sellers, different prices for each individual.

own none of them. We will neglect the seller for now. If the seller has reservation values (for bundles), he can be modeled as an additional consumer.

It is well known that an agent's preferences can be represented by a utility function only if the preference order is *rational*, that is, the preference order over alternatives is transitive and defined on all pairs of alternatives (equal preference between alternatives is fine). This preference order induces a rank function as follows.

Definition 1 (Rank function, Inverted Rank function). *Let R be the set of the first 2^m natural numbers, $\{1, \ldots, 2^m\}$. Let, for every agent i, the rational preference order $\succsim_i$ be defined over 2^Ω. A bijective function $R_i : 2^\Omega \to R$ will be called the rank function for agent i if it assigns a unique value (rank) to each bundle, such that, for every pair $x, y \subseteq \Omega$ of bundles with $x \succ_i y$, $R_i(x) < R_i(y)$ holds. The inverse R_i^{-1} of R_i gives the bundle that corresponds to a rank.*

Proposition 2. *A rank function and its inverse exist for every rational preference relation.*

A rank function is not necessarily unique. Indifferences in the preference order are arbitrarily resolved in the above definition, as the following example demonstrates.

Example 1. *Let the set of goods be $\Omega = \{A, B\}$ and let the preference order of agent i be*

$$\succsim_i : \{AB\} \succ \{A\} \sim \{B\} \succ \{\emptyset\}.$$

Only the following bijective functions are rank functions:

$$R_i^1 : \{AB\} \to 1, \{A\} \to 2, \{B\} \to 3, \{\emptyset\} \to 4$$
$$R_i^2 : \{AB\} \to 1, \{B\} \to 2, \{A\} \to 3, \{\emptyset\} \to 4$$

Now, a combination of ranks of the agents can be viewed as representing a potential solution to the allocation problem at hand. Some of these potential solutions are invalid, though. The others determine *allocations*.

Definition 3 (Combination of Ranks). *Let C be the set of all possible n-ary tuples over R, that is $C = R \times \cdots \times R$. An element $c \in C$, $c = (r_1, \ldots, r_n)$ will be called combination of ranks. (If, for every position i of c, the corresponding function R_i^{-1} is applied, a collection of bundles, b^c, will be obtained.)*

Definition 4 (Feasible Combination). *A combination c of ranks is feasible if no item is allocated more than once. Formally, a combination c of ranks is feasible if the corresponding collection b^c of bundles is a partition of a (not necessarily proper) subset of Ω. A feasible combination determines an allocation X^c with $X_i = b^c[i]$, $i = 1, \ldots, n$ and the seller keeps the item set $X_0 = \Omega / \bigcup_i b^c[i]$. Here, $[i]$ denotes the i'th element of a tuple.*

Definition 5 (Dominance of Rank Combinations). *A binary relation $\succeq \subseteq C \times C$ will be called a dominance relation if, for all $x, y \in C$, $(x, y) \in \succeq$ if and only if $x[i] <= y[i]$ for all $i \in N$.*

Proposition 6 (Rank Lattice). *$\succeq$ is a partial order and C forms a complete lattice with respect to $\succeq$ with $lub(C) = (n, \ldots, n)$ and $glb(C) = (1, \ldots, 1)$.*

The observation exploited by the following algorithms is that a combination that is feasible and not dominated by any other feasible combination will be Pareto-efficient. In addition, if the context is that of transferable utility, the welfare-maximizing allocation will be among these feasible Pareto-efficient combinations. The following algorithms will search the space of combinations (including infeasible ones) to determine the efficient allocations.

Efficient allocations

The first algorithm finds all Pareto-efficient allocations. We proposed it in (Conen & Sandholm 2001); we restate it here because we will use it to prove a bound on the number of valuations of combinations that are necessary to determine a welfare-maximizing allocation.

Algorithm **PAR (Pareto optimal)**:
(1) OPEN = $[(1, \ldots, 1)]$
(3) **while** OPEN $\neq []$ **do**
(4) $Remove(c, OPEN)$; SUC $= \mathrm{suc}(c)^3$
(5)   **if** *Feasible(c)* **then**
(6) $PAR = PAR \cup \{c\}$; $Remove(SUC, OPEN)$
(7)   **else foreach** $n \in$ SUC **do**
(8)   **if** $n \notin$ OPEN and *Undominated*(n, PAR)
(9)   **then** $Append(n, OPEN)$

Proposition 7. *The algorithm* **PAR** *determines the set of all Pareto-efficient allocations if the utility functions are injective (that is, no agent is indifferent between any two bundles).*[4]

The second algorithm (family) determines a welfare-maximizing allocation (with respect to reported utilities). It assumes a setting with transferable utility (and makes the usual assumption that utility functions are quasi-linear in money). The algorithm is an improved version of an algorithm that we developed earlier (Conen & Sandholm 2001). Unlike the earlier version, this algorithm does not assume unique utilities.

Algorithm **EBF (Efficient Best First)**:
(1) OPEN = $\{(1, \ldots, 1)\}$;
(2) **loop**
(3) **if** $|$OPEN$| = 1$ **then** $c =$ combination in OPEN
    else Determine
        $M = \{k \in OPEN | v(k) = \max_{d \in OPEN} v(d)\}$.
        if $|M| \geq 1 \wedge \exists d \in M$ with *Feasible(d)* **then return** d
        else Choose $c \in M$ such that
            no $d \in M$ exists with $d \succ c$;
        OPEN = OPEN $\setminus \{c\}$.
(4) **if** *Feasible(c)* **then return** c
(5) SUC $= \mathrm{suc}(c)$
(6) **foreach** $n \in$ SUC **do**
        if $n \notin$ OPEN **then** OPEN = OPEN $\cup \{n\}$

---

[3] $\mathrm{suc}(c)$ determines the immediate successors of c, $\{d \in C \mid \exists j$ with $d[j] = c[j] + 1$ and $d[i] = c[i]\ \forall i \neq j\}$.

[4] If the utility functions are not injective, the algorithm determines a subset of the Pareto-efficient allocations. Furthermore, (even in the non-injective case), the set of solutions found includes a welfare-maximizing allocation (whenever " welfare-maximizing" is defined in a transferable utility context).

Proposition 8. *Any* **EBF** *algorithm determines an efficient (that is, welfare-maximizing) allocation.*

Optimality considerations

The problem is to chose an allocation $X = (X_1, \ldots, X_n)$ with $\bigcup_{i \in N} X_i \subseteq \Omega$ and $\bigcap_{i \in N} X_i = \emptyset$ so as to maximize social welfare, $\sum u_i(X_i)$. Problem instances will be called *economies* $(\Omega; u_1, \ldots, u_n)$.

Definition 9. *An algorithm is* admissible, *if it determines an efficient (that is, welfare-maximizing), feasible combination for every problem instance. This combination is called a* solution combination.

As we have already seen, **EBF** is admissible. Also, **PAR** can easily be turned into an admissible algorithm by comparing the value of all the determined Pareto-optimal allocations and picking a maximizing one. We will refer to this variant of **PAR** as **MPAR** (maximizing PAR). Both algorithms make use of bundle information ("What is your k'th ranked bundle?") and value information ("What is your valuation for bundle X?"). We will now study questions related to optimal use of this information. Roughly, an algorithm (or a family of algorithms) will be considered (weakly) optimal with respect to certain kind of information, if no admissible and "similarly informed" algorithm requires less information for every problem instance. We will show that **EBF** (resp. **MPAR**) are limiting cases for the use of bundle (resp. value) information.

We assume that a valuation function $v : C \to \mathbb{Q}$ is available that can be used to determine the aggregated utility of every possible combination (note that for all a, b with $(a, b) \in \succeq$, $v(a) \geq v(b)$ holds). There is also a feasibility function $f : C \to \{T, F\}$, which allows one to check for every combination in the lattice, whether the combination is feasible (T) or infeasible (F). Further information is not available. Additionally, a successor function is available to determine the next unvisited direct successors of a combination. All of these functions will be considered elementary (with unit costs).

In addition, every algorithm will have to determine a set of combinations to start with; inserting a combination into this set is considered to be an elementary operation. Combinations that are an element of the initial set or that have been "created" as successor by applying (possibly iteratively) the successor function, will be called *visited* combinations. Only visited combinations can be valued or checked for feasibility.

Definition 10. *An algorithm is* admissibly equipped *if only the above mentioned elementary operations are used to obtain lattice-related information.*

EBF is admissibly equipped. However, as given above, **EBF** is not deterministic–it chooses a combination among a set of equally valued combinations arbitrarily in step (3). In effect, **EBF** determines a family of **EBF** algorithms differing in the choice (or *tie breaking*) rule. It is clear that different tie breaking rules may lead to differently efficient (with respect to utilization of information) solution paths.

Now, can some deterministic algorithm search the lattice more efficiently than **EBF** algorithms?

Costs of feasibility checking

Theorem 11 (Efficiency of Feasibility Checking). *There is no admissible, admissibly equipped and deterministic algorithm* **A** *which requires fewer feasibility checks for every problem instance than every algorithm of the* **EBF** *family.*

The proof uses the following propositions (whose proofs we omit due to limited space).

Proposition 12. *Let l be a solution combination that has been determined by an arbitrary* **EBF** *algorithm. Let k be a combination with $v(k) > v(l)$. Then, any admissible, admissibly equipped and deterministic algorithms A must check the feasibility of k.*

Proposition 13. *No* **EBF** *algorithm* **B** *checks a combination k for feasibility which has a lower value than the solution combination l that* **B** *will determine.*

Proposition 14.
(a) Every **EBF** *algorithm checks* precisely one *feasible combination for feasibility.*
(b) Every admissible, admissibly equipped and deterministic algorithm checks at least *one feasible combination for feasibility.*

Now, let l be a solution combination that has been found by an **EBF** algorithm. The theorem holds for every problem instance that does not imply the existence of an infeasible combination with the same value as l. If no such combination exists, **A** has to check *at least* all infeasible combinations with a value larger then $v(l)$ (which is precisely the number of checks that any **EBF** algorithm will perform). ∎

Note that from the non-injectivity of the valuations of the combinations, it follows that **EBF** algorithms may check infeasible combinations (for feasibility) that have the same value as the solution combination. There are even problem instances where *all* **EBF** algorithms will have to perform such checks. One such instance is shown in Fig. 1.
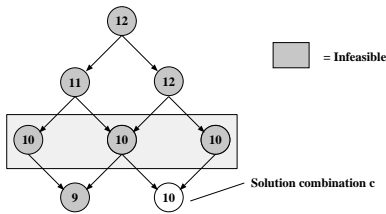

Figure 1: *Part of a lattice that will give any* **EBF** *instance reason to check at least one combination for feasibility that has a value that equals the value of the solution (these are the combinations in the emphasized region).*

In this case, it might be possible that an algorithm **A** performs fewer feasibility checks than any **EBF** algorithm – therefore, the above theorem cannot be formulated more tightly.[5]

---

[5]It is, however, straightforward to give a version of the algorithm that works with equivalence classes of ranks instead of "linearizing" the partial order of the preferences. This allows one to consider all equally valuable combinations at the same time in the selection step. It would, however, make the presentation more awkward.

We will now turn our attention to the **MPAR** algorithm that allows us to formulate some results with respect to the costs of valuating combinations.

Theorem 15. *No admissible, admissibly equipped and deterministic algorithm that calls the valuation function for feasible combinations only will require fewer calls than the* **MPAR** *algorithm.*

The worst case for the **MPAR** algorithm will occur if all feasible allocations that distribute all goods to the consumers are Pareto-optimal. This is the case if all utilities for goods are equal and utilities for bundles are additive, that is $u_i(\{x\}) = c$ for some $c \in \mathbb{Q}_0^+$ and all $x \in \Omega$, $u_i(\emptyset) = 0$, and $u_i(z) = \sum_{x \in z} u_i(\{x\}) = |z| * c$ for all $z \subseteq \Omega$, $z \neq \emptyset$, $i \in N$. It is immediate that every fully distributive allocation leads to the same welfare $W = |\Omega| * c = m * c$. Now, the number of combinations to be valued is the number of allocations that distribute all goods to the customers[6] which is equal to counting the possibilities to distribute m objects to n buckets without further restriction, that is n^m. So:

Proposition 16. **MPAR** *requires at most n^m calls to the valuation function. By the above theorem, this extends to all admissible and admissibly equipped algorithms that are restricted to call the valuation function for feasible combinations only.*

The version of **PAR** that we gave above will determine the set of all Pareto-optimal allocations that are undominated with respect to the chosen ranking. Assume that we have 2 goods, 2 agents and the following utilities $u_x(A) = u_x(B) = 3$, $u_x(AB) = 6$, $x \in \{1, 2\}$ and the ranks $r_1(A) = 1, r_1(B) = 2, r_1(AB) = 3, r_1(\emptyset) = 4$, $r_2(B) = 1, r_2(A) = 2, r_2(AB) = 3, r_2(\emptyset) = 4$. Then, **PAR** only generates the feasible rank combination $(1, 1)$ representing an efficient allocation $(\{A\}, \{B\})$. However, with an unfortunate ordering, more work would have to be done: $r_1(AB) = 1, r_1(A) = 2, r_1(B) = 3, r_1(\emptyset) = 4$ $r_2(AB) = 1, r_2(A) = 2, r_2(B) = 3, r_2(\emptyset) = 4$ would generate $(1, 4), (4, 1), (2, 3)$ and $(3, 2)$.[7] So, the above bound is not sharp but depends on the chosen order of ranks that may vary within each class of individual ranks that map to bundles with equal valuations. The worst case corresponds to the number of combinations in the middle layer of the lattice, see Fig. 2.

With respect to the **EBF** family of algorithms, intuitively a worst case with respect to the number of required valuations would occur if all infeasible combinations would have a higher value than the best feasible combination. This, of course, cannot happen in the considered setting. Instead we have to consider the maximal amount of infeasible combinations that are at least as valuable as the best allocation and are undominated by feasible allocations. This number is $(2^{mn} - n^m)/2$ in the above example and generally.[8]

---

[6]We could extend this to all allocations that include the arbitrator as well and set c to 0. However, with monotonous preferences, the arbitrator can be excluded from consideration.

[7]Requiring agents to rank bundles with equal value from small to large can be beneficial.

[8]In the particular example, all valuation information is elicited.

$$\frac{2^{mn} - n^m}{2}$$

[1,1]

[1,2]   [2,1]

[1,3]   [2,2]   [3,1]

n^m [1,4]   [2,3]   [3,2]   [4,1]

[2,4]   [3,3]   [4,2]

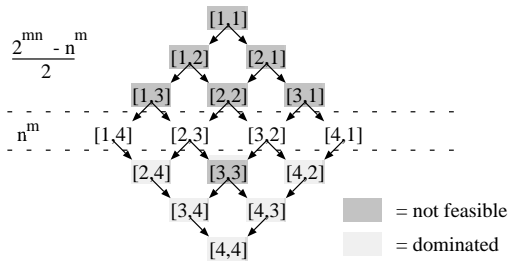[3,4]   [4,3]

[4,4]

█ = not feasible

░ = dominated

Figure 2: *A worst case: All feasible, Pareto-optimal combinations have the same value. The number of undominated Pareto-optimal combinations cannot be larger than in the example. If in the upper part of the lattice such a combination could be found, at least two of the combinations in the middle layer would be dominated. By symmetry, moving undominated combinations into the lower part will lead to fewer undominated combinations as well.*

Proposition 17. *Any* **EBF** *algorithm requires at most* $\frac{2^{mn} - n^m}{2} + 1$ *calls to the valuation function.*

Vickrey payments

Recall that the Vickrey payment of an agent i reflects the effect of her participation in an economy E: a consumer i will pay an amount equal to the utility that the other consumers will loose due to the participation of i, that is

$$t(i) \ = \ V(E_{-i}) \ - \sum_{j \in N, j \neq i} u_j(X_j)$$

where E_{-i} is the economy E without i and $V(E_{-i})$ is the utility that can be realized implementing a welfare-maximizing allocation for E_{-i}.

We will now assume that an execution of an **EBF** algorithm has determined an efficient allocation X for an economy E.

Theorem 18. *No valuation information in addition to the information already obtained by* **EBF** *is necessary to determine the Vickrey payments.*

Proof. We assume that n is the solution combination that was found by the algorithm and that it represents X.(a) First, note that valuation information for all combinations with higher value than n have been obtained already. (b) Now assume that consumer i will be removed from the allocation $X = (X_1, \ldots, X_i, \ldots, X_n)$. The value $V(X^{-X_i})$ of the reduced allocation X^{-X_i} is a lower bound for the maximal value that can be obtained from allocating the goods in Ω to the agents in the remaining set of consumers, N^{-i} (and determines the second term in the equation for the Vickrey payment of i without requiring any additional information). Now assume that a reduced $(n-1)$-ary allocation $Y^{-i} = (Y_1, \ldots, Y_n)$ (leaving out the agent i resp. its index) can be found with a value that exceeds $V(X^{-X_i})$. Further assume that additional valuation information would be required. Then a combination $c = (Y_1, \ldots, X_i, \ldots, Y_n)$ could

be constructed that would have a higher value than X and that would have required additional valuation information to determine its value, thus contradicting (a). □

It is now clear that all required valuation information has already been determined.[9] A consequence of the arguments for (b) is that the partial combinations which solve the restricted allocation problem optimally are part of the already visited combinations. This allows one to extend **EBF** straightforwardly to keep track of the best allocation for each subset of N with $n-1$ elements visited so far (an implementation is straightforward). After the execution of such an extended **EBF** algorithm, Vickrey payments can be determined immediately from the collected maximum valuations and the efficient allocation. From the argumentation above, it follows immediately that

Proposition 19. *An* **EBF** *algorithm, extended as described above, determines an efficient allocation and corresponding Vickrey payments.*

The results obtained on the costs can be carried over to extended **EBF** algorithms. The result above shows that, in the context of admissibly equipped algorithms, the complexity of the determination of Vickrey payments is directly tied to the complexity of the determination of an efficient allocation. In other words: If it is possible to determine (with an **EBF** algorithm) an efficient allocation with a tractable amount of computation (implying also that only a tractable amount of information is required if latency is neglected), it is also possible to tractably compute the Vickrey payments.

The mechanism

To outline a mechanism that is based on the extended **EBF** algorithm, a set of allowed questions, a data structure to store retrieved information and a policy to generate questions are fixed. To fulfill the informational needs of the underlying algorithm, the following two questions will be allowed: (1) *give me the bundle with the next higher rank number* (that is the next weakly less preferred bundle), (2) give me your valuation for bundle x. We assume that all participants know the underlying set of goods. The consumers will consider the first rank question as the start of the mechanism and will answer it with their most preferred bundle (with rank 1). The arbitrator is considered to be trustworthy[10]. In the end, the arbitrator will announce the computed pair $(X_i, t(i))$ to each agent i. Note that the questions allow only for a rather natural sequence of bundle questions–from most preferred towards less preferred.[11]

If we assume a trivial, binary encoding of the $n * 2^m$ values from $\mathbb{Q}_0^+$ and restrict their size to a reasonable constant k, $n * 2^m \log_2 k$ bits of information are required in the worst case.

[9] If the initial combination is feasible and, consequently, no valuation information is available, the Vickrey payments are 0 (every agent receives her most preferred bundle)–therefore no valuation information is necessary to determine Vickrey payments.

[10] An independent institution may assess the trustworthiness of the arbitrator by sampling the sent messages

[11] If the potentially exponential space requirement of the algorithm does not allow one to keep track of all received bundle and value information, polynomial space versions of the algorithm can be used (adapted to, e.g., iterative deepening, see (Zhang & Korf 1995)), requiring that sequences of questions are repeated.

The received information can, for example, be stored in a data structure similar to the augmented order graph of (Conen & Sandholm 2001). Each node represents a quadruple consisting of agent, rank, bundle and value information. Nodes will be created as preferences are explored and information will be added to the nodes upon becoming available. The elicitation of information will be tied into the execution of the algorithm in step (3). To determine the set of value-maximizing combinations in OPEN, all missing bundle/value pairs of combinations in OPEN have to be requested. With one exception, the algorithm immediately implies that valuation requests can be done with the "next worse bundle" question.[12] Together with choosing a tie breaking rule (deciding which value-maximizing combination to expand next), this fixes and implements a straightforward elicitation policy. Implementing the complete mechanism would be straightforward. The resulting mechanism is a member of the family of **RANK** mechanisms that differ only with respect to the tie breaking rule. The following propositions follow immediately from the results above.

Proposition 20. *A **RANK** Mechanism is incentive compatible and economically efficient.*

Proposition 21. *Let **B** be the **EBF** algorithm that is used in a specific **RANK** mechanism. Then there does not exists any other mechanism based on an admissible, admissibly equipped and deterministic algorithm that requires fewer checks of the feasibility of combinations for all instances of the allocation problem.*

Conclusions and future research

We presented a partial-revelation mechanism for combinatorial auctions that explores the natural lattice structure of the bidders' preference combinations. This mechanism gives a possibility to analyze computational and informational requirements of a class of mechanisms from first principles. Analytical results on the amount of elicitation were derived. Two categories of costs were studied. The results carry over to (direct or indirect) mechanisms that do not explore the rank lattice and use valuation questions only (possibly iterative).

Theorem 15 shows that it is impossible to improve upon the number of combinations to be valued without allowing the elicitation algorithm to query the value of infeasible combinations. If this is allowed, the comparison to **EBF** algorithms with respect to required feasibility checks becomes relevant (due to Theorem 11)–in this sense, **EBF** and **MPAR** are, in spite of their simplicity, limit cases (**EBF** algorithms with respect to feasibility checking and **MPAR** with respect to valuation if only feasible combinations are valued.) The mechanism is also of interest because it ties the computational effort required for Vickrey payments directly to the effort required for computing an efficient allocation.

As future work we plan to study the effect of preference restrictions on the performance of **EBF** algorithms. We have

already started conducting experiments on how effective preference elicitation in combinatorial auctions is in practice (Hudson & Sandholm 2002). Future work also includes analyzing the efficiency of elicitation in the sense of the ratio of utility information elicited to the amount of information that is required, *at a minimum* to determine an efficient allocation and corresponding Vickrey payments. We expect that, for a given problem instance, an optimally chosen tie-breaking rule will not leave much room for improvement—however, optimally choosing the tie-breaking rule will generally not be possible *a priori*.

References

Andersson, A.; Tenhunen, M.; and Ygge, F. 2000. Integer programming for combinatorial auction winner determination. *ICMAS*, 39–46.

Bikhchandani, S., and Ostroy, J. 2001. The package assignment model. UCLA Working Paper Series, mimeo.

Bikhchandani, S.; de Vries, S.; Schummer, J.; and Vohra, R. V. 2001. Linear programming and Vickrey auctions.

Conen, W., and Sandholm, T. 2001. Preference elicitation in combinatorial auctions: Extended abstract. *ACM Conference on Electronic Commerce*, 256–259. A more detailed description of the algorithms appeared in the IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms, pp. 71–80.

Fujishima, Y.; Leyton-Brown, K.; and Shoham, Y. 1999. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. *IJCAI*, 548–553.

Hoos, H., and Boutilier, C. 2001. Bidding languages for combinatorial auctions. *IJCAI*, 1211–1217.

Hudson, B., and Sandholm, T. 2002. Effectiveness of preference elicitation in combinatorial auctions. Technical report, Carnegie Mellon University, Computer Science Department, CMU-CS-02-124, March.

Larson, K., and Sandholm, T. 2001. Costly valuation computation in auctions. *Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, 169–182.

Nisan, N., and Segal, I. 2002. The communication complexity of efficient allocation problems. Draft. Second version March 5th.

Nisan, N. 2000. Bidding and allocation in combinatorial auctions. *ACM Conference on Electronic Commerce*, 1–12.

Parkes, D. C., and Ungar, L. 2000a. Iterative combinatorial auctions: Theory and practice. *AAAI*, 74–81.

Parkes, D. C., and Ungar, L. 2000b. Preventing strategic manipulation in iterative auctions: Proxy-agents and price-adjustment. *AAAI*, 82–89.

Parkes, D. C. 1999. Optimal auction design for agents with hard valuation problems. *Agent-Mediated Electronic Commerce Workshop at IJCAI*.

Rothkopf, M. H.; Pekeč, A.; and Harstad, R. M. 1998. Computationally manageable combinatorial auctions. *Management Science* 44(8):1131–1147.

Sandholm, T., and Suri, S. 2000. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. *AAAI*, 90–97.

Sandholm, T.; Suri, S.; Gilpin, A.; and Levine, D. 2001. CABOB: A fast optimal algorithm for combinatorial auctions. *IJCAI*, 1102–1108.

Sandholm, T. 1993. An implementation of the contract net protocol based on marginal cost calculations. *AAAI*, 256–262.

Sandholm, T. 2000a. eMediator: A next generation electronic commerce server. In *International Conference on Autonomous Agents (AGENTS)*, 73–96. Early version: AAAI-99 Workshop on AI in Electronic Commerce, Orlando, FL, pp. 46–55, July 1999, and as Washington University, St. Louis, Computer Science WU-CS-99-02, Jan. 1999.

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

Sandholm, T. 2002. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence* 135:1–54. First appeared as an invited talk at the International Conference on Information and Computation Economies, Charleston, SC, Oct. 25–28, 1998. Also: Washington Univ. Computer Science, WUCS-99-01, Jan 28th, 1999. Conference version: IJCAI, pp. 542–547, 1999.

Wurman, P. R., and Wellman, M. P. 2000. AkBA: A progressive, anonymous-price combinatorial auction. *ACM Conference on Electronic Commerce*, 21–29.

Zhang, W., and Korf, R. E. 1995. Performance of linear-space search algorithms. *Artificial Intelligence* 79(2):241–292.

---

[12]The exception occurs if $(1, \ldots, 1)$ is not feasible which leads to a comparison with its successors, where the implementation has to make sure that the first received value information elicited from each agent is attached to the node representing rank 1.