Approximation Algorithms and Online Mechanisms for Item Pricing

Maria-Florina Balcan* **Avrim Blum***

March 2006
CMU-CS-05-176R

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

* School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. {ninamf,avrim}@cs.cmu.edu

Research supported in part by NSF grants CCF-0514922, CCR-0122581, and IIS-0121678
This is an expanded version of CMU-CS-05-176.

Keywords: Approximation Algorithms, Online Optimization, Profit Maximization, Combinatorial Auctions, Single Minded, Unlimited Supply

Abstract

We present approximation and online algorithms for a number of problems of pricing items for sale so as to maximize seller's revenue in an unlimited supply setting. Our first result is an $O(k)$-approximation algorithm for pricing items to single-minded bidders who each want at most k items. This improves over recent independent work of Briest and Krysta [6] who achieve an $O(k^2)$ bound. For the case $k = 2$, where we obtain a 4-approximation, this can be viewed as the following *graph vertex pricing* problem: given a (multi) graph G with valuations w_e on the edges, find prices $p_i \geq 0$ for the vertices to maximize

$$\sum_{\{e=(i,j):w_e \geq p_i+p_j\}} (p_i + p_j) \, .$$

We also improve the approximation of Guruswami et al. [13] from $O(\log m + \log n)$ to $O(\log n)$, where m is the number of bidders and n is the number of items, for the "highway problem" in which all desired subsets are intervals on a line. Our approximation algorithms can be fed into the generic reduction of Balcan et al. [3] to yield an incentive-compatible auction with nearly the same performance guarantees so long as the number of bidders is sufficiently large. In addition, we show how our algorithms can be combined with results of Blum and Hartline [4], Blum et al. [5], and Kalai and Vempala [16] to achieve good performance in the online setting, where customers arrive one at a time and each must be presented a set of item prices based only on knowledge of the customers seen so far.

1 Introduction

Consider the problem of a retailer trying to price its products to make the most profit. If customers had valuations over individual items only, then the problem of setting prices would be relatively easy: for each product i, the optimal price is such that the profit margin p_i per item sold, times the number of customers willing to buy at that price, is maximized. So, each item can be considered separately, and assuming the company knows its market well, the *computational* problem of setting prices is fairly trivial.

However, suppose that customers have valuations over *pairs* of items (e.g., a computer and a monitor, or a tank of gas and a cup of coffee), and will only purchase if the combined price of the items in their pair is below their value. In this case, we can model the problem as a (multi) *graph*, where each edge e has some valuation w_e, and our goal is to set prices $p_i \geq 0$ on the vertices of the graph to maximize total *profit*: that is,

$$\text{Profit}(\mathbf{p}) \quad = \sum_{\{e=(i,j): w_e \geq p_i + p_j\}} (p_i + p_j)\,.$$

where $\mathbf{p}$ is the vector of individual prices.[1]

We call this the *graph vertex pricing* problem. More generally, if customers have valuations over larger subsets, we can model our computational problem as one of pricing vertices in a *hypergraph*, or in more standard terminology, the problem of pricing items in an unlimited-supply combinatorial auction with single-minded bidders. Guruswami et al. [13] show an $O(\log m + \log n)$-approximation for the general problem, where n is the number of items (vertices) and m is the number of customers (hyperedges). They also show that even the *graph* vertex pricing problem is APX-hard — and this is true even when all valuations are identical (if self-loops are allowed) or all valuations are either 1 or 2 (if self-loops are not allowed). In related work, Hartline and Koltun [15] give a $(1 + \epsilon)$-approximation that runs in time exponential in the number of vertices, but that is near-linear time when the total number of vertices in the hypergraph is constant. Recently, Demaine et al. [8] have shown that it is hard to approximate the hypergraph vertex pricing problem within a factor of $\log^\delta n$, for some $\delta > 0$, assuming that NP $\not\subseteq$ BPTIME$\big(2^{n^\epsilon}\big)$ for some $\epsilon > 0$.

In this paper, we give a 4-approximation for the graph vertex pricing problem, and more generally we present an $O(k)$-approximation for the case of hypergraphs in which each edge has size at most k (i.e., all customers' valuations are over subsets of size at most k). The latter result improves over the recent independent work of Briest and Krysta [6] who give a bound of $O(k^2)$.

We also consider the highway problem studied in [13]. This problem is the special case of the hypergraph pricing problem where vertices are numbered $1, \ldots, n$ and each customer wants an interval $[i, j]$.[2] For this problem, we give an $O(\log n)$-approximation, improving slightly over the $O(\log m + \log n)$ approximation of [13], and also give an $O(1)$-approximation for the case that all users want the *same* number of items up to a constant factor. Finally, we also give a fully polynomial time approximation scheme (FPTAS) for the case that the desired subsets of different customers form a hierarchy (this is defined more precisely in Section 6).

[1] This formula corresponds to a model in which items have zero marginal cost to the retailer (digital goods) so that an item sold at price p_i generates profit p_i. Alternatively, if products have a fixed marginal cost, and we cannot sell them below cost (say, due to the presence of resellers), then we can think of p_i as the profit margin on item i and simply subtract our costs for the endpoints from each valuation w_e.

[2] Previous work [15, 13] uses "m" to denote the number of items and "n" to denote the number of customers, viewing the *items* as edges in some network. Since we are viewing items as vertices and customers as (hyper)edges, we have reversed this notation.

Incentive-compatibility Our results described above assume the seller "understands the market": that is, we know how many customers will buy different sets of items and at what prices. Thus, we are simply left with a computational problem. If we do not understand the market and are in the setting of an unlimited-supply combinatorial auction, we would instead want an algorithm that is *incentive-compatible*, meaning that it is in bidders' self-interest to reveal their true valuations. Fortunately, a generic reduction of [3] shows that if there are sufficiently many bidders, then for problems of this type one can convert any approximation to the computational problem into a nearly-as-good approximation to the incentive-compatible auction problem. In particular, $\tilde{O}\left(\frac{hn}{\epsilon^2}\right)$ bidders are sufficient for this reduction to produce only a factor $(1+\epsilon)$ loss in approximation ratio when all valuations lie in the range $[1, h]$. Essentially, the idea of the reduction is to randomly partitions bidders into two sets S_1 and S_2, run the approximation algorithm separately on each set, and then use the prices found for S_1 on S_2 and vice-versa (making the process incentive-compatible); the results in [3] then show that $\tilde{O}\left(\frac{hn}{\epsilon^2}\right)$ bidders are sufficient to ensure that the resulting profit is nearly as large as if one had used prices determined on each S_i on that set itself. Related results of [12, 11] give bounds of this form for the case of a single digital good. Thus, if one has sufficiently many bidders, one can focus attention on the computational approximation problem.

The above results assume a one-shot mechanism (sealed-bid auction) in which all bidders are present at the same time. We also consider the more demanding case that bidders arrive online, and one must present to each bidder a set of item prices that depend only on bidders seen in the past. We show how methods of [4, 5] for the online digital-good auction can be applied to our algorithms for graph (or k-hypergraph) vertex pricing to achieve good performance for these problems in the online setting as well. For the highway problem, we need a somewhat more involved argument using an algorithm of Kalai and Vempala [16].

Other Related Work It is worth noting that if the goal is economic efficiency (obtaining an outcome which maximizes the sum of the utilities[3] of all participants), and not revenue maximization, then for the case of limited supply single minded bidders in combinatorial auctions, both the pricing problem and the mechanism design problem are known to have good approximate solutions [17, 1].

2 Notation and Definitions

We assume we have m customers (or "bidders") and n items (or "products"). We are in an *unlimited supply* setting, which means that the seller is able to sell any number of units of each item, and they each have zero marginal cost to the seller (or if they have some fixed marginal cost, we have subtracted that from all valuations and the seller may not sell any item below cost). We consider *single-minded bidders*, which means that each customer is interested in only a single bundle of items and has valuation 0 on all other bundles. Therefore, valuations can be summarized by a set of pairs (e, w_e) indicating that a customer is interested in bundle (hyperedge) e and values it at w_e. Given the hyperedges e and valuations w_e, we wish to compute a pricing of the items that maximizes the seller's profit. We assume that if the total price of the items in e is at most w_e, then the customer (e, w_e) will purchase all of the items in e, and otherwise the customer will purchase nothing. That is, we want the price vector $\mathbf{p}$ that maximizes

$$\text{Profit}(\mathbf{p}) = \sum_{\left\{e : w_e \geq \sum_{i \in e} p_i\right\}} \sum_{i \in e} p_i.$$

[3] If a consumer buys a subset of the items, then his utility for that subset is the difference between his valuation for the subset and the purchase price.

Let $\mathbf{p}^*$ be the price vector with the maximum profit and let $\mathrm{OPT} = \mathrm{Profit}(\mathbf{p}^*)$.

Let us denote by E the set of customers, and V the set of items, and let h be $\max_{e \in E} w_e$. Let $G = (V, E)$ be the induced hypergraph, whose vertices represent the set of items, and whose hyperedges represent the customers. Notice that G might contain self-loops (since a customer might be interested in only a single item) and multi-edges (several customers might want the same subset of items). In the special case that all customers want at most two items, so G is a graph, we call this the *graph vertex pricing* problem. As mentioned in Section 1, this pricing problem was shown to be APX-hard in [13]. If all customers want at most k items, we call this the *k-hypergraph vertex pricing* problem. In [13] a simple $O(\log m + \log n)$ polynomial time approximation algorithm is given for the general problem.

3 Graph Vertex Pricing

We begin by considering the Graph Vertex Pricing problem, and show a factor 4 approximation.

Theorem 1 *There is a 4-approximation for the Graph Vertex Pricing problem.*

Proof: First notice that if G is *bipartite* (with self-loops allowed as well), then there is a simple 2-approximation algorithm. Specifically, consider the optimal price-vector $\mathbf{p}^*$ and let OPT_L be the amount of money it makes from nodes on the left, and OPT_R be the amount it makes from nodes on the right (so $\mathrm{OPT} = \mathrm{OPT}_L + \mathrm{OPT}_R$). Notice that if one takes $\mathbf{p}^*$ and zeroes out all prices for nodes on the right, then this has profit at least OPT_L since all previous buyers still buy (and some new ones may too). Therefore, we can algorithmically make profit at least OPT_L by setting all prices on the right to 0, and then separately fixing prices for each node on the left so as to make the most money possible on each node. This makes the optimal profit subject to all nodes on the right having price 0 because no edges have two distinct endpoints on the left and so the profit made from some node i on the left does not affect the optimal price for some other node j on the left. Similarly we can make at least OPT_R by setting prices on the left to 0 and optimizing prices of nodes on the right. So, taking the best of both options, we make

$$\max\left(\mathrm{OPT}_L, \mathrm{OPT}_R\right) \geq \frac{\mathrm{OPT}}{2}.$$

Now we consider the general (non-bipartite) case. Define opt_e to be the amount of profit that OPT makes from edge e. We will think of opt_e as the *weight* of edge e, though it is unknown to our algorithm. Let E_2 be the subset of edges that go between two distinct vertices, and let E_1 be the set of self-loops. Let OPT_1 be the profit made by $\mathbf{p}^*$ on edges in E_1 and let OPT_2 be the profit made by $\mathbf{p}^*$ on edges in E_2, so $\sum_{e \in E_i} opt_e = \mathrm{OPT}_i$ for $i = 1, 2$ and $\mathrm{OPT}_1 + \mathrm{OPT}_2 = \mathrm{OPT}$. Now, *randomly* partition the vertices into two sets L and R. Since each edge $e \in E_2$ has a $\frac{1}{2}$ chance of having its endpoints on different sides, in expectation $\frac{\mathrm{OPT}_2}{2}$ weight is on edges with one endpoint in L and one endpoint in R. Thus, if we simply ignore edges in E_2 whose endpoints are on the same side and run the algorithm for the bipartite case, the profit we make in expectation is at least

$$\frac{1}{2}\left[\mathrm{OPT}_1 + \frac{\mathrm{OPT}_2}{2}\right] \geq \frac{\mathrm{OPT}}{4}.$$

This proves the desired result. ■

3.1 Derandomization

If desired, the above algorithm can be derandomized by using the fact that our analysis only needs the partitioning distribution to be pairwise-independent. In particular, pairwise-independent distributions can be realized using small (polynomial-size) sample spaces [18, 20]. Thus, given a problem instance, one can simply try each possibility in the sample space and then choose the one that produces the highest profit.

3.2 Special Cases

If the underlying graph G is in fact a tree, then the there is a simple exact dynamic programming algorithm for solving the graph vertex pricing problem, and the same holds if the graph has bounded treewidth. Combining these with the recent results of Demaine et al. [9] we obtain a PTAS for the graph vertex problem in the case when G is planar [14].

4 k-Hypergraph Vertex Pricing

We now show how to extend the algorithm in Theorem 1 to get an $O(k)$-approximation when each customer wants at most k items. This improves over the $O(k^2)$ bound of [6].

Theorem 2 *There is an $O(k)$-approximation algorithm for the k-Hypergraph Vertex Pricing problem.*

Proof: We can use the following procedure.

Step 1 Randomly partition V into V_L and V_{rest} by placing each node into V_L with probability $\frac{1}{k}$.

Step 2 Let E' be the set of edges with *exactly* one endpoint in V_L. Ignore all edges in $E - E'$.

Step 3 Set prices in V_{rest} to 0 and set prices in V_L optimally with respect to edges in E'.

To analyze this algorithm, let $\mathrm{OPT}_{i,e}$ denote the profit made by $\mathbf{p}^*$ selling item i to bidder e. (So $\mathrm{OPT}_{i,e} \in \{0, p_i^*\}$ and $\mathrm{OPT} = \sum_{i \in V, e \in E} \mathrm{OPT}_{i,e}$.) Notice that the total profit made in Step 3 is *at least* $\sum_{i \in V_L, e \in E'} \mathrm{OPT}_{i,e}$ because setting prices in V_{rest} to 0 can only increase the number of sales made by $\mathbf{p}^*$ to bidders in E'. Thus, we simply need to analyze the quantity $\mathbf{E}\left[\sum_{i \in V_L, e \in E'} \mathrm{OPT}_{i,e}\right]$.

Define indicator random variable $X_{i,e} = 1$ if $i \in V_L$ and $e \in E'$, and $X_{i,e} = 0$ otherwise. We have:

$$\mathbf{E}[X_{i,e}] = \mathbf{Pr}[i \in V_L \text{ and } e \in E'] \geq \frac{1}{k}\left(1 - \frac{1}{k}\right)^{k-1} \tag{1}$$

Therefore,

$$\begin{aligned}
\mathbf{E}\left[\sum_{i\in V_L, e\in E'} \mathrm{OPT}_{i,e}\right] &= \mathbf{E}\left[\sum_{i\in V, e\in E} X_{i,e}\mathrm{OPT}_{i,e}\right] \\
&= \sum_{i\in V, e\in E} \mathbf{E}\left[X_{i,e}\right]\mathrm{OPT}_{i,e} \\
&\geq \frac{1}{k}\left(1-\frac{1}{k}\right)^{k-1}\mathrm{OPT} \\
&= O\left(\frac{\mathrm{OPT}}{k}\right).
\end{aligned}$$

■

4.1 Derandomization

As with the algorithm of Theorem 1, the above algorithm for k-hypergraph vertex pricing can also be derandomized if desired, but in this case the more sophisticated tools of Even et al. [10] are needed. First, note that we are only interested in the case that k is $o(\log n + \log m)$, since for larger values of k we can switch to the generic algorithm of Guruswami et al. [13]. Thus, we can allow for a blowup of $2^{O(k)}$ in our running time. Now, consider the algorithm in Theorem 2 and define indicator random variables $X_i = 1$ if $i \in V_L$ and $X_i = 0$ otherwise. So, each $X_i = 1$ with probability $\frac{1}{k}$, and notice that we need only k-wise independence among the X_i to calculate $\mathbf{E}[X_{i,e}]$ in Equation (1). Even al. [10] give a construction of small sample spaces that is especially well-suited to our needs. Their construction runs in time polynomial in 2^k, n, and $\frac{1}{\epsilon}$, and produces an explicit sample space with the following property: for any k-tuple $(X_{i_1}, \ldots, X_{i_k})$ of the random variables X_i and any assignment $(v_1, \ldots, v_k)$ to their values, the fraction of points in their sample space under which these variables all take on those values is within $\pm\epsilon$ of the probability of this event under our product distribution. In particular, the k-tuples we care about are those corresponding to edges $e \in E$, with values of the form $(1, 0, \ldots, 0)$ corresponding to the event that $X_{i,e} = 1$. Setting $\epsilon = o\left(\frac{1}{k}\right)$, we get that under the uniform distribution over their sample space, Equation (1) holds up to $1 - o(1)$, which suffices for our bounds. Thus, we simply run the construction of Even al. [10] using such a value of ϵ, and try each partitioning in their explicit sample space, choosing the one that produces the highest profit.

5 The Highway Problem

A particular interesting case considered in [13] is the *highway* problem. In this problem we think of the items as segments of a highway, and each desired subset e is required to be an interval $[i, j]$ of the highway. A special case of this problem shown in [13] to be solvable in polynomial time is the case when all path requests share one common end-point r. For this case, Guruswami et al. [13] give an $O(m^2)$ exact dynamic programming algorithm, which we will call $\mathcal{A}$. They also give pseudo-polynomial dynamic programming algorithms for two particular cases: an $O(h^{h+2}m^{h+3})$-time exact dynamic programming algorithm for the case when all valuations are integral, and an $O(h^{k+1}m)$ time exact dynamic programming algorithm for the case that furthermore all requests have path lengths bounded by some constant k. The highway problem was recently shown to be weakly NP-hard by Briest and Krysta [6].

We now present an $O(\log n)$ approximation algorithm for the highway problem, improving somewhat over the previous bound of $O(\log n + \log m)$ [13].

Theorem 3 *There is an $O(\log n)$-approximation algorithm for the highway problem.*

Proof: We begin by partitioning the customers into $\log_2 n$ groups. Specifically, let S_1 be the set of all customers who want item $\frac{n}{2}$. Let S_2 be the set of all customers not in S_1 who want either item $\frac{n}{4}$ or item $\frac{3n}{4}$. More generally, let S_i be the set of customers not in $S_1 \cup \cdots \cup S_{i-1}$ who want some item in $\left\{\frac{n}{2^i}, \frac{2n}{2^i}, \ldots, \frac{(2^i-1)n}{2^i}\right\}$. Now, for each set S_i we can use algorithm $\mathcal{A}$ from [13] to get a 2-approximation to the optimal profit over S_i. Specifically, for each $j \in \{1, \ldots, 2^i - 1\}$ let S_{ij} be the subset of customers in S_i who want item $\frac{jn}{2^i}$. Notice that by design, customers in set S_{ij} do not have any desired item in common with customers in $S_{ij'}$ for $j' \neq j$, which means we can consider each of them separately. Now, for each S_{ij} we get a 2-approximation to $\mathrm{OPT}(S_{ij})$ by running $\mathcal{A}$ twice, first zeroing out all prices for items to the left of item $\frac{jn}{2^i}$ and then again zeroing out all prices for items to the right of $\frac{jn}{2^i}$ and taking the best of the two cases. Since there are only $\log_2 n$ groups S_i, we simply use the algorithm $\mathcal{A}$ from [13] to get a 2-approximation to the optimal profit over S_i, and then take the best of all options, thus obtaining a $2\log_2 n$ approximation overall. ■

5.1 Special cases

Using algorithm $\mathcal{A}$ we can also get a constant-factor approximation in the special case that everyone wants exactly k items, for any (not necessarily constant) k. To see this, split the items into groups $G_1, G_2, \ldots, G_{\frac{n}{k}}$ of size k, and let OPT_{even} and OPT_{odd} be the amount of money that OPT makes from the even-numbered groups and from the odd-numbered groups respectively. We can make at least $\frac{\mathrm{OPT}_{even}}{2}$ as follows. We first set all the prices on items in the odd groups to zero. Now notice that each customer wants items in at most one even-numbered group: let us associate that customer with that group. We can now partition the customers in each even group into two types: those that want the leftmost item in the group and those that want the rightmost item in the group; we then run the dynamic program separately over each type, and take the best outcome. In a similar way, we can make at least $\frac{\mathrm{OPT}_{odd}}{2}$ by setting prices items in the even groups to 0. So we try both and take the best, thus obtaining a factor of 4 algorithm.

Similarly we can get a factor of $4c$ approximation algorithm if everyone wants between $\frac{k}{c}$ and k elements, for any value of k.

6 When bidders form a hierarchy

We present here a fully polynomial time approximation scheme for the case that the desired subsets of different (single-minded) customers form a hierarchy.[4] Specifically, we consider the case of a hypergraph where for any two edges e, e', we have either $e \subseteq e'$ or $e \supseteq e'$ or $e \cap e' = \emptyset$. This means that the edges themselves can be viewed as forming a tree structure ordered by containment. Let T_e be the set of all bidders whose desired subset is contained in e. Note that we can assume for simplicity that we have a binary hierarchy (if the hierarchy is not binary, then we can transform it into a binary hierarchy by adding fake edges e, increasing the size of the hypergraph by at most a constant factor).

We start by presenting a pseudopolynomial algorithm for the case that the bidders have integral valuations (between 0 and h). In this case, by the integrality lemma in [13] there exists an integral optimal

[4]Independently, Briest and Krysta [6] show a similar result.

solution. For each $e \in E$ and nonnegative integer $s \leq h$, let us denote by n_s^e the number of bidders with desired set e whose valuations are at least s. Now, for each $e \in E$ and nonnegative integer $s \leq nh$, let $A[s, e]$ represent the maximum possible profit we get from bidders in T_e when the total sum of the prices on items in e is exactly s. Our dynamic programming algorithm for computing the quantities $A[s, e]$ can be now specified as follows.

Step 1 For each "leaf" e in the hierarchy (an edge e that does not contain any other edges e') initialize $A[s, e] = s \cdot n_s^e$.

Step 2 Consider any edge e with children e_1 and e_2 whose A-values have been computed. Compute $A[s, e] = \max\limits_{s_1+s_2=s} (A[s_1, e_1] + A[s_2, e_2]) + sn_s^e$.

Step 3 Return $\max\limits_{s \in Val} A[s, r]$, where r is the root V.

After computing the A-values, we can then easily determine the optimal pricing vector by backtracking. Clearly, the overall procedure above runs in time polynomial in n, m and h.

If we do not want to have a polynomial dependence on h, we can instead use the above pseudopolynomial algorithm to obtain an FPTAS in a fairly standard way as follows.

Step 1 Given $\epsilon > 0$, let $l = \frac{\epsilon h}{nm}$.

Step 2 Define $w'_e = \left\lfloor \frac{w_e}{l} \right\rfloor$, for each hyperedge $e \in E$.

Step 3 Run the dynamic programming algorithm on the instance specified by $G = (V, E)$ and valuations w'_e, and let $\mathbf{p}'$ be the returned price vector.

Step 4 Output the price vector $\tilde{\mathbf{p}}$ defined as $\tilde{p}_i = l \cdot p'_i$, for $i \in V$.

Theorem 4 *The above algorithm is an FPTAS, achieving profit at least* $(1 - \epsilon)\mathrm{OPT}$ *in time polynomial in* n, m, *and* $\frac{1}{\epsilon}$.

Proof: In the following discussion, let $\mathrm{Profit}_{w'}(\mathbf{p})$ denote the profit made by using the price vector $\mathbf{p}$ in the rounded instance specified by $G = (V, E)$ and valuations w'_e. In order to prove that the profit we obtain by using $\tilde{\mathbf{p}}$ in the original instance (given by $G = (V, E)$ and valuations w_e) is at least $(1-\epsilon)\mathrm{OPT}$, we first make some observations.

Let $\mathbf{p}$ be a pricing vector and let W be the set of winners under the pricing scheme $\mathbf{p}$ in the original instance. If $\mathbf{p}''$ is the pricing vector defined as $p''_i = \left\lfloor \frac{p_i}{l} \right\rfloor$ for $i \in V$, then $\mathrm{Profit}_{w'}(\mathbf{p}'') \geq \frac{1}{l} \cdot \mathrm{Profit}(\mathbf{p}) - nm$. To see why this is true, notice first that $W \subseteq W''$, where W'' is the set of winners under the pricing scheme $\mathbf{p}''$ in the rounded instance (specified by $G = (V, E)$ and valuations w'_e). This follows from the fact that $\sum\limits_{i \in e} p_i \leq w_e$ implies $\sum\limits_{i \in e} p''_i = \sum\limits_{i \in e} \left\lfloor \frac{p_i}{l} \right\rfloor \leq \left\lfloor \frac{w_e}{l} \right\rfloor = w'_e$. This implies $\mathrm{Profit}_{w'}(\mathbf{p}'') = \sum\limits_{e \in W''} \sum\limits_{i \in e} p''_i \geq \sum\limits_{e \in W} \sum\limits_{i \in e} \left(\frac{p_i}{l} - 1\right) = \frac{1}{l} \cdot \mathrm{Profit}(\mathbf{p}) - nm$, as desired.

Let $\mathbf{p}'$ be a pricing vector and let W' be the set of winners under the pricing scheme $\mathbf{p}'$ in the rounded instance. If $\tilde{\mathbf{p}}$ is the pricing vector defined as $\tilde{p}_i = l \cdot p'_i$ for $i \in V$, then $\mathrm{Profit}(\tilde{\mathbf{p}}) \geq l \cdot \mathrm{Profit}_{w'}(\mathbf{p}')$. To see why this is true, notice first that $W' \subseteq W$, where W is the set of winners under $\tilde{\mathbf{p}}$ in the original instance. This follows from the fact that $\sum\limits_{i \in e} p'_i \leq w'_e$ implies $\sum\limits_{i \in e} \tilde{p}_i = l \cdot \sum\limits_{i \in e} p'_i \leq w'_e = l \left\lfloor \frac{w_e}{l} \right\rfloor = w_e$. This implies $\mathrm{Profit}(\tilde{\mathbf{p}}) = \sum\limits_{e \in W} \sum\limits_{i \in e} \tilde{p}_i = \sum\limits_{e \in W} \sum\limits_{i \in e} l \cdot p'_i \geq \sum\limits_{e \in W'} \sum\limits_{i \in e} l \cdot p'_i = l \cdot \mathrm{Profit}_{w'}(\mathbf{p}')$, as desired.

We are now ready to show that $\text{Profit}(\tilde{\mathbf{p}}) \geq (1-\epsilon)\text{OPT}$. Let $\mathbf{p}^*$ and $\mathbf{p}'$ be the price vectors with the maximum profit in the original and rounded instances respectively, and let W^* and W be the corresponding set of winners. Let $\tilde{\mathbf{p}}$ be the price vector defined as $\tilde{p}_i = l \cdot p'_i$ for $i \in V$ and let $\mathbf{p}''$ is the pricing vector defined as $p''_i = \left\lfloor \frac{p^*_i}{l} \right\rfloor$ for $i \in V$. According to the previous observations we have $\text{Profit}(\tilde{\mathbf{p}}) \geq l \cdot \text{Profit}_{w'}(\mathbf{p}')$. Since $\mathbf{p}'$ is the price vector with the maximum profit in the rounded instance we have $\text{Profit}_{w'}(\mathbf{p}') \geq l \cdot \text{Profit}_{w'}(\mathbf{p}'')$. Combining these together with the fact that $\text{Profit}_{w'}(\mathbf{p}'') \geq \frac{1}{l} \cdot \text{Profit}(\mathbf{p}^*) - nm$, we get $\text{Profit}(\tilde{\mathbf{p}}) \geq \text{Profit}(\mathbf{p}^*) - lnm$, which implies $\text{Profit}(\tilde{\mathbf{p}}) \geq (1-\epsilon)\text{OPT}$, as desired.

Since $w'_e \leq \frac{nm}{\epsilon}$ for all $e \in E$, we also have that our procedure runs in polynomial time in n, m, and $\frac{1}{\epsilon}$, thus being a FPTAS for the hierarchy case. ■

7 Online Pricing

As mentioned in Section 1, results of Balcan et al. [3] can be used to convert our algorithms into incentive-compatible mechanisms in the offline "batch" setting (i.e., a sealed-bid auction). In this section we consider a natural, more demanding *online* setting in which customers arrive one at a time, and we must set prices to the items for customer t based only on information about customers $1, \ldots, t-1$.

7.1 The model

We assume customers arrive one at a time. Each customer will be shown a set of item prices, and will then decide whether to purchase or not at those prices. We assume customers cannot return and cannot control their time of arrival, so any take-it-or-leave-it set of prices for customer t based only on information received from customers $1, \ldots, t-1$ is incentive-compatible. In addition, we assume an *oblivious adversary* model: that is, our objective is to achieve good expected performance for any sequence of customers, but this sequence cannot depend on the outcome of any probabilistic choices made by our algorithm.

We consider two information models. In the *full information* model, we assume that after the t-th customer departs, we learn his desired set e_t and valuation v_t. In the more difficult *posted-price* model, we assume we only find out whether and what the customer purchased but not his actual valuations. That is, if he purchases a subset at the current prices we do not know if he still would have purchased at higher prices, and if he does not purchase at the current prices, we do not know if (or what) he would have purchased at lower prices. In both models, we will be interested in algorithms that perform well compared to the best fixed setting of prices for the entire sequence. Thus, we are comparing to the same notion of OPT as in the offline case.

7.2 The Online Graph and k-Hypergraph Pricing Problems

Our 4-approximation for graph vertex-pricing, and our $O(k)$-approximation for k-hypergraph vertex pricing, can be directly adapted to the online setting by using the results of [4, 5] for the online digital-good auction.

Specifically, note that our algorithms begin by selecting a subset V_L of items to have non-zero prices, and then achieve their approximation guarantees considering only profit made from customers who want exactly one item in V_L. Thus, we can view these algorithms as effectively performing $|V_L|$ separate digital-good auctions, ignoring customers who want zero, or more than one, item from V_L. In particular, to apply these algorithms to the full-information online setting, we begin by randomly choosing the set V_L as described in the algorithms, setting prices for items in $V - V_L$ to 0. We then instantiate a separate copy of the online digital-good auction from [4] for each item $i \in V_L$. When a customer arrives, if the customer wants exactly one item i from V_L then his valuation is given to the associated online auction algorithm. Let OPT_i denote the optimal profit achievable using a fixed price for item i from customers whose bundles contain item i but no other item in V_L. Using the results of [4], the expected profit of the online auction for item i will therefore be at least $(1-\epsilon)\mathrm{OPT}_i - O(\frac{h}{\epsilon}\log\frac{1}{\epsilon})$, where $\epsilon > 0$ is an input to the online algorithm and h is the maximum valuation of any customer seen so far. Thus, overall, we achieve profit at least $(1-\epsilon)\sum_{i\in V_L}\mathrm{OPT}_i - O(\frac{nh}{\epsilon}\log\frac{1}{\epsilon})$, where $\sum_{i\in V_L}\mathrm{OPT}_i$ is the profit of the *offline* approximation algorithm. In particular, so long as the offline algorithm's profit is $\Omega(\frac{nh}{\epsilon^2}\log\frac{1}{\epsilon})$, we lose only a $(1+O(\epsilon))$ factor in conversion to the online setting. Note that we need the assumption of an oblivious adversary for the approximation ratios proved in Sections 3 and 4 to apply.

In the posted-price setting, we can also apply the associated posted-price algorithms of [4, 5]. The only tricky issue is that a customer who chooses not to buy anything must be fed in as a non-buyer to *all* of the online algorithms, in order to ensure that the sequence of customers fed into algorithm i is a superset of the true customers for that item. In addition, the algorithms for the posted-price scenario require that the upper-bound h on the maximum valuation be known in advance.

7.3 The Online Highway Problem

For the highway problem, we cannot decompose our solution into a collection of independent digital-good auctions, so the reduction in Section 7.2 does not go through. However, for the case that all path requests have a common endpoint (for which Guruswami et al. [13] give an efficient exact algorithm using dynamic programming), we *can* convert to the online setting by placing this problem in the framework of *online geometric optimization* studied by Kalai and Vempala [16]. In particular, [16] gives a method to convert any efficient *exact* algorithm for offline optimization into an efficient near-optimal algorithm for *online* optimization, for any problem of the following type:

1. There is a set $S \subseteq \mathbb{R}^d$ of *feasible points*. At each time step t we must pick some point $\mathbf{p}^t \in S$, we are then given an objective function $\mathbf{v}^t \in \mathbb{R}^d$, and we obtain profit $\mathbf{p}^t \cdot \mathbf{v}^t$.

2. Our goal is to perform nearly as well as the best point $\mathbf{p} \in S$ in hindsight. That is, we want $\sum_t \mathbf{p}^t \cdot \mathbf{v}^t$ to be nearly as large as $\max_{\mathbf{p}\in S}\sum_t \mathbf{p}\cdot\mathbf{v}^t$.

3. We have an efficient algorithm for the *offline* optimization problem: given objective function $\mathbf{v} \in \mathbb{R}^d$, find the point $\mathbf{p} \in S$ that maximizes $\mathbf{p} \cdot \mathbf{v}$.

Kalai and Vempala [16] give a procedure for choosing points $\mathbf{p}^t$ online for any problem of the above type such that the total profit obtained, $\sum_t \mathbf{p}^t \cdot \mathbf{v}^t$, is within a $1-\epsilon$ factor of the profit of the best $\mathbf{p} \in S$ in hindsight, minus an additive term that is polynomial in the diameter of S and the maximum L_1 magnitude of any $\mathbf{v}^t$.

We can place the highway problem in which all path requests are of the form $\{1, \ldots, i\}$ into the above setting as follows. First, let $d = nh$, where we assume all bidders have integral valuations between 0 and h. Now, let S be the set of all possible item prices represented in the following way. Given a pricing $(p_1, \ldots, p_n)$ of the n items, represent p_i as a vector of length h consisting of $q_i - 1$ zeros followed by $h - q_i + 1$ entries at value q_i, where $q_i = p_1 + \ldots + p_i$ is the price of the i-th bundle. Then concatenate these n vectors together to create a point in $\mathbb{R}^d$. A bidder who desires bundle $\{1, \ldots, i\}$ at value w is represented as a vector of all zero entries except for a 1 in the w-th coordinate of the i-th block. By design, the dot product of this vector with a vector $\mathbf{p} \in S$ is exactly the profit that would be obtained from this bidder by the item-pricing corresponding to $\mathbf{p}$. Finally, we can use the optimization algorithm from [13] as our offline optimization oracle. Since in our case the diameter of S is at most $2nh^2$, and the maximum L_1 magnitude of any $\mathbf{v}^t$ is at most 1, the total profit obtained will be within a $1-\epsilon$ factor of the profit of the best $\mathbf{p} \in S$ in hindsight, minus an additive term which is $\tilde{O}\left(nh^2\right)$.

One somewhat subtle issue is that the Kalai-Vempala algorithm requires running the offline algorithm on objective functions $\mathbf{v}$ that correspond to perturbed versions of the actual history $\mathbf{v}^1 + \ldots + \mathbf{v}^{t-1}$ and so one must be careful that this might not actually correspond to a legal problem input to our offline optimization algorithm. However, because we have represented each bidder as a coordinate basis vector, we can decompose *any* vector $\mathbf{v}$ into a set of legal bidders, and so this is not a problem.

Note that for the posted-price version, we just need to apply known extensions of the Kalai-Vempala algorithm to the bandit setting [2, 19, 7] in which only the profit $\mathbf{p}^t \cdot \mathbf{v}^t$ and not the actual vector $\mathbf{v}^t$ is revealed to the algorithm.

Unfortunately, we do not know how to perform these reductions for the general highway problem, because for that problem we do not have an exact offline algorithm. However, it may be possible to view our approximation algorithm as an exact optimizer for a related problem that can itself be fit into the Kalai-Vempala framework, solving the online problem in that way.

8 Conclusions

We present approximation and online algorithms for a number of problems of pricing items to consumers so as to maximize seller's revenue in an unlimited supply setting. We achieve an $O(k)$-approximation algorithm for the case of single-minded bidders where each consumer wants at most k items, an $O(\log n)$ approximation for the highway problem from [13], and a constant factor approximation to the highway problem when all bidders want approximately (up to a constant factor) the same number of items. We also show how some of our approximation algorithms can be adapted to the more demanding online setting in which customers arrive one at a time, in both the full-information and posted-price settings.

8.1 Open Questions

There are several natural open problems left by this work.

First, can one improve on the factor of 4 for the graph vertex problem? Any method able to reduce the factor of 2 for the bipartite case would immediately result in an improved bound. Alternatively, perhaps the reduction to the bipartite case can be improved. Also, for the general hypergraph vertex pricing problem it would be good to close the gap between the upper bound we present in this paper and the lower bound of [8].

Second, is it possible to possible to perform the reduction in Section 7.3 for the general online highway problem? Or, more generally, can one convert approximation algorithms for these problems into online algorithms without using special properties of the algorithms themselves as done in Section 7.2?

Finally, an intriguing question related to this work is: what kind of approximation guarantees are achievable if one allows the seller to price some items below cost (i.e., to have "loss leaders")? For the case of digital goods, it may not make sense to allow negative prices (customers might purchase infinitely many such items), but in the case of products of fixed marginal cast, a retailer might wish to price some products below cost in order to induce more purchases of bundles containing both those and other more expensive products. For example, consider four items A, B, C, and D, and three customers: one who values $\{A, B\}$ at \$10 above their combined cost, one who values $\{B, C\}$ at \$40 above their cost, and one who values $\{C, D\}$ at \$10 above their cost. If no item can be priced at a loss, then it is not possible to have all three customers buy at their valuations. On the other hand, by pricing A and D at \$10 below cost, and B and C at \$20 above cost, the seller can extract full profit. More generally, we can construct a (bipartite) graph in which there is an $\Omega(\log n)$ gap between the optimal profit achievable without any items priced at a loss and the optimal profit if such pricing is allowed.[5] However, this does not necessarily mean that no $o(\log n)$ approximation is possible, only that our current approaches do not succeed. In particular, we do not know of any constant-factor approximation for the graph vertex pricing problem when negative profit margins on some items are allowed.

Acknowledgements

We would like to thank Jason Hartline for posing the graph vertex-pricing problem to us and for helpful discussions.

References

[1] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos. An Approximate Truthful Mechanism for Combinatorial Auctions with Single Parameter Agents. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 205 – 214, 2003.

[2] B. Awerbuch and R. Kleinberg. Adaptive Routing With End-to-End Feedback: Distributed Learning and Geometric Approaches. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, pages 45 – 53, 2004.

[3] M.-F. Balcan, A. Blum, J. Hartline, and Y. Mansour. Mechanism Design via Machine Learning. In *46th Annual IEEE Symposium on Foundations of Computer Science*, pages 605 – 614, 2005.

[4] A. Blum and J. Hartline. Near-Optimal Online Auctions. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1156 –1163, 2005.

[5] A. Blum, V. Kumar, A. Rudra, and F. Wu. Online Learning in Online Auctions. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 202 – 204, 2003.

[6] P. Briest and P. Krysta. Single-Minded Unlimited Supply Pricing on Sparse Instances. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1093 – 1102, 2006.

[5]Consider a bipartite graph with vertices $\ell_1, \ldots, \ell_n$ on the left and $r_1, \ldots, r_n$ on the right. A set S_1 of bidders each want bundles of the form $\{\ell_i, r_{i+1}\}$ at \$1 above cost, a set S_2 of bidders each want bundles of the form $\{\ell_i, r_{i+2}\}$ at \$2 above cost, a set S_4 of bidders each want bundles of the form $\{\ell_i, r_{i+4}\}$ at \$4 above cost, and so forth, up to $S_{n/2}$. Suppose there are twice as many bidders in S_1 as S_2, twice as many in S_2 as S_4 and so forth, and the bidders in each set are spread equally among all bundles of their associated type. In this case, if negative profit margins are allowed, then one can price each ℓ_i at profit $-i$ and each r_i at profit i and have all bidders buy at exactly their valuations, extracting full profit. On the other hand, the structure of bidders is such that the set of bidders who want any given item form an "equal revenue distribution". This means it is not possible to get substantial profit if one is forced to set all items on one side (left or right) to profit-margin 0. This in turn implies that it is not possible to get substantial profit using only non-negative profit margins, since we know the optimal profit achievable using non-negative profit margins is within a factor of 2 of the profit achievable setting all items on one side of the graph to 0.

[7] V. Dani and T. P. Hayes. Robbing the Bandit: Less Regret in Online Geometric Optimization Against an Adaptive Adversary. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 937 – 943, 2006.

[8] E. D. Demaine, U. Feige, M. Hajiaghayi, and M. R. Salavatipour. Combination Can Be Hard: Approximability of the Unique Coverage Problem. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 162 – 171, 2006.

[9] K. Kawarabayashi E. D. Demaine, M. T. Hajiaghayi. Algorithmic Graph Minor Theory: Decomposition, Approximation, and Coloring. In *46th Annual IEEE Symposium on Foundations of Computer Science*, pages 637 – 646, 2005.

[10] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Velickovic. Efficient Approximation of Product Distributions. *Random Structures and Algorithms*, 1998.

[11] A. Goldberg, J. Hartline, A. Karlin, M. Saks, and A. Wright. Competitive auctions. *Games and Economic Behavior*. (To appear). An earlier version available as InterTrust Technical Report STAR-TR-99.09.01.

[12] A. Goldberg, J. Hartline, and A. Wright. Competitive Auctions and Digital Goods. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 735–744, 2001.

[13] V. Guruswami, J. Hartline, A. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On Profit-Maximizing Envy-Free Pricing. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1164 – 1173, 2005.

[14] M. T. Hajiaghayi. Personal Communication, 2006.

[15] J. Hartline and V. Koltun. Near-Optimal Pricing in Near-Linear Time . In *9th Workshop on Algorithms and Data Structures*, pages 422 – 431, 2005.

[16] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291 – 307, 2005. An earlier version appeared in COLT 2003.

[17] D. Lehmann, L. I. Ocallaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM (JACM)*, 49:577 – 602, 2002.

[18] M. Luby and A. Wigderson. Pairwise Independence and Derandomization. Technical Report: CSD-95-880, 1995.

[19] H. B. McMahan and A. Blum. Online Geometric Optimization in the Bandit Setting Against an Adaptive Adversary. In *Proceedings of 17th Conference on Computational Learning Theory*, pages 109 – 123, 2004.

[20] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.