On Profit-Maximizing Envy-free Pricing

Venkatesan Guruswami* Jason D. Hartline† Anna R. Karlin* David Kempe‡
Claire Kenyon§ Frank McSherry†

Abstract

We study the problem of pricing items for sale to consumers so as to maximize the seller's revenue. We assume that for each consumer, we know the maximum amount he would be willing to pay for each bundle of items, and want to find pricings of the items with corresponding allocations that maximize seller profit and at the same time are *envy-free*, which is a natural fairness criterion requiring that consumers are maximally happy with the outcome they receive given the pricing. We study this problem for two important classes of inputs: *unit demand consumers*, who want to buy at most one item from among a selection they are interested in, and *single-minded consumers*, who want to buy one particular subset, but only if they can afford it.

We show that computing envy-free prices to maximize the seller's revenue is APX-hard in both of these cases, and give a logarithmic approximation algorithm for them. For several interesting special cases, we derive polynomial-time algorithms. Furthermore, we investigate some connections with the corresponding mechanism design problem, in which the consumer's preferences are private values: for this case, we give a $\log$-competitive truthful mechanism.

1 Introduction

Imagine that we are a company or store in the business of selling products to consumers. An important aspect of maximizing the revenue obtained is the pricing of our products: a low price will attract more customers, while a high price generates more revenue per sold item. How, then, should we choose prices optimally? For example, supposing that customers want to buy bandwidth along subpaths of a network and are willing to pay up to some amount for the bandwidth, how should one price the bandwidth along the links so as to maximize the revenue? This is the flavor of the pricing problems studied in this paper.

More formally, we assume that the seller has m different items, and a set of n consumers may be interested in purchasing some of these items. We assume that through market research or interaction with the consumers, the seller knows each customer's *valuation* for each subset (also called *bundle*) of items, the largest amount that the customer is willing to pay for that subset. If a customer buys a subset, his *utility* is the difference between the valuation and the purchase price, i.e., the amount of money he "saved" compared to his valuation.

The seller gets to assign individual *prices* to the items, and his goal is to maximize his own revenue, i.e., the sum of prices of all sold items. Which items are sold is determined by an *allocation* of bundles of items to customers (at most one bundle is allocated to each customer, and such a bundle, if any, must have nonnegative utility for that customer, since otherwise the customer will of course not buy the bundle). We require that this allocation be *envy-free*, i.e., given the pricing, no user would prefer to be assigned a different bundle. The notions of envy-free pricing and allocations model fair equilibrium pricing in a variety of economic settings [24]. Envy-freeness is particularly relevant in the case when there is only a limited supply of some items; for then, a user interested in buying a bundle at the posted price may be unable to do so, creating discontent among the customers.

1.1 Our Results.

The pricing problem as defined above is very general. Even when given envy-free prices, computing the corresponding allocation problem can be easily seen to be NP-hard. For this reason, we focus here on two important classes of consumer valuation profiles.

- *Unit-demand bidders*: Each consumer would like to buy at most one item, and is considering a number of different options with different valuations for each. For instance, the items may be houses, and the customer may be willing to pay a higher price for some than for others based on location, size, or other qualities.
- *Single-minded (or single parameter) bidders*: Each consumer is interested only in one particular bundle. He will buy the bundle if it is sufficiently cheap, and otherwise not buy anything.

*Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195. Email: {venkat,karlin}@cs.washington.edu.

†Microsoft Research Silicon Valley, Mountain View, CA 94043. Email: {hartline,mcsherry}@microsoft.com.

‡Computer Science Department, University of Southern California, Los Angeles, CA 90089. Email: dkempe@usc.edu.

§Department of Computer Science, Brown University, Providence, RI 02912. Email: claire@cs.brown.edu.

An interesting special case of these pricing problems can be obtained by assuming that the items are available in *unlimited supply*. In this special case, there is no limitation on how many copies of each item are sold, and thus any pricing is envy-free when the allocation is to give each consumer their most preferred bundle. When we wish to distinguish the unlimited supply case from the general case we will refer to the general case as the *limited supply* case.

As we will show, both of these versions are not only NP-complete, but APX-hard, even under strong additional restrictions. We therefore focus on two kinds of results: (1) approximation algorithms with logarithmic guarantees in the general case, and (2) polynomial or pseudo-polynomial time algorithms for interesting special cases.

We give an $O(\log n)$ approximation algorithm for the limited supply unit demand problem, and an $O(\log n + \log m)$ approximation for the unlimited supply single-minded bidder problem.

Furthermore, we define and give a polynomial time algorithm for the unlimited supply *pricing over time* variant of the unit demand problem. This problem models the case where there is a single item for sale at various points in time and each consumer wants to acquire the item within some time interval (e.g., for bandwidth or airline tickets).

For the single-minded case, we define and investigate, in Section 5.2, the *Tollbooth problem* on trees, and its special case, the *Highway problem* on a path. In these problems, consumers are interested in using a given path in the tree (or on the path), and the seller can place toll booths on the edges and charge prices for their usage. The additional combinatorial structure provided by this restriction allows us to give polynomial time algorithms for several cases of these problems.

The envy-free pricing problem also has a role as a lower bound in the setting of *profit maximizing combinatorial auctions*. There, customers' valuations are often not known, and what is more, customers will choose to misrepresent their valuations if they feel that they can get a better deal. *Mechanism design* studies the design of *truthful* (or *incentive-compatible*) auctions which make it in the customers' best interest to disclose their true valuations. The performance of such auctions is often measured in comparison to the profit of the seller optimal envy-free pricings [13, 12],. In Section 4, we will investigate this connection further, and present a $\log h$-competitive truthful limited-supply unit-demand combinatorial auction for the case where all consumer valuations are in the range $[1, h]$. We note that for any unlimited supply combinatorial pricing problem (including the single-minded case), a $\log h$-competitive truthful mechanism is trivial.

Related Work. Pricing is a well-studied area in economics; however, computational issues are not a major focus. Envy-free pricing is part of a trend towards understanding the algorithmic complexity of computing "equilibria": in an optimal envy-free pricing, the seller does not have any incentive to change prices and the consumers do not have any incentive to dispute the allocation. Examples of other, somewhat different work in this category, can be found in [9, 10].

As previously discussed, this work is related to truthful auction design, and especially the design of truthful auctions with worst-case performance guarantees, e.g., [12, 13, 11, 5, 4, 18].

Our interest here is in seller profit maximization. Another common goal for economically motivated algorithms and mechanisms is that of *economic efficiency*: obtaining an outcome which maximizes the sum of the utilities of all participants. For the single-minded bidder case, when the goal is efficiency, both the pricing problem and the mechanism design problem have approximate solutions [19, 3]. For the unit-demand case, the Vickrey-Clarke-Groves (VCG) mechanism [23, 8, 14] solves the problem in polynomial time (see details in Section 4), and is both truthful and envy-free. We note that computing the VCG mechanism is NP-hard for general combinatorial auction problems including the special case of single-minded bidders. The goal of economic efficiency is quite different from maximizing the seller profit, and there are simple examples where the VCG mechanism is very far from maximizing the seller profit. For example, if no two consumers want the same item then the VCG mechanism gives all items away for free!

For unit-demand envy-free pricing, a final, somewhat informal connection is to the notion of stable matchings. By introducing prices, we construct matchings of consumers to items with the property that no consumer prefers the item someone else gets to the item they get.

2 Preliminaries

We assume that there are n consumers, and a set J of m distinct items. Each item $j \in J$ exists in c_j copies (we explicitly allow $c_j = \infty$), and we denote the *supply vector* by $\mathbf{c} = (c_1, \ldots, c_m)$. Each consumer has a *valuation* $v_i(S)$ for each bundle $S \subseteq J$ of items, which measures how much receiving bundle S would be "worth" to consumer i; we denote by V the $n \times 2^J$ matrix of valuations. For convenience, we assume that $v_i(\emptyset) = 0$ for all consumers i.

Given a *price vector* $\mathbf{p} = (p_1, \ldots, p_m)$, the *utility* that consumer i derives from bundle S is $U_i(S) = v_i(S) - p_S$, where $p_S = \sum_{j \in S} p_j$; it measures the consumer's "joy" at having bought the bundle S at the given price. If consumer i's utility for the bundle S is non-negative, we call S *feasible* for i. User i's *demand set* D_i contains all bundles that would make him maximally happy, i.e., all bundles that he would most like to buy. Formally, $D_i = \{S \mid U_i(S) = \max_{S'} U_i(S')\}$. Because not buying any bundle is always an option with utility $U_i(\emptyset) = 0$, we know that $U_i(S) \geq 0$ for all $S \in D_i$.

Using this terminology, we can now define envy-free pricing and allocations, the central notion of this paper.

DEFINITION 2.1.

1. *An allocation $(S_1, \ldots, S_n)$ of bundles to consumers is* feasible *if each item j is in at most c_j sets S_i. Notice that this may leave some items unallocated.*
2. *Given a pricing $\mathbf{p} = (p_1, \ldots, p_m)$, an allocation $(S_1, \ldots, S_n)$ is* envy-free *if $S_i \in D_i$ for all i, i.e., each consumer receives a bundle from his demand set.*
3. *A pricing $\mathbf{p}$ is* envy-free *if it admits a feasible, envy-free allocation.*

Notice that if the supply is unlimited for all items, then every price vector is envy-free, as we can select an arbitrary bundle S_i from each demand set D_i, and obtain a feasible allocation.

DEFINITION 2.2. The envy-free pricing problem: *Given the input $(n, m, V, \mathbf{c})$, compute an envy-free pricing $\mathbf{p}$ and a corresponding envy-free allocation $(S_1, \ldots, S_n)$ maximizing the* seller profit $\sum_i p_{S_i}$.

When we talk about *pricing algorithms* for computing prices and allocations, we will often use the notation $\text{Algo}(V)$ to denote both the output $(\mathbf{p}, (S_1, \ldots, S_n))$ of the algorithm and its revenue $R = \sum_i p_{S_i}$.

The most general case of the combinatorial envy-free pricing problem is difficult for two reasons, (1) the mere specification of V could be of size $\Omega(2^m)$ rendering the problem intractable, and (2) even if V was a reasonable size, the question of whether a given pricing is envy-free is NP-hard to decide. Thus we are motivated to focus on more tractable, yet important and rich subclasses of the problem. The general version can be considered as an OR of ANDs: the each user wants to buy one of several bundles, and buying a bundle means buying all of its items. Two natural special cases can be obtained by conceptually making the "fan-in" of the AND/OR operators 1. Specifically, we then obtain:

- *Unit-demand consumers:* Each consumer is interested in buying exactly one item, so $v_i(S) > 0$ only when $|S| = 1$. Thus, the size of the valuation matrix reduces from $n \times 2^m$ to $n \times m$, and the entry $v_i(j)$ denotes consumer i's valuation for item j. (This corresponds to making the AND fan-in equal to 1.)
- *Single-minded consumers:* Each consumer i is interested in only one specific bundle of items S_i. Thus, the valuations V can be summarized by a set of pairs (v_i, S_i) meaning that consumer i is interested in bundle S_i, and values it at v_i. (This corresponds to making the OR fan-in equal to 1.)

3 Pricing for Unit-Demand Consumers

In this section, we consider envy-free pricing in the unit-demand setting.

3.1 APX-hardness. Even in a very restricted special case of the unit-demand setting, finding optimal prices is APX-hard, as stated by the following theorem. A similar result was proven independently in [1].

THEOREM 3.1. *The unit demand envy-free pricing problem is APX-hard, even if each item exists in unlimited supply, and each consumer has equal valuations (of either 1 or 2) for all the items he has any interest in.*

Proof. We give a reduction from the vertex cover problem on graphs of maximum degree at most B for an absolute constant B, which is known to be APX-hard (even for $B = 3$). Given a graph $G = (V, E)$ with n nodes (labeled $1, 2, \ldots, n$) and m edges (where $m \leq Bn/2 = O(n)$), our pricing instance has n items, one for each node of G, and $m + n$ consumers. (We assume G is connected w.l.o.g, so $m \geq n - 1$.) For each edge $e = (i, j)$, there is a consumer c_e whose valuation for items i, j equals 1 and valuation for all other items is 0. In addition, for each $i = 1, 2, \ldots, n$, there is a customer d_i whose valuation for item i equals 2, and valuation for all other items equals 0.

We claim that the optimal pricing and corresponding allocation of this instance achieves a total profit of $m+2n-k$ where k is the size of the smallest vertex cover of G. First, if S is a vertex cover of G with $|S| = k$, consider the pricing where items in S have cost 1, and those in $V \setminus S$ have cost 2. Since each edge has at least one endpoint incident on a node in S, we get profit 1 from each customer c_e. Also, for $i \notin S$, we get profit 2 from d_i, and for $i \in S$ we get profit 1 from d_i. Clearly, this yields a total profit of $m + 2(n - k) + k = m + 2n - k$.

For the converse, first notice that each node/item is without loss of generality priced at 1 or 2. If there exists an edge $e = (i, j)$ where both i, j are priced 2, then we are making zero profit from c_e. By reducing the price of say i to 1, we lose a profit of 1 from the customer d_i, but we make up for it by making a profit of 1 from c_e. Therefore, we may assume that set of nodes priced at 1 forms a vertex cover of G. This in turn implies that the profit is at most $m + 2n - k$ if k is the size of the smallest vertex cover.

Since $m = \Theta(n)$ and the minimum vertex cover has size at least $m/B = \Omega(n)$, a constant factor gap in the size vertex cover translates into a constant factor gap in the optimal profit for the pricing instance, which yields the desired APX-hardness result. 0pt

3.2 A logarithmic approximation algorithm. In our further discussion of the unit demand case, it will be helpful to think of allocations as matchings in bipartite graphs.

Specifically, given a price vector $\mathbf{p}$, the *demand graph* is a bipartite graph between consumers i and items j, duplicated $\min(c_j,n)$ times, containing an edge (i,j) if and only if $j \in D_i$. An *envy-free matching* is a matching M such that each consumer i with $\emptyset \notin D_i$ is matched. Then, the prices $\mathbf{p}$ are envy-free if and only if the demand graph has an envy-free matching, and the output of a pricing algorithm is the pair $(\mathbf{p}, M)$.

To simplify the presentation, we assume that there is exactly one copy of each item, i.e., $c_j = 1$ for all items j. For the unit-demand case, this assumption is without loss of generality: if there are c_j copies of an item, we can replace them in the input by $\min(c_j,n)$ distinct items, and give each user the same valuation of all of those distinct items. The envy-free condition then guarantees that all of these distinct copies of an item will have the same price.

3.2.1 Walrasian Equilibria. Our approximation algorithm builds on work in the economics literature concerning Walrasian Equilibria [17]. Given a valuation matrix V, a *Walrasian Equilibrium* $(\mathbf{p}, M)$ consists of an envy-free pricing $\mathbf{p}$ and a matching M such that all unmatched items have price zero. The following theorem, due to Gul and Stacchetti [15], characterizes Walrasian Equilibria in the unit-demand pricing problem.

THEOREM 3.2. *[15] Let $(\mathbf{p}, M)$ be a Walrasian Equilibrium. Then M is a maximum weight matching on the valuation matrix V; furthermore, for any maximum matching M', $(\mathbf{p}, M')$ is also a Walrasian equilibrium.*

For a valuation matrix V, we let $\omega(V)$ denote the weight of a maximum weight matching $\mathrm{MM}(V)$. For an item j, let V_{-j} denote the valuation matrix with item j removed, i.e., the matrix obtained by deleting column j from V. The following algorithm finds the Walrasian Equilibrium with the highest prices.

Algorithm MaxWEQ**: Maximum Walrasian Prices.**
Input: Valuation matrix V.
For each item j, let $\hat{p}_j = \omega(V) - \omega(V_{-j})$.
Output: $\hat{\mathbf{p}}$ and $\mathrm{MM}(V)$.

THEOREM 3.3. *[15] The algorithm* MaxWEQ *outputs, in polynomial time, a Walrasian Equilibrium which maximizes the item prices: if $\mathbf{p}$ is any Walrasian equilibrium, then $p_j \le \hat{p}_j$ for every item j.*

Note that the result [15] is based on the properties of monotonicity and "single-improvement" of the utility functions, properties which, as they mention, are obviously satisfied in unit demand utility functions. [20] was the first to define the prices given in Theorem 3.3 but did not argue that they were maximum.

Unfortunately, even the Walrasian Equilibrium with the highest prices can have revenue far from optimum. The problem is that selling as many items as possible, a key requirement of a Walrasian Equilibrium, may lead to very low revenue.

Our algorithm will be following a common approach to profit maximization in economics. We will augment the computation of the Walrasian equilibrium with *reserve prices*. A reserve price constrains the set of feasible pricings, by requiring that item j be priced at least at r_j. A classic example of the use of reserve prices in economics is in the Bayesian optimal auction [21, 7], where the reserve prices are based on the known prior distribution from which the consumers' valuations are drawn.

DEFINITION 3.1. *Given a valuation matrix V and a reserve price vector $\mathbf{r} = (r_1, \dots, r_m)$, a* Walrasian Equilibrium with reserve prices $\mathbf{r}$ *is an envy-free pricing* $\mathbf{p}$ *and allocation M such that (1) $p_j \ge r_j$ for all j, (2) if item j is not sold, then $p_j = r_j$, and (3) if item j is in the demand set of bidder i and j is not sold, then bidder i is allocated an item.*

Given an algorithm, Algo, to compute Walrasian Equilibria, we can use it to derive an algorithm, $\mathrm{Algo}_{\mathbf{r}}$, for computing Walrasian Equilibria with reserve prices.

We augment the valuation matrix V to a new matrix V' by creating, for each item j, two *dummy consumers*, who value item j at r_j, and all other items at 0. Running $\mathrm{Algo}(V')$ then gives us a Walrasian Equilibrium $(\mathbf{p}, M')$. From M', we deduce a matching by removing all dummy consumers and their edges; finally, while there is an unsold item j in the demand set of a real consumer i that is not allocated an item, we allocate item j to consumer i. The resulting matching, M, together with the prices, $\mathbf{p}$, is the output of our algorithm $\mathrm{Algo}_{\mathbf{r}}(V)$.

THEOREM 3.4. *If* Algo *is an algorithm computing Walrasian Equilibria, then* $\mathrm{Algo}_{\mathbf{r}}$ *outputs a Walrasian Equilibrium with reserve prices* $\mathbf{r}$.

Proof. Consider $(\mathbf{p}, M')$. For each item j, at least one of the two dummy customers is not allocated item j. Since that customer is not envious, we must have $p_j \ge r_j$.

Since $(\mathbf{p}, M')$ is envy-free, the output $(\mathbf{p}, M)$ must also be envy-free for V: the prices have not changed.

Since $(\mathbf{p}, M')$ is a Walrasian Equilibrium, the only unsold items in $(\mathbf{p}, M)$ are those that were allocated to dummy consumers. This can only happen if $p_j \le r_j$, hence $p_j = r_j$.

The last step of the algorithm only comes up for customers whose bundles in their demand set all have utility 0, and ensures that condition (3) is satisfied. 0pt

Our algorithm will compute Walrasian Equilibria for several appropriately chosen reserve prices, and output the one yielding maximum revenue among them. The analysis will be based on the following lemma.

LEMMA 3.1. *Let π be a maximum weighted matching on the valuation matrix V. Given a value r, let k_r be the number of edges of π with valuation at least r. If $(\mathbf{p}, M)$ is any Walrasian Equilibrium with reserve prices $\mathbf{r} = (r, r, \ldots, r)$, then the seller profit of $(\mathbf{p}, M)$ is at least $r \cdot k_r/2$.*

Proof. Consider an edge (i, j) of π with valuation at least r. From the definition of a Walrasian equilibrium with reserve prices, we obtain that if item j is not matched by M, then $p_j = r$; if in addition $j \in D_i$ then customer i must be matched by M, whereas if $j \notin D_i$ then the bundles in the demand set of i all have positive valuation and so customer i must also be matched by M. Thus, for every edge (i, j) of π with valuation at least r, we have that either i or j is matched in M.

Summing over edges of π with valuation at least r, we get that $k_r \leq 2|M|$, and so M sells items, all at price at least r, to at least $k_r/2$ consumers, yielding seller profit at least $r \cdot k_r/2$. 0pt

3.2.2 The Envy-free Approximation Algorithm. Using the algorithm $\mathrm{MaxWEQ}_{\mathbf{r}}$, obtained by modifying the algorithm for maximum Walrasian prices to include reserve prices, we can now give our approximation algorithm.

Envy-Free Pricing Approximation Algorithm
Input: Valuation matrix V.

1. Let π be a maximum weight matching of V, and $r_1 \geq r_2 \geq \ldots \geq r_\ell$ the valuations on the edges of π.
2. For each j, run the algorithm $\mathrm{MaxWEQ}_{\mathbf{r}}$ on input V, with reserve prices $\mathbf{r} = (r_j, r_j, \ldots, r_j)$. Let $(\mathbf{p}^{(j)}, M^{(j)})$ be the output.

Output: The pair $(\mathbf{p}^{(j^*)}, M^{(j^*)})$ with the maximum seller profit.

THEOREM 3.5. *The Envy-free Pricing Approximation Algorithm outputs, in polynomial time, an envy-free pricing and matching which has seller profit at least* $\mathrm{OPT}/(2\ln n)$, *where* OPT *is the optimal envy-free seller profit.*

Proof. The output is envy-free by Theorem 3.4, and the running time is clearly polynomial.

Let P denote the profit of our approximation algorithm. For all j we have

$$P \geq \text{Profit}(\mathbf{p}^{(j)}, M^{(j)}) \geq \frac{j \cdot r_j}{2}$$

by Lemma 3.1, and so $r_j \leq 2P/j$.

On the other hand, the optimal envy-free seller profit is at most the weight of the maximum weighted matching π, i.e.,

$$\text{OPT} \leq \sum_j r_j \leq \sum_j \frac{2P}{j} \leq 2P \ln n.$$

0pt

Notice that we only used the fact that the $(\mathbf{p}^{(j)}, M^{(j)})$ are Walrasian Equilibria with reserve prices, but did not require the maximality of the Walrasian Equilibria found by MaxWEQ. Hence, we could have used any algorithm $\mathrm{Algo}_{\mathbf{r}}$ for computing Walrasian Equilibria with reserve prices instead.

The analysis of the algorithm is tight: Consider an instance with n consumers and two copies of each of n items, where consumer i values all items $j \geq i$ at $1/i$, and all other items at 0. Then, pricing item j at $1/j$, and allocating it to consumer j, is envy-free and has profit $\Theta(\ln n)$, while our algorithm obtains profit 1. The algorithm's profit is 1 because no duplicate copies are sold; hence, all items must be priced at the reserve price.

3.3 Pricing Over Time. We now consider a special case of unit-demand envy-free pricing, the *pricing over time* problem, where the different items represent one object that is available at different points in time (for instance, an airline ticket or network bandwidth). Here, each consumer i values the item at a constant value of v_i over a time interval $[s_i, t_i]$ and at 0 at all other times. The envy-free condition means that customer i will buy the item at the lowest price below v_i that is available during the $[s_i, t_i]$ interval. The unlimited supply case of pricing over time may be a good model for selling digital content. Blum et al. [6] previously considered a similar pricing over time model; however, the problem they considered was online and did not require the pricing to be envy-free.

THEOREM 3.6. *The Unlimited-Supply Pricing Over Time problem can be solved in polynomial time.*

Proof. We give a dynamic programming algorithm. For points in time $s < t$, and a price p, we define $a_p(s, t)$ to be the maximum profit that can be obtained from consumers i with $s < s_i$ and $t_i < t$, when the minimum price over the time interval (s, t) is at least p. We are then interested in computing $a_0(0, \infty)$.

The trivial base case occurs when no consumer would pay at least p in the interval (s, t), and the profit is 0. Otherwise, given s and t, and using $n_{t',q}$ to denote the number of consumers i (among those with $s < s_i$ and $t_i < t$) who would be willing to buy the item at price q at time t' (i.e., $s_i \leq t' \leq t_i$ and $v_i \geq q$), we can express

$$a_p(s, t) \quad = \quad \max_{q \geq p, t' \in (s,t)} (a_q(s, t') + a_q(t', t) + q \cdot n_{t',q}).$$

Any particular choice of q and t' corresponds to selling the item at price q at time t', to all consumers willing to buy it then. To ensure envy-freeness, this imposes a constraint of pricing at least at q during the sub-intervals that these consumers would also be willing to buy at.

The optimality of the dynamic program follows simply by looking at the selling times and prices of the optimum solution in non-decreasing order of price. The crucial observation is that we only need to consider a polynomial number of prices q and times t'. For the latter, we notice that items only need to be sold at times s_i or t_i, i.e., endpoints of consumers' intervals. Any other selling time can be shifted slightly to the left or right without altering profit. Similarly, all selling prices are valuations v_i: if the item were ever sold at a price p not equal to a valuation, then all prices of p could be raised slightly without losing customers or creating envy. Hence, the dynamic programming table contains only $O(n^3)$ entries $a_p(s,t)$, each of which is computed in time $O(n^2)$. This completes the proof. 0pt

We can extend this approach to a pseudo-polynomial time algorithm for limited supply c_t at time t. Given s and t, let $n^+_{t',q}$ denote the number of customers with $s_i \leq t' \leq t_i$ and with valuation at t' strictly greater than q. Then the dynamic program relies on the following recurrence relation:

$$a_p(s,t) = \max_{q \geq p, t' \in (s,t)} \text{ of}$$

$$\begin{cases} (a_q(s,t') + a_q(t',t) + q \cdot n_{t',q}). & \text{if} \quad n_{t',q} \leq c_{t'} \\ (a_q(s,t') + a_q(t',t) + q \cdot c_{t'}). & \text{if} \quad n^+_{t',q} \leq c_{t'} \leq n_{t',q} \\ -\infty & \text{if} \quad c_{t'} < n^+_{t',q} \end{cases}$$

Whether the problem can also be solved in polynomial time in the presence of limited supply is an open question. The best-known approximation algorithm is the general logarithmic approximation from the previous section; however, we do not know the limited supply problem to be NP-hard.

4 Towards Truthful Competitive Mechanisms

As discussed in the introduction, an additional motivation for the study of envy-free pricing is the fact that the profit obtainable from such a pricing is a natural lower bound to analyze truthful mechanisms for profit maximizing combinatorial auctions. In a combinatorial auction, the setting is the same as before, except that the valuations $v_i(S)$ are known only to consumer i, but not the seller. The seller solicits, implicitly or explicitly, the valuations from the consumers in the process of running an auction. The consumers, who know the auction mechanism, will choose to misrepresent their valuations if they derive more utility from the resulting outcome. A popular approach to dealing with this kind of strategizing is to design *truthful* mechanisms, in which it is in the consumers' own best interest to disclose their true valuations.

In analyzing the performance of a mechanism, it is natural to compare it with an optimal omniscient seller via a *competitive analysis* [13, 11, 5]. If the seller is allowed to sell identical items to different consumers at different prices, then no truthful mechanism can be competitive [13]. Thus, [13, 11, 5] consider the optimal omniscient seller that uses a single price for identical items. The natural generalization of the single-price condition to the case where distinct items are for sale is that of envy-freeness.

Building on the ideas used in the approximation algorithm in Section 3.2, we present a new truthful $\log h$-competitive mechanism mechanism for unit-demand combinatorial auctions when all valuations are in the interval $[1, h]$. That is, the mechanism is guaranteed to achieve at least a $1/\log h$ fraction of the optimal envy-free pricing profit.

Our mechanism builds on the Vickrey-Clarke-Groves (VCG) mechanism with reserve prices. The VCG mechanism, much like Walrasian Equilibria, outputs the *efficient* allocation given by the maximum weighted matching, M. However, the VCG mechanism aims at achieving truthfulness, and therefore computes prices different from MaxWEQ. If V_{-i} denotes the matrix of valuations with the i^{th} row (corresponding to the consumer i) deleted, the payment of consumer i, and thus the price of the item j allocated to consumer i by M, is computed as:

$$p_j \quad = \quad v_i(j) - \omega(V) + \omega(V_{-i}),$$

where $\omega(V)$ again denotes the weight of a maximum matching of the valuation matrix V.

VCG is known to be truthful; interestingly, its prices are exactly the minimum Walrasian prices [20], and thus the VCG prices are in fact envy-free. Since VCG computes a Walrasian Equilibrium, we can use the generic technique presented in Section 3.2.1 to obtain the VCG mechanism with reserve prices, $\mathrm{VCG}_{\mathbf{r}}$; the correctness of the mechanism follows from Theorem 3.4. The construction of adding reserve prices to VCG is well known, and the resulting mechanism $\mathrm{VCG}_{\mathbf{r}}$ is truthful [23, 8, 14].

We can use this fact to obtain a truthful $\log h$-competitive auction, in a manner similar to the envy-free pricing approximation algorithm. However, to preserve truthfulness, we cannot run $\mathrm{VCG}_{\mathbf{r}}$ for different choices of reserve prices; instead, we will choose the reserve price randomly. The complete mechanism is then:

Unit Demand Combinatorial Auction Input: Valuation matrix V, where $1 \leq v_i(j) \leq h$ for all i, j.

1. Pick an integer k uniformly at random from $\{1, \ldots, \lfloor \log h \rfloor\}$, and let $r = 2^k$.
2. Compute prices $\mathbf{p}$ and an allocation M by running the algorithm $\text{VCG}_\mathbf{r}$ on input V with reserve prices $\mathbf{r} = (r, r, \ldots, r)$.

Output: the pair $(\mathbf{p}, M)$.

THEOREM 4.1. *The unit-demand combinatorial auction is truthful and $4 \log h$ competitive for any input in which all consumer valuations are in the interval $[1, h]$.*

Proof. That the auction is truthful follows directly from the truthfulness of $\text{VCG}_\mathbf{r}$.

Let $r_1 \geq r_2 \geq \ldots \geq r_m$ be the prices of the items sold in the maximum weighted matching M. Clearly, the optimum profit is bounded from above by $\sum_j r_j$. Let n_r be the number of items sold in M at price r or more (i.e., n_r is the index j such that $r_j \geq r > r_{j+1}$). If $\mathbf{r} = (r, r, \ldots, r)$, then Lemma 3.1 implies that the revenue of $\text{VCG}_\mathbf{r}$ is at least $r \cdot n_r/2$, so the expected revenue R of the auction is at least $\sum_{k=0} \lfloor \log h \rfloor \frac{2^k n_{2^k}}{2 \log h}$.

On the other hand, we can bound each price r_j as $r_j \leq 2 \sum_{k=0}^{\lfloor \log h \rfloor} 2^k \cdot [r_j \geq 2^k]$, where $[r_j \geq 2^k]$ is 1 if $r_j \geq 2^k$ and 0 otherwise. Summing over all j, we obtain that

$$\sum_j r_j \quad \leq \quad 2 \sum_{k=0}^{\lfloor \log h \rfloor} 2^k \cdot n_{2^k} \quad \leq \quad (4 \log h) \cdot R,$$

completing the proof. 0pt

Note that if h is polynomial in n, this auction has the same approximation ratio as the non-truthful approximate envy-free pricing algorithm given earlier. However, there are instances where the latter algorithm outperforms this mechanism by a factor of $\log n$, e.g., when using a single price is close to optimal.

5 Single-Minded Consumers

We now turn to the problem of envy-free pricing for single-minded consumers. Recall that the input in this case is given by the pairs (v_i, S_i): consumer i values the bundle S_i at v_i, and all other bundles at 0. We will also refer to S_i as consumer i's *request*, and v_i as his *bid*. Even under strong additional restrictions, the pricing problem for single-minded consumers is APX-hard, as shown below.

THEOREM 5.1. *Envy-free pricing for single-minded consumers is APX-hard, even when all items are available in unlimited supply, and*

1. *$v_i = 1$ and $|S_i| \leq 2$ for all i, or*
2. *$v_i \in \{1, 2\}$, and $|S_i| = 2$, for all i.*

Proof. We first show that the problem is APX-hard when $v_i = 1$ for all requests i and $|S_i| \leq 2$ for all i by a reduction from the MAXCUT problem for 3-regular graphs (which was shown to be APX-hard in [2]). Let $G = (V, E)$ be the instance of MAXCUT (a 3-regular graph), with $n = |V|$ nodes and $m = |E|$ edges.

The items consist of an element v for each node $v \in V$, as well as a *positive dummy* d_p and a *negative dummy* d_n.

All consumer valuations are equal to 1, and requests fall into three classes: *edge requests*, *node requests*, and *dummy requests*. The edge requests are simply one request for each edge $e = \{u, v\} \in E$. The node requests are 4 requests each for the sets $\{d_p, v\}$ and $\{d_n, v\}$, for each node v. The dummy requests are $5m$ requests for the set $\{d_p\}$, and $5m$ requests for the set $\{d_p, d_n\}$.

We show that there is a price vector with profit at least $10m + 4n + k$ if and only if there is a cut that cuts at least k edges.

For the easy direction, assume that there is a cut cutting at least k edges. Assign price 1 to d_p and to all nodes on one side of the cut, and price 0 to d_n and to all other nodes. We can then verify that the profit is at least $10m + 4n + k$.

For the converse direction, let $\mathbf{p}$ be a price vector with maximum profit, and assume that the profit is at least $10m + 4n + k$. By simple exchange arguments, we can first show that the optimality implies that $p_{d_p} = 1$, and $p_{d_n} = 0$. Using this, we can next use exchange arguments to show that each p_v is either 0 or 1, by first ruling out that any p_v is between 0 and $\frac{1}{2}$, and then rounding all non-zero prices to 1.

The price vector then defines a natural cut between nodes of price 0 and nodes of price 1. Because the total revenue from dummy requests is exactly $10m$, and from node requests $4n$, the total profit from edge requests must be k; and it can be seen easily that an edge contributes profit 1 if and only if it is cut.

Because the maximum cut cuts at least half of the edges, we are only interested in the case where $k \geq m/2$, and because the graph is 3-regular, we also have $n \leq m$. Then, a few straightforward calculations show that a $\frac{28+\alpha}{29}$ approximation for the envy-free pricing problem would yield an α-approximation to cubic MAXCUT, proving that the envy-free pricing problem is NP-hard to approximate within $\frac{28+\alpha}{29}$, where α is the approximation hardness constant in [2] (not given explicitly there).

The above reduction can be adapted very slightly to yield the same result when all set sizes are exactly 2, and all valuations are either 1 or 2. To avoid the single element requests $\{d_p\}$, we add one more dummy element d, and replace the requests $\{d_p\}$ with $5m$ requests for the set $\{d, d_p\}$ with valuation 2 each, and $5m$ requests for the set $\{d, d_n\}$ with valuation 1 each.

Now, a very similar argument can be used to show that in an optimal solution, the prices are $p_d = p_{d_p} = 1$, and $p_{d_n} = 0$; the remainder of the proof stays unchanged. 0pt

5.1 A Logarithmic Approximation Algorithm for Items in Unlimited Supply. The previous hardness result shows that we are unlikely to find a PTAS for the single-minded bidder case. However, it is fairly straightforward to get a logarithmic approximation for single-minded bidders in the case of unlimited supply. Our algorithm only considers pricings in which all items are priced the same. The candidate prices are $q_i = v_i/|S_i|$, for each customer i. Among the pricings assigning all items price q_i, our algorithm simply selects the one giving largest profit and outputs it.

THEOREM 5.2. *This algorithm is a $\log n + \log m$ approximation for envy-free pricing when bidders are single-minded and items are available in unlimited supply.*

Proof. We assume that the consumers are ordered such that $q_1 \geq q_2 \geq \ldots \geq q_n$. If all items are priced at q_i, then the seller profit is $R_i = \sum_{1 \leq j \leq i} |S_j| \cdot v_i/|S_i|$. Rearranging yields that $v_i = |S_i|R_i/\sum_{1 \leq j \leq i} |S_j|$. Because the algorithm chooses the price R maximizing profit, we have that $R_i \leq R$ for all i, and thus

$$\begin{aligned}
\sum_{i=1}^{n} v_i &= \sum_{i=1}^{n} \frac{|S_i|R_i}{\sum_{1 \leq j \leq i} |S_j|} \\
&\leq R \cdot \sum_{i=1}^{n} \sum_{k=1}^{|S_i|} \frac{1}{k + \sum_{1 \leq j \leq i-1} |S_j|} \\
&\leq R \cdot \ln(\sum_i |S_i|).
\end{aligned}$$

$\sum_i v_i$ is a trivial upper bound on the optimum, so the theorem follows because $\sum_i |S_i| \leq nm$. 0pt

The analysis of this algorithm is tight, as can be seen by the example in which customer i wants to buy only item i, with valuation $1/i$. In this case, the trivial upper bound is easily achievable, while our algorithm only has revenue 1. On the other hand, any analysis using only this trivial upper bound on the optimal revenue cannot prove an approximation guarantee better than $O(\log n)$, as can be seen by the simple example of all users requesting item 1, with user i's valuation being $1/i$.

5.2 The Tollbooth Problem. While the envy-free pricing problem, even for single-minded bidders, is hard to approximate, there are interesting and more tractable special cases. Here, we study the *tollbooth problem*. The items are now the edges of a graph G, which we may think of as highway segments, and customers' requests are for paths in the graph. A customer's valuation may be derived from the price in money and inconvenience of using an alternate method of transportation. The seller is the owner of the highway system, and would like to choose tolls for the segments so as to maximize profits. Notice that the APX-hardness reduction in Theorem 5.1 can be thought of as generating length-1 or length-2 paths in a star graph, so the tollbooth problem on trees is still APX-hard.

A special case that is polynomial-time solvable is the case when all path requests share one common endpoint r, which we consider as the root of the tree. This case is motivated by commuter traffic in the vicinity of a large city: most cars are either originating from or destined for the large city, and the paths used by most of the cars forms a tree. We first assume that edges have infinite capacity, which corresponds to unlimited supply.

THEOREM 5.3. *The unlimited supply tollbooth problem on rooted trees can be solved in polynomial time.*

Proof Sketch. We give a dynamic programming algorithm. For a node w, let R_w denote the set of all requests originating in the subtree T_w rooted at w. We define $a(w, b)$ to be the optimum revenue obtainable from requests in R_w if the path from w to the root costs exactly b. We are then interested in computing $a(r, 0)$.

For node w, let $w_1, \ldots, w_k$ denote its children, and $n_w(b)$ the number of requests originating at w with valuation b or higher. Then:

$$a(w, b) = b \cdot n_w(b) + \textstyle\sum_i \max_{b' \geq b} a(w_i, b').$$

The crucial observation is that we only need to consider a polynomial number of costs b. Indeed, all selling path prices are valuations: if there is a feasible request $(w, b) \in R_w$ in the optimum pricing, then w.l.o.g., the total price of the path from w to the root is b', for some $(w', b') \in R_w$. This claim is easily proved by induction, starting at the leaves of the tree.

Hence in the maximum over $b' \geq b$, we only have to consider values $b' \in R_{w_i}$, so the dynamic programming table has size $O(n^2)$, and $a(w, b)$ can be computed in polynomial time. ■

We can extend the previous approach to obtain a pseudo-polynomial time dynamic programming algorithm for the case of edge capacities c_e. We let $a(w, c, b)$ denote the maximum revenue that can be obtained if at most c requests from R_w are feasible, and the path from w to the root has total price exactly b. Here, we write $n_w^+(b)$ for the number of requests originating with node w with valuation strictly greater than b. When, writing e_i for the edge from w to its child w_i, the envy-freeness condition implies that we can compute $a(w, c, b)$ as the maximum, over $(c_0, c_1, \ldots, c_k)$ such that $0 \leq c_i \leq c_{e_i}$, $n_w^+(b) \leq c_0 \leq n_w(b)$, and

$\sum_i c_i \leq c$, of

$$(b \cdot c_0 + \sum_{i \geq 1} \max_{b' \geq b} a(w_i, c_i, b')).$$

Here, c_0 is the capacity allotted to requests originating with w, and the envy-freeness condition requires all requests with valuation strictly exceeding b to be served. While the maximum is seemingly taken over $\prod_i c_{e_i}$ k-tuples, it can in turn be computed by dynamic programming over values of c_j vs. $\sum_{j'>j} c_{j'}$.

5.3 The Highway Problem. Another "simple" case of the tollbooth problem is when the underlying graph is in fact a path, and all requests are subpaths (not necessarily sharing a common endpoint). This case is clearly motivated by tolls to be charged on a single freeway. Even though a path is about as simple a graph as we can hope for, the problem is surprisingly complex: at this point, we do not know if optimal prices can be computed in polynomial time, or whether the problem is NP-hard on a path. However, we can derive pseudo-polynomial dynamic programming algorithms when some of the parameters are bounded. Here again we assume infinite edge capacity. The algorithms rely on the following integrality lemma:

LEMMA 5.1. *If all valuations v_i are integral, then there is an optimal solution in which all prices p_e are integral.*

Proof. Let $\mathbf{p}$ be any price vector, and R the set of all requests feasible under $\mathbf{p}$. We show that there is an integral assignment $\mathbf{p}'$ such that each request in R is still feasible under $\mathbf{p}'$, and the total profit obtained from R is at least as large as under $\mathbf{p}$. Applying this to the optimal assignment $\mathbf{p}$ then clearly proves the lemma.

Each request is a subpath, so if the edges are numbered $1, \ldots, m$, with prices $p_1, \ldots, p_m$, then each request i uses some edges $l_i, \ldots, r_i$. Given the set R of requests that must be feasible, the optimal assignment that makes all of R feasible is the solution to the following linear program:

$$\begin{array}{lllll} \text{Maximize} & \sum_{i \in R} \sum_{j=l_i}^{r_i} p_j & & & \\ \text{subject to} & \sum_{j=l_i}^{r_i} p_j & \leq & v_i & \text{for each } i \in R \\ & p_j & \geq & 0 & \text{for each } j \end{array}$$

Notice that each row in the matrix for the LP is of the form $0^*1^*0^*$. This is enough to prove that the matrix is totally unimodular, i.e., that the determinant of each non-singular square submatrix is ± 1. Indeed, consider any submatrix. It still satisfies the property that each row is of the form $0^*1^*0^*$. To compute the determinant, reorder the rows by non-decreasing l_i, then by non-decreasing r_i, subtract the first row from every row which starts with a 1. These steps do not change the determinant except for perhaps its sign, and we are now left with a matrix whose first column has a 1 at the first row and 0's everywhere else, and every row is of the form $0^*1^*0^*$. Expanding by the first column, we conclude unimodularity by induction. Because the matrix is totally unimodular, we can apply a theorem of Hoffman and Kruskal [16, 22], which states that for an integral right-hand side vector $V = (v_i)$, all vertex solutions of the LP are integral. In particular, as there is an optimal solution that is a vertex, we obtain that there is an integer optimum, which completes the proof. 0pt

We can use the integrality lemma to obtain a pseudo-polynomial time dynamic programming algorithm for the following special cases.

THEOREM 5.4.

1. *If there is a constant upper bound B on all valuations v_i, and all valuations are integral, then there is a polynomial-time ($O(B^{B+2} n^{B+3})$) dynamic programming algorithm to find an optimal price vector.*
2. *If all requests have path lengths bounded by some constant k, and all valuations are integral, then there is a pseudo-polynomial dynamic programming algorithm with running time $O(B^{k+1} \cdot n)$ for computing an optimal price vector.*

Proof Sketch. We sketch dynamic programs with an analysis for both restrictions. Details of the analysis will be given in the full version.

1. Our dynamic programming algorithm maintains a table with entries $a_{j,(k_1,\gamma_1),\ldots,(k_{B+1},\gamma_{B+1})}$, which denotes the maximum profit that can be obtained from requests i with right endpoint $r_i \leq j$, given that the rightmost $B+1$ edges with non-zero price to the left of j are $k_1 < k_2 < \ldots < k_{B+1} \leq j$ with associated prices $\gamma_1, \ldots, \gamma_{B+1} > 0$. A crucial observation here is that the above integrality lemma (Lemma 5.1) guarantees that we need only consider $\gamma_b \in \{1, \ldots, B\}$, and thus also $\sum_{b=1}^{B+1} \gamma_b > B$. This in turn implies that no request i with $l_i < k_1$ and $r_i \geq k_{B+1}$ can be feasible, so we can safely ignore these requests.

The initialization for the table is fairly clear. For the update step, consider the pricing of the edge $j+1$, starting from a table entry $a = a_{j,(k_1,\gamma_1),\ldots,(k_{B+1},\gamma_{B+1})}$. If edge $j+1$ is priced at 0, then we add to a the profit obtained from paths ending at $j+1$ to obtain a candidate for $a_{j+1,(k_1,\gamma_1),\ldots,(k_{B+1},\gamma_{B+1})}$. Otherwise, when edge $j+1$ is given a positive price γ, we obtain a candidate for table entry $a_{j+1,(k_2,\gamma_2),\ldots,(k_{B+1},\gamma_{B+1}),(j+1,\gamma)}$, again by adding to a the profit obtained from paths ending at $j+1$.

It is not difficult to see that we can conversely reconstruct all $O(nB)$ candidate table entries that could result in candidates for a particular desired entry $a_{j+1,(k_1,\gamma_1),\ldots,(k_{B+1},\gamma_{B+1})}$, and then choose the maximum among all those candidates. Hence, the total running time (with $O(B^{B+1} n^{B+2})$ table entries) is $O(B^{B+2} n^{B+3})$.

2. The algorithm maintains a table $a_{j,\gamma_1,\ldots,\gamma_k}$, where an entry is the maximum amount of profit that can be obtained from requests ending at position j or below, given that the edges $j, j-1, \ldots, j-k+1$ are priced at $\gamma_1, \ldots, \gamma_k$, respectively.

To calculate an entry $a_{j+1,\gamma_1,\ldots,\gamma_k}$, consider all possible values of $\gamma' \in \{0, \ldots, B\}$, and add to $a_{j+1,\gamma_2,\ldots,\gamma_k,\gamma'}$ the maximum profit for all feasible requests ending exactly at $j+1$ with cost assignments $\gamma_2, \ldots, \gamma_k, \gamma'$ to edges $j, j-1, \ldots, j+2-k, j+1-k$. Because no path has length exceeding k, the values of $(\gamma_i)_i$ and γ' are sufficient to determine whether a request is feasible, and we can restrict the choices of γ' to $\{0, \ldots, B\}$ because of the Integrality Lemma 5.1.

The computation of a new table entry takes time $O(B)$, and because the table size is $O(B^k n)$, the running time is $O(B^{k+1} n)$. ∎

References

[1] G. Aggarwal, T. Feder, R. Motwani, and A. Zhu. Algorithms for multi-product pricing. In *Proc. of ICALP*, 2004.

[2] P. Alimonti and V. Kann. Hardness of approximating problems on cubic graphs. In *Proc. 3rd Ital. Conf. on Algorithms and Complexity*, pages 288–298. Springer, 1997.

[3] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proc. 14th ACM Symp. on Discrete Algorithms*. ACM/SIAM, 2003.

[4] A. Archer and E. Tardos. Frugal path mechanisms. In *Proc. 13th ACM Symp. on Discrete Algorithms*, pages 991–999. ACM/SIAM, 2002.

[5] A. Blum, V. Kumar, A. Rudra, and F. Wu. Online learning in online auctions. In *Proc. 14th ACM Symp. on Discrete Algorithms*. ACM/SIAM, 2003.

[6] A. Blum, T. Sandholm, and M. Zinkevich. Online algorithms for market clearing. In *Proc. 13th ACM Symp. on Discrete Algorithms*, pages 971–980. ACM/SIAM, 2002.

[7] J. Bulow and J. Roberts. The simple economics of optimal auctions. *The Journal of Political Economy*, 97:1060–90, 1989.

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

[9] X. Deng, C. Papadimitriou, and M. Safra. On the complexity of equilibria. In *Proc. 34th ACM Symp. on Theory of Computing*. ACM Press, 2002.

[10] N. Devanur, C. Papadimitriou, A. Saberi, and V. Vazirani. Market equilibria via primal-dual-type algorithm. In *Proc. 43rd IEEE Symp. on Foundations of Computer Science*, 2002.

[11] A. Fiat, A. Goldberg, J. Hartline, and A. Karlin. Competitive generalized auctions. In *Proc. 34th ACM Symp. on Theory of Computing*. ACM Press, 2002.

[12] A. Goldberg and J. Hartline. Competitive auctions for multiple digital goods. In *Proc. 9th European Symposium on Algorithms*, pages 416–427. Springer, 2001.

[13] A. V. Goldberg, J. D. Hartline, and A. Wright. Competitive auctions and digital goods. In *Proc. 12th ACM Symp. on Discrete Algorithms*, pages 735–744. ACM/SIAM, 2001.

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

[15] F. Gul and E. Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87:95–124, 1999.

[16] A. Hoffman and J. Kruskal. Integral boundary points of convex polyhedra. In H. Kuhn and A. Tucker, editors, *Linear Inequalities and Related Systems*, pages 223–246. Princeton University Press, 1956.

[17] T. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

[18] R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Proc. 44th IEEE Symp. on Foundations of Computer Science*, 2003.

[19] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. In *Proc. 1st ACM Conf. on Electronic Commerce*, pages 96–102. ACM Press, 1999.

[20] H. Leonard. Elicitation of honest preferences for the assignment of individuals to positions. *Journal of Political Economy*, 91:1–36, 1983.

[21] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.

[22] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Dover, 1982.

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

[24] L. Walras. *Elements of Pure Economics*. Allen and Unwin, 1954.