Approximation Algorithms for Combinatorial Auctions with Complement-Free Bidders

Shahar Dobzinski[*] Noam Nisan[†] Michael Schapira[‡]

February 24, 2005

Abstract

We exhibit three approximation algorithms for the allocation problem in combinatorial auctions with complement free bidders. The running time of these algorithms is polynomial in the number of items m and in the number of bidders n, even though the "input size" is exponential in m. The first algorithm provides an $O(\log m)$ approximation. The second algorithm provides an $O(\sqrt{m})$ approximation in the weaker model of value oracles. This algorithm is also incentive compatible. The third algorithm provides an improved 2-approximation for the more restricted case of "XOS bidders", a class which strictly contains submodular bidders. We also prove lower bounds on the possible approximations achievable for these classes of bidders. These bounds are not tight and we leave the gaps as open problems.

1 Introduction

In a combinatorial auction, a set M of items, $|M|=m$, is sold to n bidders. The combinatorial character of the auction comes from the fact that each bidder values *bundles* of items, rather than valuing items directly. I.e., the i'th bidder's value for each bundle is given by a valuation function v_i, where for each subset $S \subseteq M$, $v_i(S)$ denotes the value (maximum willingness to pay) of the bundle S for bidder i. We assume that for each bidder i, v_i is normalized ($v_i(\emptyset) = 0$), and monotone (for each $S \subseteq T \subseteq M$, $v_i(S) \leq v_i(T)$). The goal is to partition the items between the bidders in a way that maximizes the "social welfare" –

[*]The School of Computer Science and Engineering, The Hebrew University of Jerusalem, `shahard@cs.huji.ac.il`. Supported by grants from the Israel Science Foundation and the USA-Israel Bi-national Science Foundation.

[†]The School of Computer Science and Engineering The Hebrew University of Jerusalem, `noam@cs.huji.ac.il`. Supported by grants from the Israel Science Foundation and the USA-Israel Bi-national Science Foundation.

[‡]The School of Computer Science and Engineering The Hebrew University of Jerusalem `mikesch@cs.huji.ac.il`. Supported by grants from the Israel Science Foundation and the USA-Israel Bi-national Science Foundation.

Valuation Class	**Value**	**Demand**	**General Comm.**
General	$O(\frac{m}{\log m})$ [4] $\Omega(\frac{m}{\sqrt{\log m}})$ [11]	$O(m^{\frac{1}{2}})$ [4]	$\Omega(m^{\frac{1}{2}-\epsilon})$ [20]
CF	$O(m^{\frac{1}{2}})$ (new)	$O(\log m)$ (new)	≥ 2 (new)
XOS			≤ 2 (new) $\geq \frac{4}{3}$ (new)
SM	≤ 2 [14] ≥ 1.02 (P≠NP, new)		$\geq 1 + \frac{1}{2m}$ [20]
GS	1 [3]		

Figure 1: For each class of valuations, we specify the best approximation factor that is acheivable in polynomial time.

the sum of bidders' values of the sets that they get. I.e. to find an allocation $S_1 ... S_m$, $S_i \cap S_j = \emptyset$ for $i \neq j$, that maximizes $\sum_i v_i(S_i)$. This problem is the common abstraction of many complex resource allocation problems both in computational settings and in economic settings and has received much attention from both from both computational and economic points of view – see the recent book [5].

The computational hardness of this problem is double: first, the "input" here is of exponential size – each v_i is described by 2^m real numbers, while we would like our algorithms to run in time polynomial in both m and n – i.e. in time that is poly-logarithmic in the input size. Second, even for valuations that are succinctly described, the optimization problem is computationally hard. Much work has thus been directed at identifying special cases that can be efficiently solved or approximated, as well as understanding the underlying computational limitations – see chapters $10 - 13$ of [5].

Due to the exponential size of the input (that is, exponential in the parameters we care about), there are two approaches to formalize the computational model in which the allocation algorithms must work – specifically how the input valuations are specified and accessed. The first approach calls for fixing some "bidding language" in which the input valuations will be encoded, and requires the algorithms to run in polynomial time in the input length, under this encoding. This approach makes sense in cases for which a sufficiently natural bidding language exists. The second approach is a "concrete complexity" approach: the input valuations are given as "black boxes" and the type of queries that the algorithm may make to these input valuations is fixed. There are three types of query models that are commonly used:

1. **Value queries:** The query specifies a subset $S \subseteq M$ of items and receives the value $v_i(S)$ as the reply. This query is very natural from a "computer science" point of view, but in general is quite weak.

2. **Demand queries:** The query specifies a vector $p = (p_1 ... p_m)$ of "item prices", and the reply is the set that would be "demanded" by the queried bidder under these item prices. I.e., the subset S that maximizes $v_i(S) - \sum_{j \in S} p_j$. This query is natural from an economic point of view as it corresponds to "revealed preferences" of the bidders (i.e. what is directly observable from their behavior). It is known that these queries are strictly stronger than value queries (and in particular can simulate them in polynomial time) [4, 6].

3. **General queries:** In this model we allow *any* kind of query to each valuation function (but the query is always to a single valuation function). This model captures the communication complexity (between the bidders) of the problem, and due to its strength is mostly interesting for lower bounds.

The computational complexity of the allocation problem with general valuations is almost completely understood: in polynomial time, the optimal allocation can be approximated to within a factor of $O(\sqrt{m})$ but not to a factor of $m^{1/2-\epsilon}$ for any $\epsilon > 0$. This is true both in the bidding langauge model for even single-minded bidders [15, 21]. The lower bound applies to general queries, where the upper bound requires demand queries, but value queries do not suffice [6, 4].

In this paper we study the complexity of the allocation problem in the important special case where the input valuations are known not to have any complementarities. I.e., where all input valuations are sub-additive: $v(S \cup T) \leq v(S) + v(T)$ for all S, T [1]. In [14] a strict hierarchy of subclasses within this class of valuations is exhibited: $OXS \subset GS \subset SM \subset XOS \subset CF$. The classes CF and SM are easy to define: CF is the class of sub-additive (complement-free) valuations; SM is the set of submodular valuations, i.e., $v(S \cup T) + v(S \cap T) \leq v(S) + v(T)$ for all S, T. We will not define the class of GS, "(gross) substitute", valuations here, but we will note that economists often assume valuations to be in this class as in some sense this corresponds to "convex economies". The classes OXS and XOS are defined syntactically as what can be defined by OR-of-XORs (resp. XOR-of-ORs) of singleton valuations. See [18] for definitions of the OR and XOR operations. For our purposes, we offer a natural semantic characterization of the syntactically defined XOS language.

The allocation problem gets gradually harder within this hierarchy: a strongly polynomial time algorithm exists if the input valuations are given in the OXS language; a polynomial time algorithm based on linear programming exists for the class GS [20]. For the class SM no polynomial time algorithm exists: NP-hardness for some simple bidding language is shown in [14] and an exponential communication lower bound is shown in [20]. However, [14] exhibit a

[1] It is also possible to consider the "dual" class of substitute-free valuations ($v(S \cup T) \geq v(S) + v(T)$, for disjoint $S, T \subseteq M$). However, it turns out that the lower bound for general valuations [20], also stands for this class.

polynomial-time 2-approximation algorithm that uses only value queries. No approximation algorithms (better than the $O(\sqrt{m})$-approximation for general valuations) were previously known for the higher levels in this hierarchy[2].

We provide new approximation algorithms for these two levels:

Theorem: There exists a polynomial time algorithm that finds a $O(\log m)$ approximation for valuations in the class CF using demand queries.

This algorithm is based on a careful randomized rounding of the linear programming formulation of the problem; a deterministic algorithm is obtained via derandomization.

We also provide an algorithm that uses the weaker model of value queries, and yields a worse approximation ratio. The main novelty in this algorithm is the fact that it is incentive compatible. I.e. the dominant strategy of all bidders is to always report their true valuations. Surprisingly, this is one of the only two incentive compatible approximation algorithms known for combinatorial auctions, that do not apply only to very restricted "single-parameter" domains. The second algorithm is described in [2].

Theorem: There exists an *incentive compatible* polynomial time algorithm that finds a $O(\sqrt{m})$-approximation for valuations in the class CF using *value queries*.

For the more restricted class XOS we obtain an improved approximation ratio.

Theorem: There exists a polynomial time algorithm that finds a 2-approximation for valuations given in the XOS language.

The algorithm is greedy but very different from the algorithm of [14] designed for the more restricted SM class. We also provide a semantic characterization of the class XOS.

We prove lower bounds for approximation for CF and XOS. The class CF does not have a natural bidding language and so the lower bound is in the query model. The lower bound for the class XOS is actually two separate lower bounds: an NP-hardness result for the bidding language model, and a communication lower bound for the query model. No hardness result for approximation to within any constant factor for any of these classes was previously known.

Theorem: Exponential communication is required for approximating the optimal allocation among CF valuations to within any factor less than 2

Theorem: (1) It is NP-hard to approximate the optimal allocation among valuations given in the XOS language to within any constant factor less than $e/(e-1)$. (2) Exponential communication is required for approximating the optimal allocation among XOS valuations to within any factor less than 4/3.

[2]This situation is similar to several other cases where submodular functions can be handled in sub-linear time, but nothing positive is known for more general functions. In particular this includes the celebrated algorithms for minimization of submodular functions [8, 22].

Our results do not completely settle the complexity of approximate allocation among either submodular (SM) or sub-additive (CF) valuations (or XOS valuations). We suggest here the following "natural" conjectures:

Conjecture 1: Finding a better than $O(\log m)$-approximation of the optimal allocation among CF valuations requires an exponential amount of communication.

This would match our upper bound.

Conjecture 2: Finding a better than $O(\sqrt{m})$-approximation of the optimal allocation among CF valuations requires an exponential number of *value queries*.

This would match our upper bound, and highlight the gap between CF and SM (for which a 2-approximation using value queries exists), and also show that demand oracles are indeed required to achieve the $O(\log m)$ approximation.

Conjecture 3: There is a constant $c > 1$ such that any approximation of the optimal allocation for SM valuations to a factor better than c requires an exponential amount of communication.

This would strengthen the known lower bound of $1 + 1/(2m)$ of [20]. In appendix A.1 we prove a result in this direction that is weaker in two aspects: the bound is only for value queries rather than general communication, and it is conditional upon $P \neq NP$. After the submission of this paper, Khot et. al. [12] strengthen this lower bound. They show that if only value queries are used, it is NP-hard to approximate the optimal allocation for submodular valuations to within any constant factor better than $\frac{e}{e-1}$.

Structure of the Paper

In section 2 we present the approximation algorithms for the class CF and the associated lower bound. Section 3 presents the algorithm for the class XOS and the associated lower bounds.

2 Approximating Auctions with CF Valuations

In sections 2.1 and 2.2 we present a top-level description of the $O(\log m)$ approximation algorithm and then fill in the details of the steps. The proof of correctness appears in subsection 2.3.

In section 2.4 we describe an incentive compatible $O(\sqrt{m})$-approximation algorithm). In section 2.5 we present a lower bound – the lower bound is for a general communication model and thus applies to any algorithm that has any type of oracle access to the valuations.

2.1 Using Demand Queries for Approximation

Let us start with an imprecise description of the main idea of the algorithm. Randomized rounding of the LP-relaxation is a standard technique, and our algorithm uses it. However, when one attempts randomized rounding on packing problems such as combinatorial auctions, the results are not good: a randomized choice will very likely yield non-feasible solutions, unless the probabilities taken reduce the expected quality of solution by a large $O(\sqrt{m})$ factor. However, these non-feasible solutions are only a logarithmic factor away from feasibility. For general valuations this does not help, but this is the reason that the k-duplicate version of combinatorial auctions can be well approximated [6, 2, 1].

The main observation at the heart of our algorithm is that one may partition this logarithmically-non-feasible solution into a logarithmic size family of feasible solutions. For the case of complement-free valuations, the quality of one of these solutions may be bounded from below.

Let us now get more precise. We present here an algorithm for the allocation problem with n complement-free valuations. We assume that we have a demand oracle for each of the input valuations.

2.2 The Algorithm

Input: The input is given as a set of n demand oracles for the n valuations v_i.

Output: An allocation T_1,...,T_n which is an $O(\log m)$ approximation to the optimal allocation.

The Algorithm: We first describe the basic steps of the algorithm and then provide the details regarding the implementation of each step.

1. Solve the linear relaxation of the problem:

   Maximize: $\Sigma_{i,S} x_{i,S} v_i(S)$

   Subject to:

   - For each item j: $\Sigma_{i,S|j \in S} x_{i,S} \leq 1$
   - for each bidder i: $\Sigma_S x_{i,S} \leq 1$
   - for each i, S: $x_{i,S} \geq 0$

2. Use randomized rounding to find a "pre-allocation" $S_1, ..., S_n$ of pairs $< i, S_i >$ with the following properties, where $k = c \cdot \log(m)$, and $c > 0$ is a constant to be chosen later:

   - Each item j appears at most k times in $\{S_i\}_i$, with $j \in S_i$.
   - $\sum_i v_i(S_i) \geq \frac{1}{3} \cdot (\Sigma_{i,S} x_{i,S} v_i(S))$.

3. For each bidder i, partition S_i into a disjoint union $S_i = S_i^1 \cup ... \cup S_i^k$ such that for each $1 \leq i_1 < i_2 \leq n$ and $1 \leq r \leq k$, it holds that $S_{i_1}^r \cap S_{i_2}^r = \emptyset$.

4. Find the r that maximizes $\sum_i v_i(S_i^r)$, and for each i allocate $T_i = S_i^r$ to bidder i.

We now mention the details of each step:

1. **Solving the Linear Program:** Even though the linear program has exponentially many variables, it may still be solved in polynomial time. This is done by solving the dual linear program using the ellipsoid method. Using the ellipsoid method requires a "separation" oracle, and this may be directly implemented using the demand oracles of the bidders. This was first proved in [20], and in more details in [4].

2. **Randomized Rounding:** For each bidder i we independently choose a set S_i by performing the following random experiment: each set S is chosen with probability $x_{i,S}$, and the empty set is chosen with probability $1 - \Sigma_S x_{i,S}$. If any of the required constraints is violated, then this stage is repeated from scratch. This randomized step may be converted to be deterministic by derandomizing using the generator of [17] as explained in [23].

3. **Partitioning each S_i:** This is done as follows: for each $i = 1...n$ and each $r = 1..k$, we let $S_i^r = \{j \in S_i | j \text{ appears in exactly } r-1 \text{ of the sets } S_1...S_{i-1}\}$.

4. **Choosing the best partition:** This step is straightforward.

Theorem 2.1 *If all input valuations are complement-free then the algorithm produces an allocation that is a $2k = O(\log m)$-approximation to the optimal one.*

2.3 Analysis

Let us keep track of the "quality" of solution implied by the intermediate steps.

1. The first step returns the optimal fractional solution $OPT^* = \Sigma_{i,S} x_{i,S} v_i(S)$, which is an upper bound to the value of the integral optimal allocation, OPT.

2. The detailed calculations needed to prove that this step indeed ends with a solution that satisfies all the required conditions are given later. At this point we will indicate the types of calculations used and what they yield. From the first inequality of the LP and using Chernoff bounds one can show that for every item j, the probability that it appears in more than k chosen sets is exponentially small in k. The expected value of $\sum_i v_i(S_i)$ at this stage is only slightly less than $\Sigma_{i,S} x_{i,S} v_i(S) = OPT^*$. It follows that with very high probability none of the required constraints are violated, and thus we have $\sum_i v_i(S) \geq \frac{1}{3} \cdot OPT^*$

3. The main point here is that indeed for every fixed r, the sets $\{S_i^r\}_i$ are pairwise disjoint and are thus a valid allocation. This follows directly from the construction, as every duplicate instances of every item j are allocated to sets S_i^r with sequentially increasing r. Note that we always keep $r \leq k$ since each item appears in at most k sets in $\{S_i\}$.

4. The crucial use of complement-freeness comes here: since for each fixed i, $S_i = \bigcup_r S_i^r$, the fact that v_i is complement free implies that $\sum_r v_i(S_i^r) \geq v_i(S_i)$. By summing over all i we get that $\sum_r \sum_i v_i(S_i^r) = \sum_i \sum_r v_i(S_i^r) \geq \sum_i v_i(S_i) \geq \frac{1}{3} \cdot OPT^*$. It is now clear that by choosing the r that maximizes $\sum_i v_i(S_i^r)$ we get that $\sum_i v_i(S_i^r) \geq \frac{OPT^*}{3k}$. Thus the allocation $T_1 = S_1^r, ..., T_n = S_n^r$ is an $O(\log(m))$ approximation to the optimal allocation (and even to the optimal fractional allocation).

2.3.1 Details of Stage 2

We will require the following version of the Chernoff bounds:

Theorem: (Chernoff Bound) Let X_1,...X_n be independent Bernoulli trials such that for $1 \leq i \leq n$, $\Pr[X_i = 1] = p_i$. Then for $X = X_1 + ... + X_n$, $\mu \geq p_1 + ...p_n$, and any $\delta \geq 2e - 1$ we have:

$$\Pr[X > (1+\delta)\mu] < 2^{-\mu\delta}$$

For each $j \in M$, let E_j denote the random variable that indicates whether j was allocated more than k times. Let B be the random variable that indicates whether $v_i(S_i) < \frac{1}{3} OPT^*$. We will prove that $\Pr[\vee_j E_j \vee B] < 1$.

We will first prove that $\Pr[\vee_j E_j] < \frac{1}{20}$. Fix an item j. Let $Z_{i,j}$ be the random variable that determines whether $j \in S_i$. Obviously, $Z_{i,j}$ receives values in $\{0,1\}$. Because of the randomized rounding method we used, we have that the variables $\{Z_{i,j}\}_i$ are independent. We define $Z_j = \Sigma_i Z_{i,j}$ (i.e. Z_j is the number of times item j appears in $\{S_i\}$). By the linearity of expectation and the first condition of the LP formulation we have that $E[Z_j] \leq 1$. We can now use the Chernoff bound, and choose a c such that:

$$\Pr[\text{item } j \text{ appears in more than } k \text{ bundles in } \{S_i\}] =$$

$$\Pr[Z_j > c \cdot \log(m)] < 2^{-(c-1)\cdot\log(m)} < \frac{1}{20m}$$

By applying the union bound we get that the probability that any one of the items appears in more than k bundles in $\{S_i\}$ is smaller than $m \cdot \frac{1}{20m} = \frac{1}{20}$.

We will now prove that $\Pr[B] < \frac{3}{4}$. We assume that $\max_i v_i(M) = 1$ (otherwise, we can divide all valuations by $\max_i v_i(M)$). If $OPT^* \leq 3$, then giving M to the bidder that maximizes $v_i(M)$, is a feasible allocation which provides a good approximation. Therefore, we will assume that $OPT^* > 3$. Let A be the random variable that gets the value of $\Sigma_i v_i(S_i)$ after step 2. We will see that that $A \geq \frac{\Sigma_i v_i(S)}{3}$ with high probability.

We make use of the following corollary from Chebyshev's inequality:

Claim: Let X be the sum of independent random variables, each of which lies in $[0,1]$, and let $\mu = E[X]$. Then, for any $\alpha > 0$, $\Pr[|X - \mu| \geq \alpha] \leq \frac{\mu}{\alpha^2}$.

We can now upper bound the the probability that event B occurs.

$$\Pr[B] = \Pr[A < \frac{OPT^*}{3}] \leq$$

$$\Pr[|A - OPT^*| \geq \frac{2OPT^*}{3}] \leq \frac{9}{4OPT^*} \leq \frac{3}{4}$$

the last inequality is because $OPT^* > 3$. Therefore, using the union bound:

$$\Pr[\vee_{t=1}^{m} E_t \vee B] \leq \Sigma_{j \in M} \Pr[E_j] + \Pr[B] \leq \frac{1}{20} + \frac{3}{4} = 0.8$$

We have shown that with good probability it is possible to create a solution for which all the necessary conditions hold.

2.4 An Incentive-Compatible CF Auction with Value Queries

The only general technique known for making combinatorial auctions incentive compatible is the VCG mechanism (see [9]). Unfortunately, using VCG requires solving the auction optimally - approximations of the optimal social welfare do not suffice [19, 15]. However, the amount of communication required for optimally solving combinatorial auctions is exponential.

In this section we present a way of overcoming this obstacle: limiting the set of possible allocations to a much simpler set. Optimal allocations within this set can be found in polynomial time. Thus, VCG prices can be efficiently calculated, and incentive compatibility follows. Therefore, we are left with showing that the optimal solution within the restricted set of solutions always provides an approximation to the original problem.

In addition, the algorithm answers another question: how well can the optimal social welfare be approximated using only value queries, given that all bidders' valuations are CF. The approximation ratio achieved in this section is $O(\sqrt{m})$. In contrast, for general valuations a lower bound of $O(\frac{m}{\log m})$ is known [6, 4].
Let us now describe the algorithm:

Input: The input is given as a set of n value oracles for the n valuations v_i.

Output: An allocation T_1,...,T_n which is an $O(\sqrt{m})$ approximation to the optimal allocation.

The Algorithm:

1. Query each bidder i for $v_i(M)$, and for $v_i(\{j\})$, for each item j.
2. Construct a bipartite graph by defining a vertex a_j for each item j, and a vertex b_i for each bidder i. Let the set of edges be $E = \cup_{i \in N, j \in M}(a_j, b_i)$. Define the cost of each edge (a_j, b_i) to be $v_i(\{j\})$. Compute the maximum weighted matching $|P|$ in the graph.

3. If the valuation of the bidder i that maximizes $v_i(M)$ is higher than the value of $|P|$, allocate all items to i. Otherwise, for each edge $(a_j, b_i) \in P$ allocate the j'th item to the i'th bidder.

4. Let each bidder pay his VCG price.

Theorem 2.2 *If all the valuations are CF, the algorithm runs in time polynomial in n and m, provides an $O(\sqrt{m})$-approximation to the optimal allocation, and is incentive compatible.*

Proof: Observe that the algorithm indeed runs in polynomial time in n and m: first, the maximal weighted match in bipartite graphs can be solved in polynomial time in m and n [24]. Second, calculation of the VCG prices requires solving only an additional auction for each of the bidders. Note that these additional auctions are smaller in size (one bidder less), and thus can also be done in polynomial time.

Let us now prove that the algorithm provides the desired approximation ratio. Let $OPT = \{T_1, ..., T_k, Q_1, ..., Q_l\}$ be the optimal allocation in the original auction, where for each $1 \leq i \leq k$, $|T_i| < \sqrt{m}$, and for each $1 \leq i \leq l$, $|Q_i| \geq \sqrt{m}$. Let $|OPT| = \Sigma_{i=1}^{l} v_i(Q_i) + \Sigma_{i=1}^{k} v_i(T_i)$.

The first case we consider is when $\Sigma_{i=1}^{l} v_i(Q_i) \geq \Sigma_{i=1}^{k} v_i(T_i)$. Clearly, $\Sigma_{i=1}^{l} v_i(Q_i) \geq \frac{|OPT|}{2}$. Since $l \leq \sqrt{m}$ (otherwise, more than m items were allocated), for the bidder i that maximizes $v_i(O_i)$ it holds that $v_i(M) \geq v_i(Q_i) \geq \frac{|OPT|}{2\sqrt{m}}$. Thus, by assigning all items to bidder i we get the desired approximation ratio.

Consider the case where $\Sigma_{i=1}^{k} v_i(T_i) > \Sigma_{i=1}^{l} v_i(Q_i)$. Clearly, $\Sigma_{i=1}^{k} v_i(T_i) > \frac{|OPT|}{2}$. For each i, $1 \leq i \leq k$, let $c_i = \arg\max_{j \in T_i} v_i(\{j\})$. Notice, that $v_i(\{c_i\}) \geq \frac{v_i(T_i)}{|T_i|}$ (this is due to the CF property: $|T_i| \cdot v_i(\{c_i\}) \geq \Sigma_{j \in T_i} v_i(\{j\}) \geq v_i(T_i)$). Since for all i's $|T_i| < \sqrt{m}$, we have that: $\Sigma_{i=1}^{k} v_i(c_i) > \frac{\Sigma_i v_i(T_i)}{\sqrt{m}} \geq \frac{|OPT|}{2\sqrt{m}}$. By assigning c_i to bidder i we get an allocation in which every bidder gets at most one item with a social welfare of $\Sigma_{i=1}^{k} v_i(\{c_i\}) \geq \frac{|OPT|}{2\sqrt{m}}$. The second allocation, therefore, guarantees at least that social welfare.

We conclude that the approximation the algorithm produces is at least $O(\sqrt{m})$.

Incentive compatibility is guaranteed by the use of the VCG prices. □

2.5 A Lower Bound

Theorem 2.3 *For every $\epsilon > 0$, any $(2-\epsilon)$-approximation algorithm for a combinatorial auction with bidders that have CF valuations, requires an exponential amount of communication.*

Proof: Nisan and Segal [16] present an auction in which each bidder's valuation is restricted to values in $\{0, 1\}$. They show that distinguishing between the case that the optimal social welfare is 1, and the case that it is n, requires exponential communication.

Denote by v_i the valuation of the i'th bidder in the Nisan-Segal auction. Define new CF valuations in the following manner: $v'_i(S) = v_i(S) + 1$. One can easily verify that these new valuations are indeed CF.

Let us now consider a combinatorial auction with these valuations. We can see that distinguishing between the following cases requires exponential communication: the optimal social welfare is $n + 1$, and the optimal social welfare is $2n$. Hence, we have proved that for every $n \geq 2$ achieving $\frac{2n}{n+1}$-approximation requires exponential communication, and the theorem follows. $\square$

3 A 2-Approximation Algorithm for XOS Valuations

The class XOS of valuations was defined in [14] as those valuations that can be syntactically described in the "XOR-of-ORs of singletons" bidding language. This is natural from the point of view of bidding languages (see [18]), and is the strongest known language that is syntactically restricted to representing only complement-free valuations. It is general enough as to represent all submodular valuations as well as some non-submodular valuations (and, as we will show, some interesting computational instances).

For our purposes, we offer a natural semantic characterization of the syntactically defined XOS language. First, we define a query type that we call a "supporting prices" query. We then present our approximation algorithm in a general setting where a "supporting prices" oracle is available (as well as a demand oracle).

In section 3.4, we prove that both "supporting prices" and demand queries can be efficiently answered when the input is given in the XOS bidding language.

The notation of [14] for representing XOS valuations will be used in some of the proofs.

3.1 Supporting Prices

Definition 3.1 *We say that a vector of non-negative prices $p_1, ..., p_m$ supports the bundle S in valuation v if:*

- $\forall j \notin S \; p_j = 0$
- $\sum_{j \in S} p_j = v(S)$.
- *For every bundle T,* $\sum_{j \in T} p_j \leq v(T)$.

The next proposition shows that existence of supporting prices for every bundle is a semantic characterization of the syntactic XOS class.

Proposition 3.2 *A valuation v is in the class XOS if and only if every bundle S has supporting prices.*

Proof: First, we show that XOS valuation indeed has supporting prices. Given an XOS valuation and a set S we choose the clause that maximizes the value of S, and assign the values of the items in the clause to be the prices of the items in S (items which do not appear in the clause get the value of 0). Obviously the sum of prices is indeed $v(S)$. The value of each $T \subseteq S$ is at least the sum of prices, since the clause sets a lower bound to the possible value of $v(T)$. In the other direction, given a valuation which has supporting prices, we will build an XOS valuation as follows: For each $S \subseteq M$ create a clause which consists of the supporting prices of S. Observe that the valuations are indeed equal. □

Our algorithm uses oracles for "supporting prices" queries as well as demand queries.

Definition 3.3 *In a supporting prices query the question is a bundle S and the answer is a vector of supporting prices for it.*

3.2 The Algorithm

This greedy algorithm chooses an arbitrary order of bidders and goes over them, one by one. Each bidder is asked for his demand bundle at given prices. The prices are updated after each step according to the supporting prices.

Input: n valuations v_i, for each of them we are given a demand oracle and a supporting prices oracle.

Output: An allocation S_1,...,S_n which is a 2 approximation to the optimal allocation.

The Algorithm:

```
1. Initialize $S_1 = ... = S_n = \emptyset$, and $p_1, ..., p_m = 0$
2. For each bidder $i = 1...n$ :
   (a) Let $S_i$ be the demand of bidder $i$ at prices $p_1...p_m$.
   (b) For all $i' < i$ take away from $S_i'$ any items from $S_i$: $S_{i'} \leftarrow S_{i'} - S_i$.
   (c) Let $q_1...q_m$ be supporting prices for $S_i$ in $v_i$.
   (d) For all $j \in S_i$, update $p_j = q_j$.
```

3.3 Analysis

Theorem 3.4 *The algorithm provides a 2 approximation to the optimal allocation.*

Proof: For each $T \subseteq M$, we denote by $p^i(T)$ the total price of the items in subset T at the i'th stage of the algorithm. Let $\Delta^i = p^i(M) - p^{i-1}(M)$, i.e. the total difference in prices between stages $(i-1)$ and i (with $p^0(M) = 0$). Let $A_1, ..., A_n$ be the allocation generated by the algorithm. Let $O_1, ..., O_n$ be the optimal allocation. We will prove the $\Sigma_i v_i(O_i) \leq 2\Sigma_i v_i(A_i)$. To do so, we prove three simple lemmas:

Lemma 3.5 *The social welfare of the allocation generated by the algorithm is at least the sum of items' prices at the end of the algorithm (after the n'th stage). I.e.* $p^n(M) \leq \Sigma_i v_i(A_i)$.

Proof: Consider a specific bidder i. Let T be the bundle assigned to that bidder by the algorithm in stage i. Obviously $A_i \subseteq T$. Due to the supporting prices property, we have that $p^i(A_i) \leq v_i(A_i)$. However, since the items in A_i were not reassigned after the i'th stage, and so their prices were not altered, $p^i(A_i) = p^n(A_i)$. We have that $p^n(A_i) \leq v_i(A_i)$, and so $p^n(M) = \Sigma_{i=1}^n p^n(A_i) \leq \Sigma_{i=1}^n v_i(A_i)$. □

Lemma 3.6 *The prices assigned to the items throughout the algorithm are non-decreasing.*

Proof: By contradiction. Let S be the set that maximizes the demand of the i'th bidder at the i'th stage of the algorithm. Let q be a supporting prices vector for S. Now, assume there is an item $j \in S$ for which $q_j < p^i_j$. The supporting prices property ensures that $\Sigma_{t\in(S-\{j\})} q_t \leq v_i(S - \{j\})$ and $\Sigma_{r\in S} q_r = v_i(S)$. Hence:

$$v_i(S) - \Sigma_{r\in S} p^i_r = \Sigma_{r\in S} q_r - \Sigma_{r\in S} p^i_r =$$

$$(q_j - p^i_j) + (\Sigma_{t\in(S-\{j\})} q_t - \Sigma_{t\in(S-\{j\})} p^i_t) <$$

$$(\Sigma_{t\in(S-\{j\})} q_t - \Sigma_{t\in(S-\{j\})} p^i_t) < v_i(S - \{j\}) - \Sigma_{t\in(S-\{j\})} p^i_t$$

and this is a contradiction to the definition of S. □

Lemma 3.7 *The social welfare of the optimal allocation is at most twice the sum of items' prices at the end of the algorithm. I.e.* $\Sigma_i v_i(O_i) \leq 2p^n(M)$.

Proof: Since for each i, $1 \leq i \leq n$, $\Delta^i = \max_{T\subseteq M}(v_i(T) - p^{i-1}(T))$, we have:

$$v_i(O_i) - p^{i-1}(O_i) \leq \Delta^i.$$

since the prices do not decrease throughout the algorithm, the following inequality holds:

$$v_i(O_i) - p^n(O_i) \leq \Delta^i.$$

by summing up on both sides of the equation we get:

$$\Sigma_{i=1}^n v_i(O_i) - \Sigma_i p^n(O_i) \leq \Sigma_i \Delta^i$$

$$\Sigma_i v_i(O_i) - p^n(M) \leq p^n(M)$$

$$\Sigma_i v_i(O_i) \leq 2p^n(M)$$

□

Putting the lemmas together we have that

$$\Sigma_i v_i(O_i) \leq 2p^n(M) \leq 2\Sigma_i v_i(A_i)$$

□

The following example shows that the algorithm does not achieve an approximation ratio better than 2: consider a combinatorial auction with two goods, x_1 and x_2, and two bidders. The first bidder's valuation is $v_1(\{x_1\}) = v_1(\{x_2\}) = v_1(\{x_1 \cup x_2\}) = 1$. The valuation of the second bidder is $v_2(\{x_1\}) = 0$, $v_2(\{x_2\}) = v_2(\{x_1 \cup x_2\}) = 1$. Clearly, a social welfare of 2 can be achieved by allocating x_1 to the first bidder, and x_2 to the second bidder. However, the first bidder might wish to get x_2 at the first stage, and the optimal social welfare achieved is only 1. Hence, the approximation ratio achieved by the algorithm is only 2.

However, if all bidders have the same valuation the approximation guaranteed by the algorithm can be improved:

Theorem 3.8 *If all bidders have the same XOS valuation, the algorithm provides an $\frac{e}{e-1}$-approximation ratio.*

Proof: The proof of the theorem is similar to the proof of [13] which studies the Budgeted Maximum Coverage Problem. We will use the same notation as in the proof of theorem 3.4. We will first prove two lemmas:

Lemma 3.9 *After iteration i of the algorithm we have that*

$$\frac{1}{n}(\Sigma_r v_r(O_r) - p^{i-1}(M)) \leq p^i(M) - p^{i-1}(M) \ (= \Delta^i).$$

Proof: By definition $\Delta^i = max_{T \subseteq M}(v_i(T) - p^{i-1}(T))$. Hence, for each j, $1 \leq j \leq n$

$$v_i(O_j) - p^{i-1}(O_j) \leq \Delta^i.$$

Since all bidders have the same valuation the following inequality holds:

$$v_j(O_j) - p^{i-1}(O_j) \leq \Delta^i.$$

By summing up on j (on both sides of the equation) we get:

$$\Sigma_r v_r(O_r) - p^{i-1}(M) \leq n\Delta^i.$$

and the lemma follows. □

Lemma 3.10 *After the i'th iteration of the algorithm the following holds:*

$$(1 - (1 - \frac{1}{n})^i)(\Sigma_r v_r(O_r)) \leq p^i(M)$$

Proof: We prove this lemma by induction on i. For i=1, $p^1(M) = v_1(M) \geq \frac{1}{n}\Sigma_r v_r(O_r)$. If we assume correctness for $(i-1)$ then:

$$p^i(M) = p^{i-1}(M) + \Delta^i$$

by the previous lemma and the induction step,

$$p^i(M) \geq p^{i-1}(M) + \frac{1}{n}(\Sigma_r v_r(O_r) - p^{i-1}(M))$$

$$= \frac{1}{n}\Sigma_r v_r(O_r) + (1-\frac{1}{n})p^{i-1}(M) \geq$$

$$\frac{1}{n}\Sigma_r v_r(O_r) + (1-\frac{1}{n})(1-(1-\frac{1}{n})^{i-1})(\Sigma_r v_r(O_r))$$

$$= (1-(1-\frac{1}{n})^i)(\Sigma_r v_r(O_r))$$

□

Using the previous lemma we get:

$$\frac{e-1}{e}\Sigma_i v_i(O_i) \leq (1-(1-\frac{1}{n})^n)(\Sigma_i v_i(O_i)) \leq p^n(M) \leq \Sigma_i v_i(A_i)$$

the right inequality was proved in theorem 3.4. □

Theorem 3.13 shows that this ratio is essentially the best possible, as no polynomial-time algorithm achieves a better ratio, unless $P = NP$.

3.4 Efficiently Simulating "Supporting Prices" and Demand Queries

We will now show that if the input is given in the form of an XOS expression, these two oracles can be simulated in time polynomial in the input size.

Lemma 3.11 *The following two queries can be answered for an XOS valuation given as an XOS expression in time polynomial in the input size:*

- *A demand query.*
- *A supporting prices query.*

Proof: Given an XOS valuation and a vector of prices we wish to simulate a demand oracle. First, let us note that it is easy to simulate a demand oracle for an additive valuation in polynomial time, by simply choosing all the profitable items. Since the input is given as an XOS formula (in the form of a matrix) and each clause is an additive valuation, it is enough to simulate a demand oracle for each clause and choose the most profitable option. As there is only a polynomial number of such clauses the entire process requires polynomial time.

The proof of proposition 3.2 contains a method of finding supporting prices to a given bundle of items. □

We do not know whether a supporting prices query can be simulated using only demand oracle queries if the input is not given in the XOS language. However, for the more restricted class of submodular valuations the following holds:

Proposition 3.12 *The supporting prices of submodular valuations can be calculated in polynomial time using only a value oracle.*

Proof: To calculate the supporting prices for some group of $S \subseteq M$ items, first choose an arbitrary order of these items. Set the price p_j of the j'th item to be its marginal utility given the previous $j-1$ items (i.e. $v(1 \cup 2 \cup ... \cup j) - v(1 \cup 2 \cup ... \cup (j-1))$). Observe that $v(S) = \Sigma_{i \in S} p_i$. It's easy to see that for every $T \subseteq S$ $v(T) \geq \Sigma_{i \in T} p_i$, since by definition of submodular valuation, the marginal utility does not increase when items are added. $\Box$

3.5 Lower Bounds

We prove two lower bounds: The first is a lower bound in the NP-hardness model that applies when the input is in XOS encoding. The second lower bound is in communication and applies when the input is accessed via oracles.

Theorem 3.13 *It is NP-hard to approximate the optimal allocation among valuations given in the XOS language to within any constant factor less than $e/(e-1)$.*

Proof: We will show a polynomial-time reduction from MAX-k-COVER. MAX-k-Cover is defined as follows: Given m items, and a collection of subsets of these items, the objective is to maximize the number of items which can be covered by k subsets. It is known that it is NP-hard to approximate this problem within a better factor than $\frac{e}{e-1}$ [7]. This problem can be converted into a combinatorial auction with XOS valuations: given an instance of MAX-k-COVER, we create an auction with k bidders and m goods. Each bidder has the same XOS valuation in which there is a clause for each subset in the original problem, and the value of every item in the clause is 1. Clearly, every choice of k subsets in the original problem corresponds to an allocation in the combinatorial auction with the same value. All we need to do is to assign all items in set i to bidder i (and avoid assigning one item to more then one bidder). In the other direction, every allocation corresponds to a choice of k sets in the original problem with at least the social welfare value: We choose k subsets, so that subset i contains the items in the clause maximizing bidder i's gain. This way we are guaranteed that the number of items covered is no less than the social welfare. The theorem follows. $\Box$

Theorem 3.14 *Exponential communication is required for approximating the optimal allocation among XOS valuations to within any factor less than* $4/3$

Proof: In [16] it is shown that any algorithm achieving a $(\frac{1}{2}\log m - \epsilon)$ approximation for the SET-COVER problem (as defined there) requires exponential communication. In [7] (proposition 12), it is shown how to deduce a lower bound for MAX-k-COVER by a reduction from SET-COVER. An analogous method produces a communication lower bound of $\frac{4}{3}$ to the communication version of MAX-k-COVER. In this version of the problem we have k players, each holding a collection of subsets of M. We are interested in maximizing the number of items covered by k subsets when subset i can only be chosen from player i's subsets. An argument similar to the one in theorem 3.13 shows that there is an approximation preserving reduction from this version of MAX-k-COVER to an auction where the bidders have different XOS valuations. Hence, any algorithm with approximation ratio better than $\frac{4}{3}$ requires exponential communication. □

Acknowledgments

We would like to thank Daniel Lehmann for comments on an earlier draft of this paper.

References

[1] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agent. In *Proceedings of the 14th Annual ACM Symposium on Discrete Algorithms (SODA)*, 2003.

[2] Yair Bartal, Rica Gonen, and Noam Nisan. Incentive compatible multi unit combinatorial auctions. In *TARK 03*, 2003.

[3] Alejandro Bertelsen and Daniel Lehmann. Substitutes valuations: M#-concavity. Working Paper.

[4] Liad Blumrosen and Noam Nisan. On the computational power of iterative auctions I: Demand queries, 2005. Working paper.

[5] P. Cramton, Y. Shoham, and R. Steinberg (Editors). *Combinatorial Auctions.* MIT Press. Forthcoming., 2005. http://www.cramton.umd.edu/papers2000-2004/cramton-shoham-steinberg-combinatorial-auctions.pdf.

[6] Shahar Dobzinski and Michael Schapira. Optimal upper and lower approximation bounds for k-duplicates combinatorial auctions, 2005. Working paper.

[7] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[8] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *IIASA Collab. Proceedings Ser. CP-81-S1*, pages 511–546, 1981.

[9] T. Groves. Incentives in teams. *Econometrica*, pages 617–631, 1973.

[10] J. Håstad. Some optimal inapproximability results. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 1–10, 1997.

[11] Ron Holzman, Noa Kfir-Dahav, Dov Monderer, and Moshe Tennenholtz. Bundling equilibrium in combinatrial auctions. *Games and Economic Behavior*, 47:104–123, 2004.

[12] Subhash Khot, Richard Lipton, Evangelos Markakis, and Aranyak Mehta. Inapproximability results for combinatorial auctions with submodular utility functions, 2005. Working paper.

[13] Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

[14] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *ACM conference on electronic commerce*, 2001.

[15] Daniel Lehmann, Liadan Ita O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. In *JACM 49(5)*, pages 577–602, Sept. 2002.

[16] Noam Nisan. The communication complexity of approximate set packing and covering. In ICALP 2002.

[17] Noam Nisan. $RL \subseteq SC$. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 619–623. ACM Press, 1992.

[18] Noam Nisan. *Chapter 9: Bidding Languages. In P. Cramton and Y. Shoham and R. Steinberg (Editors), Combinatorial Auctions.* MIT Press. Forthcoming, 2005. Available from http://www.cs.huji.ac.il/˜noam/mkts.html.

[19] Noam Nisan and Amir Ronen. Computationally feasible vcg-based mechanisms. In *ACM Conference on Electronic Commerce*, 2000.

[20] Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices, 2003. Working paper. Available from http://www.cs.huji.ac.il/˜noam/mkts.html.

[21] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[22] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory Ser. B*, 80(2):346–355, 2000.

[23] D. Sivakumar. Algorithmic derandomization via complexity theory. In *Proceedings of the 34th annual ACM Symposium on Theory of Computing (STOC)*, 2002.

[24] Robert Endre Tarjan. *Data structures and network algorithms.* Society for Industrial and Applied Mathematics, 1983.

A A Constant Lower Bound for Approximating Submodular Auctions

Theorem A.1 *Using value queries only, it is NP-hard to approximate the optimal allocation among submodular to within any constant factor less than* 1.02.

Proof: We will prove the lower bound by reducing from MAX-CUT. MAX-CUT is the problem of finding a partition of vertices in a graph G, such that the cut (the number of edges with one vertex in either side of the partition) is maximized. Given a graph $G = (V, E)$, define a two bidder auction as follows: the goods will be the vertices, and the two bidders will have an identical valuation functions: $v(S) = |\cup_{(i,j)\in E|i\in S}(i, j)|$. Observe that these valuations are indeed submodular, and that for every S, $v(S)$ can be calculated in polynomial time. Each allocation defines a cut, and each cut defines an allocation. The total social welfare of an allocation is $|E|$, plus the number of edges which have been counted twice (once for each bidder). The edges which have been counted twice, are the cut defined by the allocation. Denote by ALG the number of edges that have been counted twice in the combinatorial auction, and by OPT the number of edges in the maximum cut. Recalling that the maximum cut consists of at least half of the edges, we have that

$$\frac{|E| + ALG}{|E| + OPT} \leq \frac{2OPT + ALG}{3OPT} = \frac{2}{3} + \frac{ALG}{3OPT}$$

[10] sets a lower bound of $\frac{17}{16} - \epsilon$ for approximating MAX-CUT. We can now conclude that it is NP-hard to approximate combinatorial auctions with submodular valuations within a factor of $\frac{51}{50} - \epsilon$. $\square$