On the Computational Power of Iterative Auctions I: Demand Queries

Liad Blumrosen and Noam Nisan *

Abstract

We study the computational power and limitations of iterative combinatorial auctions. Most existing iterative combinatorial auctions are based on repeatedly suggesting prices for bundles of items, and querying the bidders for their "demand" under these prices. We prove several results regarding such auctions that use a polynomial number of demand queries: (1) that such auctions can simulate several other natural types of queries; (2) that such auctions can solve linear programming relaxations of winner determination problems; (3) that they can approximate the optimal allocation as well as generally possible using polynomial communication or computation, while weaker types of queries can not do so. We also initiate the study of how can the prices of bundles be represented when they are not linear, and show that the "default" representation has severe limitations.

*Email: {liad,noam}@cs.huji.ac.il. School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel. Supported by grants from the Israeli Academy of Sciences and the USA-Israel Binational Science Foundation.

1 Introduction

Combinatorial auctions have recently received a lot of attention. In a combinatorial auction, a set M of m non-identical items are sold in a single auction to n competing bidders. The bidders have preferences regarding the *bundles of items* that they may receive. The preferences of bidder i are specified by a valuation function $v_i : 2^M \rightarrow R^+$, where $v_i(S)$ denotes the value that bidder i attaches to winning the bundle of items S. We assume "free disposal", i.e., that the v_i's are monotone non-decreasing. The usual goal of the auctioneer is to optimize the social welfare $\sum_i v_i(S_i)$, where the allocation $S_1...S_n$ must be a partition of the items. Applications include many complex resource allocation problems and, in fact, combinatorial auctions may be viewed as *the* common abstraction of many complex resource allocation problems. Combinatorial auctions face both economic and computational difficulties and are a central problem in the recently active border of economic theory and computer science. A forthcoming book [7] addresses many of the issues involved in the design and implementation of combinatorial auctions.

The design of a combinatorial auction involves many considerations. In this paper we focus on just one central issue: the communication between bidders and the allocation mechanism – "preference elicitation". Transferring all information about bidders' preferences requires an infeasible (exponential in m) amount of communication. Thus, "direct revelation" auctions in which bidders simply declare their preferences to the mechanism are only practical for very small auction sizes or for very limited families of bidder preferences. We have therefore seen a multitude of suggested "iterative auctions" in which the auction protocol repeatedly interacts with the different bidders, aiming to adaptively elicit enough information about the bidders' preferences as to be able to find a good (optimal or close to optimal) allocation.

Most of the suggested iterative auctions proceed by maintaining temporary prices for the bundles of items and repeatedly querying the bidders as to their preferences between the bundles under the current set of prices, and then updating the set of bundle prices according to the replies received (e.g., [15, 8, 11, 27, 1]). Effectively, such an iterative auction accesses the bidders' preferences by repeatedly making the following type of *demand query* to bidders: "Query to bidder i: a vector of bundle prices $p = \{p(S)\}_{S \subseteq M}$; Answer: a bundle of items $S \subseteq M$ that maximizes $v_i(S) - p(S)$.". These types of queries are very natural in an economic setting as they capture the "revealed preferences" of the bidders. Some auctions, called *item-price* or *linear-price* auctions, specify a price p_i for each *item*, and the price of any given bundle S is always linear, $p(S) = \sum_{i \in S} p_i$. Other auctions, called *bundle-price* auctions, allow specifying arbitrary (non-linear) prices $p(S)$ for bundles.

In this paper, we embark on a systematic analysis of the computational power of iterative auctions that are based on demand queries. We do not aim to present auctions for practical use but rather to understand the limitations and possibilities of these kinds of auctions. Our main question is what can be done using a polynomial number of these types of queries? That is, polynomial in the main parameters of the problem: n, m and the number of bits t needed for representing a single value $v_i(S)$. Note that from an algorithmic point of view we are talking about sub-linear time algorithms: the input size here is really $n(2^m - 1)$ numbers – the descriptions of the valuation functions of all bidders. There are two aspects to computational efficiency in these settings: the first is the communication with the bidders, i.e., the number of queries made, and the second is the "usual" computational tractability. Our lower bounds will depend only on the number of queries – and hold independently of any computational assumptions like $P \neq NP$. Our upper bounds will always be computationally efficient both in terms of the number of queries and in terms of regular computation. As mentioned, this paper concentrates on the single aspect of preference elicitation and on its computational consequences and does not address issues of incentives. This strengthens our lower bounds, but means that the upper bounds require evaluation from this perspective also before being used in any real combinatorial auction.[1]

[1] We do observe however that some weak incentive property comes for free in demand-query auctions since "myopic" players will answer all demand queries truthfully. We also note that in some cases (but not always!) the incentives issues can be handled orthogonally to the preference elicitation issues, e.g., by using Vickrey-Clarke-Groves (VCG) prices (e.g., [2, 25]).

Valuation family	Upper bound	Reference	Lower bound	Reference
General	$min(n, O(\sqrt{m}))$	[18], Section 5	$min(n, m^{1/2-\epsilon})$	[23]
Substitutes	1	[23]		
Submodular	2	[17],	$1+\frac{1}{2m}$	[23]
Subadditive	$O(logm)$	[9]	2	[9]
k-duplicates	$O(m^{1/k+1})$	[10]	$O(m^{1/k+1})$	[10]
Procurement	$\ln m$	[23]	$(\log m)/2$	[20, 23]

Figure 1: The best approximation factors currently achievable by computationally-efficient combinatorial auctions, for several classes of valuations. All lower bounds in the table apply to all iterative auctions; all upper bounds in the table are achieved with item-price demand queries.

In a companion paper ([5]) we study similar questions for the more restricted natural case of *ascending-price* combinatorial auctions.

1.1 Extant Work

Many iterative combinatorial auction mechanisms rely on demand queries (see the survey in [26]). For our purposes, two families of these auctions serve as the main motivating starting points: the first is the ascending item-price auctions of [15, 11] that with computational efficiency find an optimal allocation among "(gross) substitutes" valuations, and the second is the ascending bundle-price auctions of [27, 1] that find an optimal allocation among general valuations – but not necessarily with computational efficiency. The main lower bound in this area, due to [23], states that indeed, due to inherent communication requirements, it is not possible for any iterative auction to find the optimal allocation among general valuations with sub-exponentially many queries. A similar exponential lower bound was shown by [23] also for even approximating the optimal allocation to within a factor of $m^{1/2-\epsilon}$. Several lower bounds and upper bounds for approximation are known for some natural classes of valuations – these are summarized in Figure 1.

In [23], the universal generality of demand queries is also shown: any *non-deterministic* communication protocol for finding an allocation that optimizes the social welfare can be converted into one that only uses demand queries (with bundle prices). In [29] this was generalized also to non-deterministic protocols for finding allocations that satisfy other natural types of economic requirements (e.g., approximate social efficiency, envy-freeness). However, in [24] it was demonstrated that this "completeness" of demand queries holds only in the nondeterministic setting, while in the usual deterministic setting, demand queries (even with bundle prices) may be exponentially weaker than general communication.

Bundle-price auctions are a generalization of (the more natural and intuitive) item-price auctions. It is known that indeed item-price auctions may be exponentially weaker: a nice example is the case of valuations that are an XOR of k bundles[2], where k is small (say, polynomial). Lahaie and Parkes [16] show an economically-efficient bundle-price auction that uses a polynomial number of queries whenever k is polynomial. In contrast, [4] show that there exist valuations that are XORs of $k = \sqrt{m}$ bundles such that any item-price auction that finds an optimal allocation between them requires exponentially many queries.

1.2 Our Results

Comparison of query types

We first ask what other natural types of queries could we imagine iterative auctions using? Here is a list of such queries that are either natural, have been used in the literature, or that we found useful.

1. *Value query*: The auctioneer presents a bundle S, the bidder reports his value $v(S)$ for this bundle.

[2]These are valuations where bidders have values for k specific packages, and the value of each bundle is the maximal value of one of these packages that it contains.

2. *Marginal-value query*: The auctioneer presents a bundle A and an item j, the bidder reports how much he is willing to pay for j, given that he already owns A, i.e., $v(j|A) = v(A \cup \{j\}) - v(A)$.

3. *Demand query (with item prices)*: The auctioneer presents a vector of item prices $p_1...p_m$; the bidder reports his demand under these prices, i.e., some set S that maximizes $v(S) - \sum_{i \in S} p_i$.[3]

4. *Indirect-utility query*: The auctioneer presents a set of item prices $p_1...p_m$, and the bidder responds with his "indirect-utility" under these prices, that is, the highest utility he can achieve from a bundle under these prices: $max_{S \subseteq M}(v(S) - \sum_{i \in S} p_i)$.[4]

5. *Relative-demand query*: the auctioneer presents a set of non-zero prices $p_1...p_m$, and the bidder reports the bundle that maximizes his value per unit of money, i.e., some set that maximizes $\frac{v(S)}{\sum_{i \in S} p_i}$.[5]

Theorem: Each of these queries can be efficiently (i.e., in time polynomial in n, m, and the number of bits of precision t needed to represent a single value $v_i(S)$) simulated by a sequence of demand queries with item prices.

In particular this shows that demand queries can elicit all information about a valuation by simulating all $2^m - 1$ value queries. We also observe that value queries and marginal-value queries can simulate each other in polynomial time and that demand queries and indirect-utility queries can also simulate each other in polynomial time. We prove that exponentially many value queries may be needed in order to simulate a single demand query.[6]

Demand queries and Linear Programs

The winner determination problem in combinatorial auctions may be formulated as an integer program. In many cases solving the linear-program relaxation of this integer program is useful: for some restricted classes of valuations it finds the optimum of the integer program (e.g., substitute valuations [15, 11]) or helps approximating the optimum (e.g., by randomized rounding [9, 10]). However, the linear program has an exponential number of variables. Nisan and Segal [23] observed the surprising fact that despite the exponential number of variables, this linear program may be solved within polynomial communication. The basic idea is to solve the dual program using the Ellipsoid method (see, e.g., [14]). The dual program has a polynomial number of variables, but an exponential number of constraints. The Ellipsoid algorithm runs in polynomial time even on such programs, provided that a "separation oracle" is given for the set of constraints. Surprisingly, such a separation oracle can be implemented using a single demand query (with item prices) to each of the bidders.

The treatment of [23] was somewhat ad-hoc to the problem at hand (the case of substitute valuations). Here we give a somewhat more general form of this important observation. Let us call the

[3] A tie breaking rule should be specified. All of our results apply for any fixed tie breaking rule.

[4] This is exactly the utility achieved by the bundle which would be returned in a demand query with the same prices. This notion relates to the Indirect-utility function studied in the Microeconomic literature (see, e.g., [19]).

[5] Note that when all the prices are 1, the bidder actually reports the bundle with the highest per-item price. We found this type of query useful, for example, in the design of the approximation algorithm described in Figure 4 in Section 5.

[6] It is interesting to note that for the restricted class of substitutes valuations, demand queries may be simulated by polynomial number of value queries [3].

following class of linear programs "generalized-winner-determination-relaxation (GWDR) LPs":

$$\begin{aligned}
\textbf{Maximize} \quad & \sum_{i \in N, S \subseteq M} w_i \, x_{i,S} \, v_i(S) & \\
\textbf{s.t.} \quad & \sum_{i \in N,\ S | j \in S} x_{i,S} \leq q_j & \forall j \in M \\
& \sum_{S \subseteq M} x_{i,S} \leq d_i & \forall i \in N \\
& x_{i,S} \geq 0 & \forall i \in N, S \subseteq M
\end{aligned}$$

The case where $w_i = 1, d_i = 1, q_j = 1$ (for every i, j) is the usual linear relaxation of the winner determination problem. More generally, w_i may be viewed as the weight given to bidder i's welfare, q_j may be viewed as the quantity of units of good j, and d_i may be viewed as duplicity of the number of bidders of type i.

Theorem: Any GWDR linear program may be solved in polynomial time (in n, m, and the number of bits of precision t) using only demand queries with item prices.[7]

Welfare approximation

The next question that we ask is how well can a computationally-efficient auction that uses only demand queries *approximate* the optimal allocation? Two separate obstacles are known: In [23], a lower bound of $min(n, m^{1/2-\epsilon})$, for any fixed $\epsilon > 0$, was shown for the approximation factor obtained using any polynomial amount of communication. A computational bound with the same value applies even for the case of single-minded bidders, but under the assumption of $NP \neq ZPP$ [28]. As noted in [23], the computationally-efficient greedy algorithm of [18] can be adapted to become a polynomial-time iterative auction that achieves a nearly matching approximation factor of $min(n, O(\sqrt{m}))$. This iterative auction may be implemented with bundle-price demand queries but, as far as we see, not as one with item prices. Since in a single bundle-price demand query an exponential number of prices can be presented, this algorithm can have an exponential communication cost. In Section 5, we describe a different item-price auction that achieves the same approximation factor with a polynomial number of queries (and thus polynomial communication).

Theorem: There exists a computationally-efficient iterative auction with item-price *demand queries* that finds an allocation that approximates the optimal welfare between arbitrary valuations to within a factor of $min(n, O(\sqrt{m}))$.

One may then attempt obtaining such an approximation factor using iterative auctions that use only the weaker value queries. However, we show that this is impossible:

Theorem: Any iterative auction that uses a polynomial (in n and m) number of *value queries* can not achieve an approximation factor that is better than $O(\frac{m}{\log m})$.[8]

Note however that auctions with only value queries are not completely trivial in power: the bundling auctions of [13] can easily be implemented by a polynomial number of value queries and can achieve an approximation factor of $O(\frac{m}{\sqrt{\log m}})$ by using $O(\log m)$ equi-sized bundles. We do not know how to close the (tiny) gap between this upper bound and our lower bound. Figure 2 summarizes all these results.

Representing bundle prices

We then deal with a critical issue with bundle-price auctions that was side-stepped by our model, as well as by all previous works that used bundle-price auctions: how are the bundle prices represented? For item-price auctions this is not an issue since a query needs only to specify a small number, m, of prices.

[7] The produced optimal solution will have polynomial support and thus can be listed fully.

[8] This was also proven independently by Shahar Dobzinski and Michael Schapira.

Query type	Upper bound	Reference	Lower bound	Reference
General Communication	$min(n, O(m^{1/2}))$	[18]	$min(n, m^{1/2-\epsilon})$	[23]
Demand Queries	$min(n, O(m^{1/2}))$	**new**	$min(n, m^{1/2-\epsilon})$	[23]
Value Queries	$O(\frac{m}{\sqrt{\log m}})$	[13])	$O(\frac{m}{\log m})$	**new**

Figure 2: Achievable approximation factors for the social welfare using polynomially many value queries, demand queries (with item prices), and general queries (communication).

In bundle-price auctions that situation is more difficult since there are exponentially many bundles that require pricing. Our basic model (like all previous work that used bundle prices, e.g., [27, 25, 1]), ignores this issue, and only requires that the prices be determined, *somehow*, by the protocol. A finer model would fix a specific *language* for denoting bundle prices, force the protocol to represent the bundle-prices in this language, and require that the *representations of the bundle-prices* also be polynomial.

What could such a language for denoting prices for all bundles look like? First note that specifying a price for each bundle is equivalent to specifying a *valuation*. Second, as noted in [22], most of the proposed *bidding languages* are really just languages for representing valuations, i.e., a syntactic representation of valuations – thus we could use any of them. This point of view opens up the general issue of *which* language should be used in bundle-price auctions and what are the implications of this choice.

Here we initiate this line of investigation. We consider bundle-price auctions where the prices must be given as a XOR-bid, i.e., the protocol must explicitly indicate the price of every bundle whose value is different than that of all of its proper subsets. Note that all bundle-price auctions that do not explicitly specify a bidding language must implicitly use this language or a weaker one, since without a specific language one would need to list prices for all bundles, perhaps except for trivial ones (those with value 0, or more generally, those with a value that is determined by one of their proper subsets.) We show that once the representation length of bundle prices is taken into account (using the XOR-language), bundle-price auctions are no more strictly stronger than item-price auctions. Define the *cost* of an iterative auction as the total length of the queries and answers used throughout the auction (in the worst case).

Theorem: For some class of valuations, bundle price auctions that use the XOR-language require an exponential cost for finding the optimal allocation. In contrast, item-price auctions can find the optimal allocation for this class within polynomial cost.[9]

This put doubts on the applicability of bundle-price auctions like [1, 27], and it may justify the use of "hybrid" pricing methods such as Ausubel, Cramton and Milgrom's Clock-Proxy auction ([6]).

The organization of the rest of the paper is as follows: Section 2 describes our model. In Section 3 we discuss the power of different types of queries, and in Section 4 we show how demand queries help solving linear programs for winner determination problems. Section 5 studies approximations with a polynomial number of queries, and finally, Section 6 studies the representation of bundle-price demand queries. All proofs can be found in the appendix.

2 The Model

A single auctioneer is selling m indivisible non-homogeneous items in a single auction, and let M be the set of these items and N be the set of bidders. Each one of the n bidders in the auction has a valuation function $v_i : 2^M \rightarrow \{0, 1, ..., L\}$, where for every bundle of items $S \subseteq M$, $v_i(S)$ denotes the value of bidder i for the bundle S and is an integer in the range $0...L$. We will sometimes denote the number of

[9] Our proof relies on the sophisticated known lower bounds for constant depth circuits. We were not able to find an elementary proof.

bits needed to represent an integer in the range $0...L$ by $t = \log L$. We assume free disposal, i.e., $S \subseteq T$ implies $v_i(S) \leq v_i(T)$ and that $v_i(\emptyset) = 0$ for all bidders.

A valuation is called a *k-bundle XOR* if it can be represented as a XOR combination of at most k atomic bids [21], i.e., if there are at most k bundles S_i and prices p_i such that for all S, $v(S) = max_{i|S \supseteq S_i} p_i$.[10]

2.1 Iterative Auctions

The auctioneer sets up a protocol (equivalently an "algorithm"), where at each stage of the protocol some information q – termed the "query" – is sent to some bidder i, and then bidder i replies with some reply that depends on the query as well as on his own valuation. In this paper, we assume that we have complete control over the bidders' behavior, and thus the protocol also defines a reply function $r_i(q, v_i)$ that specifies bidder i's reply to query q. The protocol may be adaptive: the query value as well as the queried bidder may depend on the replies received for past queries. At the end of the protocol, an *allocation* $S_1...S_n$ must be declared, where $S_i \cap S_j = \emptyset$ for $i \neq j$.

We say that the auction finds an *optimal allocation* if it finds the allocation that maximizes the social welfare $\sum_i v_i(S_i)$. We say that it finds a c-approximation if $\sum_i v_i(S_i) \geq \sum_i v_i(T_i)/c$ where $T_1...T_n$ is an optimal allocation. The running time of the auction on a given instance of the bidders' valuations is the total number of queries made on this instance. The running time of a protocol is the worst case cost over all instances. Note that we impose no computational limitations on the protocol or on the players.[11] This of course only strengthens our hardness results. Yet, our positive results will not use this power and will be efficient also in the usual computational sense.

Our goal will be to design computationally-efficient protocols. We will deem a protocol computationally-efficient if its cost is polynomial in the relevant parameters: the number of bidders n, the number of items m, and $t = \log L$, where L is the largest possible value of a bundle. Note that all of our results give concrete bounds, where the dependence on the parameters is given explicitly; we use the standard big-Oh notation just as a shorthand.

2.2 Demand Queries

Most of the paper will be concerned with a common special case of iterative auctions that we term "demand auctions". In such auctions, the queries that are sent to bidders are demand queries: the query specifies a price $p(S) \in \Re^+$ for each bundle S. The reply of bidder i is simply the set most desired – "demanded" – under these prices. Formally, a set S that maximizes $v_i(S) - p(S)$. It may happen that more than one set S maximizes this value. In which case, ties are broken according to some fixed tie-breaking rule, e.g., the lexicographically first such set is returned. All of our results hold for any fixed tie-breaking rule.

Note that even though in our model valuations are integral, we allow the demand query to use arbitrary real numbers. A practical issue here is how will the query be specified: in the general case, an exponential number of prices needs to be sent in a single query. Formally, this is not a problem as the model does not limit the length of queries in any way – the protocol must simply define what the prices are in terms of the replies received for previous queries. We look into this issue further in Section 6.

Many auctions in the literature restrict the prices' representation to item prices (or linear prices):

Definition 1. Item Prices: The prices in each query are given by prices p_j for each item j. The price of a set S is additive: $p(S) = \sum_{j \in S} p_j$.

[10] For example, consider a bidder with values of 5,3,4 for the atomic bundles $abcd, ac, b$, respectively. For this valuation, $v(ac) = 3$, $v(dcb) = 4$ but $v(abcd) = 5$.

[11] The running time really measures communication costs and not computational running time.

	Value	Mar-value	Demand	Ind-util	Rel-demand
Value query	1	2	exp	exp	exp
Marginal-value query	m	1	exp	exp	exp
Demand query	mt	poly	1	mt+1	poly
Indirect-utility query	1	2	m+1	1	poly
Relative-demand query	-	-	-	-	1

Figure 3: Each entry in the table specifies how many queries of this row are needed to simulate a query from the relevant column.

3 The Power of Different Types of Queries

In this section we compare the power of the various types of queries defined in the introduction. We will present computationally-efficient simulations of these query types using item-price demand queries. In a companion paper [5] we show that these simulations can also be done using item-price *ascending* auctions.

Lemma 1. *A value query can be simulated by m marginal-value queries. A marginal-value query can be simulated by two value queries.*

Lemma 2. *A value query can be simulated by mt demand queries (where $t = \log L$ is the number of bits needed to represent a single bundle value).*[12]

As a direct corollary we get that demand auctions can always fully elicit the bidders' valuations by simulating all possible $2^m - 1$ queries and thus elicit enough information for determining the optimal allocation. Note, however, that this elicitation may be computationally inefficient.

The next lemma shows that demand queries can be exponentially more powerful than value queries.

Lemma 3. *An exponential number of value queries may be required for simulating a single demand query.*

Indirect utility queries are, however, equivalent in power to demand queries:

Lemma 4. *An indirect-utility query can be simulated by $mt + 1$ demand queries. A demand query can be simulated by $m + 1$ indirect-utility queries.*

Demand queries can also simulate relative-demand queries:[13]

Lemma 5. *Relative-demand queries can be simulated by a polynomial number of demand queries.*

According to our definition of relative-demand queries, they clearly cannot simulate even value queries. Figure 3 summarizes the relations between these query types.

[12]Note that t *bundle-price* demand queries can easily simulate a value query by setting the prices of all the bundles except S (the bundle with the unknown value) to be L, and performing a binary search on the price of S.

[13]Note: although in our model values are integral, we allow the query prices to be arbitrary real numbers, thus we may have bundles with arbitrarily close relative demands. In this sense the simulation above is only up to any given ϵ (and the number of queries is $O(\log L + \log \frac{1}{\epsilon})$). When the relative-demand query prices are given as rational numbers, exact simulation is implied when $\log \epsilon$ is linear in the input length.

4 Demand Queries and Linear Programming

Consider the following linear-programming relaxation for the generalized winner-determination problem in combinatorial auctions (the "primal" program):

$$\begin{aligned}
\textbf{Maximize} \quad & \sum_{i \in N, S \subseteq M} w_i \, x_{i,S} \, v_i(S) & \\
\textbf{s.t.} \quad & \sum_{i \in N,\ S | j \in S} x_{i,S} \leq q_j & \forall j \in M \\
& \sum_{S \subseteq M} x_{i,S} \leq d_i & \forall i \in N \\
& x_{i,S} \geq 0 & \forall i \in N, S \subseteq M
\end{aligned}$$

Note that the primal program has an exponential number of variables. Yet, we will be able to solve it in polynomial time using demand queries to the bidders. The solution will have a polynomial size support (non-zero values for $x_{i,S}$), and thus we will be able to describe it in polynomial time.

Here is its dual:

$$\begin{aligned}
\textbf{Minimize} \quad & \sum_{j \in M} q_j p_j + \sum_{i \in N} d_i u_i & \\
\textbf{s.t.} \quad & u_i + \sum_{j \in S} p_j \geq w_i v_i(S) & \forall i \in N, \ \ S \subseteq M \\
& p_i \geq 0, \ \ u_j \geq 0 & \forall i \in M, \ j \in N
\end{aligned}$$

Notice that the dual problem has exactly $n+m$ variables but an exponential number of constraints. Thus, the dual can be solved using the Ellipsoid method in polynomial time – *if* a "separation oracle" can be implemented in polynomial time. Recall that a separation oracle, when given a possible solution, either confirms that it is a feasible solution, or responds with a constraint that is violated by the possible solution.

We construct a separation oracle for solving the *dual* program, using a single demand query to each of the bidders. Consider a possible solution $(\overline{u}, \overline{p})$ for the dual program. We can re-write the constraints of the dual program as:

$$u_i / w_i \geq v_i(S) - \sum_{j \in S} p_j / w_i$$

Now a demand query to bidder i with prices p_j / w_i reveals exactly the set S that maximizes the RHS of the previous inequality. Thus, in order to check whether $(\overline{u}, \overline{p})$ is feasible it suffices to (1) query each bidder i for his demand D_i under the prices p_j / w_i; (2) check only the n constraints $u_i + \sum_{j \in D_i} p_j \geq w_i v_i(D_i)$ (where $v_i(D_i)$ can be simulated using a polynomial sequence of demand queries as shown in Lemma 2). If none of these is violated then we are assured that $(\overline{u}, \overline{p})$ is feasible; otherwise we get a violated constraint.

What is left to be shown is how the *primal* program can be solved. (Recall that the primal program has an exponential number of variables.) Since the Ellipsoid algorithm runs in polynomial time, it encounters only a polynomial number of constraints during its operation. Clearly, if all other constraints were removed from the dual program, it would still have the same solution (adding constraints can only decrease the space of feasible solutions). Now take the "reduced dual" where only the constraints encountered exist, and look at its dual. It will have the same solution as the original dual and hence of the original primal. However, look at the form of this "dual of the reduced dual". It is just a version of the primal program with a polynomial number of variables – those corresponding to constraints that remained in the reduced dual. Thus, it can be solved in polynomial time, and this solution clearly solves the original primal program, setting all other variables to zero.

An Approximation Algorithm:

Initialization: Let $T \leftarrow M$ be the current items for sale.
Let $K \leftarrow N$ be the currently participating bidders.
Let $S_1^* \leftarrow \emptyset, ..., S_n^* \leftarrow \emptyset$ be the provisional allocation.

Repeat until $T = \emptyset$ **or** $K = \emptyset$**:**
Ask each bidder i in K for the bundle S_i that maximizes her per-item value, i.e., $S_i \in argmax_{S \subseteq T} \frac{v_i(S)}{|S|}$.
Let i be the bidder with the maximal per-item value, i.e., $i \in argmax_{i \in K} \frac{v_i(S_i)}{|S_i|}$, and set: $s_i^* = s_i, \quad K = K \setminus i, \quad M = M \setminus S_i$

Finally: Ask the bidders for their values $v_i(M)$ for the grand bundle.
If allocating all the items to some bidder i improves the social welfare achieved so far (i.e., $\exists i \in N$ such that $v_i(M) > \sum_{i \in N} v_i(S_i^*)$), then allocate all items to this bidder i.

Figure 4: This algorithm achieves a $\min\{n, 4\sqrt{m}\}$-approximation for the social welfare, which is asymptotically the best worst-case approximation possible with polynomial communication. This algorithm can be implemented with a polynomial number of demand queries.

5 Approximating the Social Welfare with Value and Demand Queries

We know from [23] that iterative combinatorial auctions that only use a polynomial number of queries can not find an optimal allocation among general valuations and in fact can not even approximate it to within a factor better than $min\{n, m^{1/2-\epsilon}\}$. In this section we ask how well can this approximation be done using demand queries with item prices, or using the weaker value queries. We show that, using demand queries, the lower bound can be matched, while value queries can only do much worse.

Figure 4 describes a polynomial-time algorithm that achieves a $min(n, O(\sqrt{m}))$ approximation ratio. This algorithm greedily picks the bundles that maximize the bidders' per-item value (using "relative-demand" queries, see Section 3). As a final step, it allocates all the items to a single bidder if it improves the social welfare (this can be checked using value queries). Since both value queries and relative-demand queries can be simulated by a polynomial number of demand queries with item prices (Lemmas 2 and 5), this algorithm can be implemented by a polynomial number of demand queries with item prices.[14]

Theorem 1. *The auction described in Figure 4 can be implemented by a polynomial number of demand queries and achieves a* $\min\{n, 4\sqrt{m}\}$*-approximation for the social welfare.*

We now ask how well can the optimal welfare be approximated by a polynomial number of *value queries*. First we note that value queries are not completely powerless: In [13] it is shown that if the m items are split into k fixed bundles of size m/k each, and these fixed bundles are auctioned as though each was indivisible, then the social welfare generated by such an auction is at least $\frac{m}{\sqrt{k}}$-approximation of that possible in the original auction. Notice that such an auction can be implemented by $2^k - 1$ value queries to each bidder – querying the value of each bundle of the fixed bundles. Thus, if we choose $k = \log m$ bundles we get an $\frac{m}{\sqrt{\log m}}$-approximation while still using a polynomial number of queries.

We show that not much more is possible using value queries:

Lemma 6. *Any iterative auction that uses only value queries and distinguishes between k-tuples of 0/1 valuations where the optimal allocation has value 1, and those where the optimal allocation has value k requires at least $2^{\frac{m}{k}}$ queries.*

We conclude that a polynomial time protocol that uses only value queries cannot obtain a better than $O(\frac{m}{\log m})$ approximation of the welfare:

[14] In Figure 5 in the appendix, we show an implementation of this algorithm by two descending-price auctions (where we allow removing items during the auction).

Theorem 2. *An iterative auction that uses a polynomial number of value queries cannot achieve better than $O(\frac{m}{\log m})$-approximation for the social welfare.*

6 The Representation of Bundle Prices

In this section we explicitly fix the language in which bundle prices are presented to the bidders in bundle-price auctions. This language requires the algorithm to explicitly list the price of every bundle with a non-trivial price. "Trivial" in this context is a price that is equal to that of one of its proper subsets (which was listed explicitly). This representation is equivalent to the XOR-language for expressing valuations. Formally, each query q is given by an expression: $q = (S_1 : p_1) \oplus (S_2 : p_2) \oplus ... \oplus (S_l : p_l)$. In this representation, the price demanded for every set S is simply $p(S) = max_{\{k=1...l|S_k \subseteq S\}} p_k$.

Definition 2. The *length* of the query $q = (S_1 : p_1) \oplus (S_2 : p_2) \oplus ... \oplus (S_l : p_l)$ is l. The *cost* of an algorithm is the sum of the lengths of the queries asked during the operation of the algorithm on the worst case input.

Note that under this definition, bundle-price auctions are not necessarily stronger than item-price auctions. An item-price query that prices each item for 1, is translated to an exponentially long bundle-price query that needs to specify the price $|S|$ for each bundle S. But perhaps bundle-price auctions can still find optimal allocations whenever item-price auction can, without directly simulating such queries? We show that this is not the case: indeed, when the representation length is taken into account, bundle price auctions are sometimes seriously inferior to item price auctions.

Consider the following family of valuations: Each item is valued at 3, except that for some single set S, its value is a bit more: $3|S| + b$, where $b \in \{0, 1, 2\}$. Note that an item price auction can easily find the optimal allocation between any two such valuations: Set the prices of each item to $3 + \epsilon$; if the demand sets of the two players are both empty, then $b = 0$ for both valuations, and an arbitrary allocation is fine. If one of them is empty and the other non-empty, allocate the non-empty demand set to its bidder, and the rest to the other. If both demand sets are non-empty then, unless they form an exact partition, we need to see which b is larger, which we can do by increasing the price of a single item in each demand set.

We will show that any bundle-price auction that uses only the XOR-language to describe bundle prices requires an exponential cost (which includes the sum of all description lengths of prices) to find an optimal allocation between any two such valuations.

Lemma 7. *Every bundle-price auction that uses XOR-expressions to denote bundle prices requires $2^{\Omega(\sqrt{m})}$ cost in order to find the optimal allocation among two valuations from the above family.*

The complication in the proof stems from the fact that using XOR-expressions, the length of the price description depends on the number of bundles whose price is strictly larger than each of their subsets – this may be significantly smaller than the number of bundles that have a non-zero price. (The proof becomes easy if we require the protocol to explicitly name every bundle with non-zero price.) We do not know of any elementary proof for this lemma (although we believe that one can be found). Instead we reduce the problem to a well known lower bound in boolean circuit complexity [12] stating that boolean circuits of depth 3 that compute the majority function on m variables require $2^{\Omega(\sqrt{m})}$ size.

References

[1] L. M. Ausubel and P. R. Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1:1–42, 2002.

[2] Lawrence Ausubel. An efficient dynamic auction for heterogeneous commodities, 2000. Working paper, University of Maryland.

[3] Alejandro Bertelsen. Substitutes valuations and m$^\natural$-concavity". M.Sc. Thesis, The Hebrew University of Jerusalem, 2005.

[4] Avrim Blum, Jeffrey C. Jackson, Tuomas Sandholm, and Martin A. Zinkevich. Preference elicitation and query learning. *Journal of Machine Learning Research*, 5:649–667, 2004.

[5] Liad Blumrosen and Noam Nisan. On the computational power of iterative auctions II: ascending auctions. Working paper, The Hebrew University of Jerusalem. Available from http://www.cs.huji.ac.il/~noam/mkts.html.

[6] P. Cramton, L.M. Ausubel, and P.R. Milgrom. *In P. Cramton and Y. Shoham and R. Steinberg (Editors), Combinatorial Auctions. Chapter 5. The Clock-Proxy Auction: A Practical Combinatorial Auction Design.* MIT Press. Forthcoming, 2005.

[7] P. Cramton, Y. Shoham, and R. Steinberg (Editors). *Combinatorial Auctions.* MIT Press. Forthcoming, 2005.

[8] G. Demange, D. Gale, and M. Sotomayor. Multi-item auctions. *Journal of Political Economy*, 94:863–872, 1986.

[9] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for cas with complement-free bidders. In *The 37th ACM symposium on theory of computing (STOC).*, 2005.

[10] Shahar Dobzinski and Michael Schapira. Optimal upper and lower approximation bounds for k-duplicates combinatorial auctions. Working paper, the Hebrew University.

[11] Faruk Gul and Ennio Stacchetti. The english auction with differentiated commodities. *Journal of Economic Theory*, 92(3):66 – 95, 2000.

[12] J. Hastad. Almost optimal lower bounds for small depth circuits. In *18th STOC*, pages 6–20, 1986.

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

[14] H. Karloff. *Linear Programming.* Birkhäuser Verlag, 1991.

[15] A.S. Kelso and V.P. Crawford. Job matching, coalition formation, and gross substitute. *Econometrica*, 50:1483–1504, 1982.

[16] Sebastien Lahaie and David C. Parkes. Applying learning algorithms to preference elicitation. In *EC 04.*

[17] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *ACM conference on electronic commerce. To appear, Games and Economic Behaviour.*, 2001.

[18] 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.

[19] A. Mas-Collel, W. Whinston, and J. Green. *Microeconomic Theory.* Oxford university press, 1995.

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

[21] Noam Nisan. Bidding and allocation in combinatorial auctions. In *ACM Conference on Electronic Commerce*, 2000.

[22] Noam Nisan. *In P. Cramton and Y. Shoham and R. Steinberg (Editors), Combinatorial Auctions. Chapter 1. Bidding Languages.* MIT Press. Forthcoming, 2005.

[23] 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 Forthcoming in the Journal of Economic Theory.

[24] Noam Nisan and Ilya Segal. Exponential communication inefficiency of demand queries, 2004. Working paper. Available from http://www.stanford.edu/ isegal/queries1.pdf.

[25] D. C. Parkes and L. H. Ungar. An ascending-price generalized vickrey auction. Tech. Rep., Harvard University, 2002.

[26] David Parkes. *In P. Cramton and Y. Shoham and R. Steinberg (Editors), Combinatorial Auctions. Chapter 3. Iterative Combinatorial Auctions.* MIT Press. Forthcoming, 2005.

[27] David C. Parkes and Lyle H. Ungar. Iterative combinatorial auctions: Theory and practice. In *AAAI/IAAI*, pages 74–81, 2000.

[28] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. In *Artificial Intelligence*, volume 135, pages 1–54, 2002.

[29] Ilya Segal. The communication requirements of social choice rules and supporting budget sets, 2004. Working paper. Available from http://www.stanford.edu/ isegal/rules.pdf.

A Proofs

A.1 The Power of Different Types of Queries

Proof for Lemma 1:

Proof. The simulation of a marginal value query by is value queries is direct from the definition: $v(j|S) = v(S \cup \{j\}) - v(S)$. The simulation of a value query S by $|S| \leq m$ marginal value queries is given by the equation $v(S) = \sum_{j \in S} v(j|\{j' \in S | j' < j\})$. □

Proof for Lemma 2:

Proof. We will show that demand queries can simulate any marginal value query $v(j|S)$ using t queries, and then invoke the previous lemma. Set the prices of all the items in S to zero, and the prices of all other items (except j) to ∞. Then, we perform a binary search on p_j to find its lowest value for which the bidder demands $v(S)$. It is straightforward to see that this price is indeed the marginal value of item j: at this price, the utilities from the bundles S and $S \cup \{j\}$ are equal, thus $v(S) - 0 = v(S \cup \{j\}) - p_j$ and the claim follows.

A binary search makes t demand queries, and m marginal value queries are needed to simulate a single value query thus $v(S)$ can be simulated by mt demand queries. □

Proof for Lemma 3:

Proof. We will actually show an example where a single demand query suffices for finding the optimal allocation, but an exponential number of value queries are required for that. Consider a player with a valuation of $2|S|$ for any bundle S, except for some "hidden" bundle H of size $\frac{m}{2}$ with a valuation of $2|S| + 2$, and a second player with a known valuation of $2|S| + 1$ for every bundle S. The only optimal allocation gives the hidden set H to the first bidder. In a demand query with a price of $2 + \epsilon$ for every item, the first bidder demands his "hidden" set, and thus reveals the optimal allocation.

However, consider any algorithm that uses only value queries. An adversary will answer each value query $v(S)$ to the first bidder with $v(S) = 2|S|$. As long as two sets S of size $\frac{m}{2}$ have not been queried any of them can be the hidden set H and the optimal allocation can not be determined. Thus, $\Omega 2^m$ value queries will be needed in the worst case. □

Proof for Lemma 4:

Proof. An indirect-utility query with prices $\overrightarrow{p}$ can be answered by first querying for the demand D under these prices and then simulating the value query $v(D)$.

The following algorithm uses $m+1$ indirect-utility queries to simulate a demand query with some price vector $\overrightarrow{p}$:
Initialization: start with the price vector $\overrightarrow{p}$ for which the player answers some utility x.
Repeat: for every item $i = 1, ..., m$, raise the price of item i by some $\epsilon \in (0, 1)$. If the answer to the indirect-utility query now is other than x, we decrease its price back by ϵ in all future queries. If the answer was x, we use the new price for i in all future queries.
Finally: After all the $m+1$ indirect-utility queries are done, return the bundle of all items for which the answer was changed when we increased their prices.

In the algorithm above, if we raised the price of some item i, and the reported maximal-utility did not change, then there would clearly be utility-maximizing bundles that do not contain i, thus we can ignore this item. If the maximal-utility changed, then any utility-maximizing bundle under the current prices clearly contains i, thus we include it in our answer. Leaving the price of item i (of the first kind) at $p_i + \epsilon$, ensures that any bundle that contains it will not be output (but we are guaranteed to have other utility-maximizing bundles). □

Proof for Lemma 5:

Proof. For any $\epsilon > 0$, we simulate $RD(\overrightarrow{p})$ by the following binary search (up to an ϵ, see below):
Initialization: start with a price vector $c\overrightarrow{p}$ $(c > 0)$.
Binary search: find with a binary search the value $c^* \in \Re^+$ for which the bidder has a non-empty demand for the price vector $c^* \cdot \overrightarrow{p}$ and the bidder demands the empty set for $(c^* + \epsilon) \cdot \overrightarrow{p}$.
Finally: return the bundle S demanded under the price vector $c^* \cdot \overrightarrow{p}$.

Now we show that for the price vector $\overrightarrow{p}$, every other bundle T has a smaller weight than S (up to ϵ), i.e.,

$$\frac{v(S)}{p(S)} \geq \frac{v(T)}{p(T)} - \epsilon. \tag{A.1}$$

Denote $c^* = \epsilon t$ for some $t \in \Re^+$. The bundle S was demanded under the prices $\epsilon t \cdot \overrightarrow{p}$, therefore $v(S) - \epsilon t p(S) \geq 0$. Thus, $\frac{v(S)}{p(S)} \geq \epsilon t$. Assume that inequality A.1 does not hold, then it follows that $\frac{v(T)}{p(T)} - \epsilon > \epsilon t$, or $v(T) > \epsilon(t+1)p(T)$. But for the price vector $(c^* + \epsilon)\overrightarrow{p} = \epsilon(t+1)\overrightarrow{p}$ no bundle achieved a positive utility. Contradiction. □

A.2 Approximating the Social Welfare with Value and Demand Queries

Proof for Theorem 1:

Proof. We first observe that the algorithm can be implemented by a polynomial number of value queries and relative demand queries: querying a bidder for the bundle that maximizes his per-item value is a relative-demand query when all the item prices are 1. Querying a bidder for his value for the grand bundle can be done by a value query. In Section 3 we show that any value query and any relative-demand query can be implemented by a polynomial number of demand queries. Each bidder is asked at most m relative demand queries, and exactly one value query, thus a polynomial number of demand queries can implement this algorithm.

Next, we prove that the algorithm achieves an approximation ratio of at least $\min\{n, 4\sqrt{m}\}$. The algorithm will clearly achieve a $\frac{1}{n}$-approximation since we allocate the whole bundle M to the bidder with the highest valuation if it improves the welfare achieved. Next, we prove that the algorithm achieves at least $\frac{1}{4\sqrt{m}}$ of the optimal welfare.

Let $OPT = \{T_1, ..., T_k, Q_1, ..., Q_l\}$ be an optimal allocation where for every $i \in \{1, ..., k\}$ $|T_i| \leq \sqrt{m}$ and for every $j \in \{1, ..., l\}$ $|Q_j| > \sqrt{m}$ $(l, k \in \{0, ..., n\})$. Let ALG be the allocation found by the algorithm, and let $v(OPT)$ and $v(ALG)$ be the optimal welfare and the welfare achieved by the algorithm, respectively. First, we analyze cases where "large" bundles contribute most of the optimal welfare, i.e.,

$$\sum_{i=1}^{l} v_i(Q_i) \geq \sum_{i=1}^{k} v_i(T_i)$$

Then,

$$v(OPT) \leq 2\sum_{i=1}^{l} v_i(Q_i) \leq 2\sum_{i=1}^{l} v(ALG) = 2l \cdot v(ALG) \leq 2\sqrt{m} \cdot v(ALG)$$

Where the first inequality holds since $v(OPT) = \sum_{i=1}^{l} v_i(Q_i) + \sum_{i=1}^{k} v_i(T_i)$ and the second holds since the last stage of the algorithm verifies that the welfare achieved by the algorithm is at least the valuation of every player for the whole bundle M. The last inequality holds since there are no more than $\sqrt{m}$ bundles of size of at least $\sqrt{m}$.

The analysis of the case where "small" bundles contribute most of the optimal welfare (i.e., $\sum_{i=1}^{l} v_i(Q_i) < \sum_{i=1}^{k} v_i(T_i)$) is more involved. Let $I \subseteq \{1, ..., k\}$ be the set of bidders that receives a "small" bundle (i.e., bundles in $\{T_1, ..., T_k\}$) in OPT that does not intersect any bundle in ALG. Consider the following sum:

$$\sum_{i=1}^{k} \frac{v_i(T_i)}{|T_i|} = \sum_{i \in I} \frac{v_i(T_i)}{|T_i|} + \sum_{i \in \{1,...,k\} \setminus I} \frac{v_i(T_i)}{|T_i|} \tag{A.2}$$

In the two claims below, we show that each of the summands in the right side of Equation A.2 is not greater than $v(ALG)$. This immediately derives that $\sum_{i=1}^{k} \frac{v_i(T_i)}{|T_i|} \leq 2 \cdot v(ALG)$. Since for every $i \in 1, ..., k,\ |T_i| \leq \sqrt{m}$:

$$\sum_{i=1}^{k} v_i(T_i) \leq 2\sqrt{m} \cdot v(ALG)$$

Since most of the optimal welfare is contributed by "small" bundles,

$$v(OPT) \leq 2\sum_{i=1}^{k} v_i(T_i) \leq 4\sqrt{m} \cdot v(ALG)$$

What is left to be proved is that both summands in Equation A.2 are not greater than $v(ALG)$:

Claim 1.

$$\sum_{i \in I} \frac{v_i(T_i)}{|T_i|} \leq v(ALG)$$

Proof. Consider a bidder i that receives a small bundle T_i in OPT such that T_i is disjoint to all bundles in ALG. We observe that this bidder surely receives a non-empty bundle S_i in ALG. This holds since the items in T_i are not allocated at the end of the algorithm (they are not in any bundle in ALG), but player i has a non-zero value for T_i.

Since the algorithm picked some $S_i \in ALG$ and not T_i for bidder i, $\frac{v_i(T_i)}{|T_i|} \leq \frac{v_i(S_i)}{|S_i|}$. Therefore,

$$\sum_{i \in I} \frac{v_i(T_i)}{|T_i|} \leq \sum_{i \in I} \frac{v_i(S_i)}{|S_i|} \leq \sum_{i \in I} v_i(S_i) \leq \sum_{i=1}^{n} v_i(S_i) = v(ALG)$$

□

Claim 2.

$$\sum_{i\in\{1,...,k\}\setminus I} \frac{v_i(T_i)}{|T_i|} \leq v(ALG)$$

Proof. For every bidder $i \in \{1,...,k\} \setminus I$, T_i intersects at least one bundle from ALG, and let $F(i)$ be the first bidder for which the algorithm allocates a bundle that intersects T_i. Then,

$$\sum_{i\in\{1,...,k\}\setminus I} \frac{v_i(T_i)}{|T_i|} \leq \sum_{j=1}^{n} \sum_{i|F(i)=j} \frac{v_i(T_i)}{|T_i|} \leq \sum_{j=1}^{n} \sum_{i|F(i)=j} \frac{v_i(S_j)}{|S_j|} \leq \sum_{j=1}^{n} |S_j| \frac{v_i(S_j)}{|S_j|} \leq \sum_{j\in ALG} v_j(S_j)$$

Where the second leftmost inequality holds since bidder $j = F(i)$ demands $S_j \in ALG$ when all the items in T_i are still on sale and the third inequality holds since each S_j intersects at most $|S_j|$ bundles from $\{T_1,...,T_k\}$ (all T_i's are disjoint). □

We showed before how the theorem follows from these two claims. □

Proof for Lemma 6:

Proof. Consider the following family of valuations: for every S, such that $|S| > m/2$, $v(S) = 1$, and there exists a single set T, such that for $|S| \leq m/2$, $v(S) = 1$ iff $T \subseteq S$ and $v(S) = 0$ otherwise. Now look at the behavior of the protocol when all valuations v_i have $T = \{1...m\}$. Clearly in this case the value of the best allocation is 1 since no set of size $\frac{m}{2}$ or lower has non-zero value for any player. Fix the sequence of queries and answers received on this k-tuple of valuations.

Now consider the k-tuple of valuations chosen at random as follows: a partition of the m items into k sets $T_1...T_k$ each of size $\frac{m}{k}$ each is chosen uniformly at random among all such partitions. Now consider the k-tuple of valuations from our family that correspond to this partition – clearly T_i can be allocated to i, for each i, getting a total value of k. Now look at the protocol when running on these valuations and compare its behavior to the original case. Note that the answer to a query S to player i differs between the case of T_i and the original case of $T = \{1...m\}$ only if $|S| \leq \frac{m}{2}$ and $T_i \subseteq S$. Since T_i is distributed uniformly among all sets of size exactly $\frac{m}{k}$, we have that for any fixed query S to player i, where $|S| \leq \frac{m}{2}$,

$$Pr[T_i \subseteq S] \leq \left(\frac{|S|}{m}\right)^{|T_i|} \leq 2^{-\frac{m}{k}}$$

Using the union-bound, if the original sequence of queries was of length less than $2^{\frac{m}{k}}$, then with positive probability none of the queries in the sequence would receive a different answer than for the original input tuple. This is forbidden since the protocol must distinguish between this case and the original case – which cannot happen if all queries receive the same answer. Hence there must have been at least $2^{\frac{m}{k}}$ queries for the original tuple of valuations. □

Proof for Theorem 2:

Proof. Immediate from Lemma 6: achieving any approximation ratio k which is asymptotically greater than $\frac{m}{\log m}$ needs an exponential number of value queries. □

A.3 The Representation of Bundle Prices

Proof for Lemma 7:

Proof. Consider the protocol running on the following two valuations: the first has $b = 0$ (i.e. is simply additive), and the second has $b = 1$ for the set S of all items. In this case the outcome must be to allocate all to the second bidder. Let $e_1...e_t$ be the queries made on this input, where each $e_i = E_i^1 \oplus E_i^2 \oplus ... \oplus E_i^{l_i}$. Now consider what happens when the first valuation is changed so that for some S of size exactly $m/2$, be get a bonus $b = 2$ – clearly the allocation must change so that this S is allocated to the first player – hence one of the queries $e_1...e_t$ must change its answer. We will see that the fact that this is true for every such S implies that $\sum_{i=1}^{t} l_i$ is exponential.

First note that if in e_i there exists some set of size $m/2 + 1$ that has price zero, then the answer will not change as this set will give a surplus of at least $3m/2 + 3$ as opposed to at most $3m/2 + 2$ that S gives. Let us focus at an e_i that does not have such a set. We build a boolean DNF formula from this expression as follows: the variable set will be $x_1...x_m$ – a variable for each item. Consider a term (atomic bid) $E_i^j = (B_i^j, p_i^j)$ in e_i. We call this term essential if there exists some bundle of size exactly $m/2 + 1$ whose price in e_i is exactly p_i^j. For every *essential* term (B_i^j, p_i^j) in e_i we build a conjunction of the variables in it (ignoring the price for this bundle). We then take the disjunction of all of these conjunctions. First notice that this DNF must accept all inputs with more than $m/2$ 1's in the input – since otherwise consider a set that is not accepted by this expression, and the value of this set in e_i must be zero.

Now notice that if an input with 1's in exactly the set S of size exactly $m/2$ is accepted by this formula, then the answer to query e_i will not change. The reason is that an accepted set S contains some essential bundle B_i^j, and thus its price in e_i would be at least p_i^j. However, since the bundle is essential, there exists some set of size $m/2 + 1$ that is priced at exactly p_i^j – this set would clearly be preferable to S – the only set whose value has changed. Since for *every* set S of size exactly $m/2$ the answer to one of the queries must change, at least one of the formulas constructed must reject the input with 1's exactly in S.

We now take the conjunction of all boolean expressions built for all i. This formula accepts all inputs with exactly $m/2+1$ 1's, and rejects all inputs with exactly $m/2$ 1's. Note that this formula is a conjunction of disjunctions of conjunctions of variables – a, so called, monotone depth 3 formula. Since it is a monotone formula, it computes the majority function. Its size is clearly bounded from above by the total length of all expressions e_i. We are now ready to invoke the well known lower bound that states that a depth 3 formula for majority must have size at least $2^{\Omega(\sqrt{m})}$. □

An auction with two (almost) descending trajectories:

Stage 1:

Initialization: set all item prices to L.

Repeat: At each stage, decrease all prices by $\epsilon < \frac{\delta}{m^2}$. When some bidder i demands some bundle S, provisionally allocate the items in S to i. Remove the items in S from the auction, and continue similarly without bidder i until all items are provisionally allocated.

Stage 2:

Initialization: set all item prices to L.

Repeat: At each stage, decrease the price of one item by δ, in a round-robin fashion.

Finally: Compute the valuation of each bidder for the whole bundle M according to the data collected by this auction (see [5]). Allocate the whole bundle to the bidder with the highest valuation for it if it improves the welfare.

Figure 5: This is an item-price demand-query implementation of the approximation algorithm described in Theorem 1 using two descending-price auctions (in one of them, we remove sold items during the auction).