Approximation Algorithms for Item Pricing

Maria-Florina Balcan[*] **Avrim Blum**[*]

July 2005
CMU-CS-05-176

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

We present approximation algorithms for a number of problems of pricing items for sale so as to maximize seller's revenue in an unlimited supply setting. Our main result is an $\mathcal{O}(k)$-approximation for pricing items to single-minded bidders who each want at most k items. For the case $k = 2$ (where we get a 4-approximation) this can be viewed as the following *graph vertex pricing* problem: given a graph G with valuations v_e on the edges, find prices p_i for the vertices to maximize $\sum_{\{e=(i,j):v_e \geq p_i+p_j\}} p_i + p_j$. We also improve the approximation of [6] from $\mathcal{O}(\log m + \log n)$ to $\mathcal{O}(\log n)$ for the "highway problem" in which all desired subsets are intervals on a line.

Research supported in part by NSF grants CCR-0105488, CCR-0122581, and IIS-0121678.

Keywords: Approximation Algorithms, Auction Design, Profit Maximization, Item Pricing

1 Introduction

Consider the problem of a company 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 soda), 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 *graph* (actually a multigraph with self-loops if some customers have valuations only over individual items), where each edge e has some valuation v_e, and our goal is to set prices p_i on the vertices of the graph to maximize total *profit*: that is,[1]

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

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

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*. Guruswami et al. [6] show an $\mathcal{O}(\log n + \log m)$-approximation for the general hypergraph pricing problem, and 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 [4] 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.

In this paper, we give a 4-approximation for the graph vertex pricing problem, and more generally an $\mathcal{O}(k)$-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).

We also consider the highway problem studied in [6]. 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 $\mathcal{O}(\log n)$-approximation, improving slightly over the $\mathcal{O}(\log m + \log n)$ approximation of [6], and also give an $\mathcal{O}(1)$-approximation for the case that all users want the *same* number of items up to a constant factor. Finally, we give a $(1 + \epsilon)$-approximation using Dynamic Programming for the case that the desired subsets of different customers form a hierarchy (this is defined more precisely in Section 6).

[1]Our model assumes that items have a fixed marginal cost to us, and given that assumption we can without loss of generality view that marginal cost as zero (equating "profit" and "revenue") by simply subtracting the costs of the endpoints from each valuation v_e. That is, v_e now represents the total markup of the two endpoints that the customer is willing to pay, and the p_i are the markups.

[2]Previous work [4, 6] 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 model assumes 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 want an algorithm that is *incentive-compatible*, meaning that it is in bidders' self-interest to reveal their true valuations. If there are sufficiently many bidders, however, then for problems of this type one can convert an approximation to the computational problem into a nearly-as-good approximation to the incentive-compatible auction problem [1]. In particular, the idea is one can randomly partition 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. Results of [1] give bounds on the number of bidders needed for this to work well, and related results of [3, 2, 5] give bounds of this form for the case of a single digital good.

2 Notation and Definitions

We assume we have m customers (also called "consumers" or "bidders") and n items (also called "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). We consider *single-minded bidders*, which means that each customer is interested in only a single bundle of items and therefore valuations can be summarized by a set of pairs (e, v_e) meaning that a customer is interested in bundle (hyperedge) e and values it at v_e. Given the hyperedges e and valuations v_e, we wish to compute a pricing of the items that maximizes seller's profit. We assume that if the total price of the items in e is at most v_e, the consumer will purchase all of the items in e, otherwise the consumer will purchase nothing.[3] That is, we want the price vector $\mathbf{p}$ that maximizes

$$\mathrm{Profit}(\mathbf{p}) = \sum_{\{e: v_e \geq \sum_{i\in e} p_i\}} \sum_{i \in e} p_i.$$

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 bidders, and V the set of items, and let h be $\max_e v_e$. Let $G = (V, E)$ be the induced hypergraph, whose vertices represent the set of items, and whose hyperedges represent the bidders. Notice that G might contain self-loops (since a consumer might be interested in only a single item) and multi-edges (more consumers might want the same subset of items). In the special case that all bidders 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 [6], where a simple $\mathcal{O}(\log m + \log n)$ polynomial time approximation algorithm was also proposed.

[3] Or, if customers want multiple bundles, an equivalent assumption is that they will make their purchase decisions independently over their bundles: that is, they can be split into multiple "virtual" single-minded bidders.

3 A 4-Approximation for 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. 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 $\mathrm{OPT} = \mathrm{OPT}_L + \mathrm{OPT}_R$). We can make 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 that node. In other words, because no edges have two distinct endpoints on the left, 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, we make $\max(\mathrm{OPT}_L, \mathrm{OPT}_R) \geq \mathrm{OPT}/2$.

Now we consider the general (non-bipartite) case. Define w_e to be the amount of profit that OPT makes from edge e. We will think of w_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 $\sum_{e \in E_i} w_e = \mathrm{OPT}_i$ 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 $1/2$ chance of having its endpoints on different sides, in expectation $\mathrm{OPT}_2/2$ weight is on edges with one endpoint in L and one endpoint in R. Thus, if we ignore edges in E_2 whose endpoints are on the same side and run the algorithm for the bipartite case, we make in expectation at least $\frac{1}{2}[\mathrm{OPT}_1 + \mathrm{OPT}_2/2] \geq \mathrm{OPT}/4$.

If desired, this algorithm can be derandomized by partitioning vertices using a pairwise-independent distribution. ■

4 Hypergraph Vertex Pricing

We now show how to extend the previous algorithm to get an $\mathcal{O}(k)$-approximation when each consumer wants at most k items. We begin with a simpler straightforward extension of the previous argument that yields a weaker guarantee.

Theorem 2 *If each consumer is interested in at most k items, then there is a $\frac{k^{k+1}}{k!}$-approximation algorithm for the Hypergraph Vertex Pricing problem.*

Proof: The proof is analogous to the proof of Theorem 1. First notice that if G is *k-partite* with a given k-partition $\{V_1, \ldots, V_k\}$ of V (by this we mean that no customer wants more than one item from any given V_i), then there is a simple k-approximation. Specifically, consider the optimal price-vector $\mathbf{p}^*$ and let OPT_i be the amount of money it makes from nodes in V_i (and therefore $\mathrm{OPT} = \mathrm{OPT}_1 + \cdots + \mathrm{OPT}_k$). As in the proof of Theorem 1 we can make at least OPT_i by setting all prices on the nodes in the other partitions to 0, and then separately fixing prices for each node in partition i so as to make the most money possible on that node.

For the general case, we randomly color the nodes with using k colors to create $V_1, \ldots, V_k$. We throw away from E all edges that contain two distinct vertices with the same color, thus getting the set E'; we then run the algorithm for the k-partite case on E'. For $i = 1, \cdots, k$ let E_i be the subset of hyperedges containing i distinct vertices. Now notice that each edge $e \in E_i$ has a $\frac{k!}{(k-i)!k^i}$ chance of being in E'. If we again weight each edge e by the profit made by OPT on e, this implies that in expectation at least a $\frac{k!}{k^k}$ fraction of the weight of OPT is on edges in E'. We then lose an additional factor of k solving the resulting k-partite problem, completing the proof. ■

We now show how to improve the above bound to $\mathcal{O}(k)$.

Theorem 3 *If each consumer is interested in at most k items, then we can get an $\mathcal{O}(k)$-approximation.*

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 $1/k$.

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 $\mathrm{OPT}_{i,e} \in \{0, \mathbf{p}_i^*\}$ 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 $\mathbf{E}[\sum_{i \in V_L, e \in E'} \mathrm{OPT}_{i,e}]$.

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:

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

Therefore,

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

■

5 The Highway Problem

A particular interesting case considered in [6] 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 solvable in polynomial time ([6]) is the case when all path requests share one common end-point r. For this case, [6] give a $\mathcal{O}(m^2)$ exact dynamic programming algorithm $\mathcal{A}$. [6] also give pseudo-polynomial dynamic programming algorithms for two particular cases. Specifically, they give an $\mathcal{O}(h^{h+2}m^{h+3})$-time exact dynamic programming algorithm for the case when all valuations are integral, and an $\mathcal{O}(h^{k+1}m)$ time exact dynamic programming algorithm for the case that furthermore all requests have path lengths bounded by some constant k.

We now give an $\mathcal{O}(\log n)$ approximation algorithm for the highway problem. We begin by partitioning the customers into $\log_2 n$ groups. Specifically, let S_1 be the set of all customers who want item $n/2$. Let S_2 be the set of all customers not in S_1 who want either item $n/4$ or item $3n/4$. 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 $\{n/2^i, 2n/2^i, \ldots, (2^i - 1)n/2^i\}$. Now, for each set S_i we can use algorithm $\mathcal{A}$ from [6] 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 $jn/2^i$. 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 $jn/2^i$ and then again zeroing out all prices for items to the right of $jn/2^i$ and taking the best of the two cases. Since there are only $\log_2 n$ groups S_i, we get a $2\log_2 n$ approximation overall. Note that we can replace our use of algorithm $\mathcal{A}$ with a special-purpose $(1+\epsilon)$-approximation Dynamic-Programming algorithm, yielding a $(1+\epsilon)\log_2 n$ approximation overall.

Also note that using algorithm $\mathcal{A}$ we can get a constant-factor approximation in the special case when everyone wants exactly k elements, for any arbitrary (not necessarily constant) k. To see this, split the items into groups $G_1, G_2, \ldots, G_{n/k}$ 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 $OPT_{even}/2$ 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 $OPT_{odd}/2$ 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 $2c$ approximation algorithm if everyone wants between k/c and k elements, for any arbitrary k.

6 When bidders form a hierarchy

We now give a $(1+\epsilon)$-approximation for the case that the desired subsets of different (single-minded) customers form a hierarchy. 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.

We will use the fact, shown in [4], that we can cover the space of price vectors with a small set of price vectors $\mathcal{Z}$ such that there exists $\mathbf{p} \in \mathcal{Z}$ such that $\mathrm{Profit}_{\mathbf{p}} \geq \mathrm{Profit}_{\mathbf{p}^*}/(1+\epsilon)$. For completeness, we include the proof of this fact below.

Lemma 1 *There exists* $\tilde{\mathbf{p}}$ *with* $\tilde{\mathbf{p}}_j \in \left\{0 \cup \left[\frac{\delta h}{nm}, h\right]\right\}$ *for all* j *and* $\mathrm{Profit}_{\tilde{\mathbf{p}}} \geq (1-\delta)\mathrm{OPT}$.

Proof: We can assume without loss of generality that $\mathbf{p}^*_j \leq h$, since setting a price above the highest valuation cannot increase the profit. Consider $\tilde{\mathbf{p}}_j = 0$ if $\mathbf{p}^*_j \leq \frac{\delta h}{nm}$ and $\tilde{\mathbf{p}}_j = \mathbf{p}^*_j$ otherwise. Since $\mathrm{OPT} \geq h$ and $\mathrm{Profit}_{\tilde{\mathbf{p}}} \geq \mathrm{Profit}_{\mathbf{p}^*} - \delta h$, we clearly get that $\mathrm{Profit}_{\tilde{\mathbf{p}}} \geq (1-\delta)\mathrm{OPT}$. ■

Let $z_i = \frac{\delta h}{nm}(1+\delta)^i$, and let Z be the set of z_i values in the interval $\left[\frac{\delta h}{nm}, h\right)$, augmented by value 0. Then clearly $|Z| = \lfloor \log_{1+\delta} \frac{nm}{\delta} \rfloor$. Denote by $\mathcal{Z} = Z^n$. Then we can prove that:

Lemma 2 *For any* $\tilde{\mathbf{p}} \in \left[\frac{\delta h}{nm}, h\right]^n$*, there exists* $\mathbf{p} \in \mathcal{Z}$ *such that* $\mathrm{Profit}_{\mathbf{p}} \geq \mathrm{Profit}_{\tilde{\mathbf{p}}}/(1+\delta)$.

Proof: Let $\mathbf{p} \in \mathcal{Z}$ be the price vector obtained by taking the coordinates of $\tilde{\mathbf{p}}$ and rounding them down to the nearest value in Z. Since the price of any bundle under $\mathbf{p}$ is at least the price of the bundle under $\tilde{\mathbf{p}}$ divided by $1+\delta$, we clearly have $\mathrm{Profit}_{\mathbf{p}} \geq \mathrm{Profit}_{\tilde{\mathbf{p}}}/(1+\delta)$. ■

Combining Lemma 1 and Lemma 2, we clearly get (by setting $\delta = \Theta(\epsilon)$) that there exists $\mathbf{p} \in \mathcal{Z}$ such that $\mathrm{Profit}_{\mathbf{p}} \geq \mathrm{Profit}_{\mathbf{p}^*}/(1+\epsilon)$.

The idea of our algorithm is now the following. For $\epsilon > 0$ let $\mathcal{Z}_\epsilon$ be the grid with the property that there exists $\mathbf{p} \in \mathcal{Z}_\epsilon$ such that $\mathrm{Profit}_{\mathbf{p}} \geq \mathrm{Profit}_{\mathbf{p}^*}/(1+\epsilon)$. Let $\mathcal{Z}_\epsilon = Z_\epsilon^n$ and let $Val_\epsilon = \{p_1 + \ldots + p_n | p_i \in Z_\epsilon, 1 \leq i \leq n\}$. We will use an exact dynamic programming algorithm to optimize over $\mathcal{Z}_\epsilon$, which then implies a $(1+\epsilon)$-approximation algorithm for the hierarchy case.

For $s \in Val_\epsilon$ and $e \in E$ denote by n^e_s the number of bidders with desired set e whose valuations are at least s. Note that these values can be computed in polynomial time.

For $e \in E$ and $s \in Val_\epsilon$ denote by $A[s,e]$ 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. 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 now give a dynamic programming algorithm we can use to compute the maximum profit we can extract over $\mathcal{Z}_\epsilon$.

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

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

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

After computing the A-values, we can easily then determine the optimal pricing scheme by backtracking. The overall procedure above runs in time polynomial in n, m, and $1/\epsilon$.

7 Conclusions and Open Questions

We present approximation 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 $\mathcal{O}(k)$-approximation for the case of single-minded bidders where each consumer wants at most k items. We do not know if it is possible to achieve a constant-factor approximation in general for single-minded bidders, or even (as posed in [6]) for the special case of the highway problem. One natural approach might be to show that for any set of bidders there must be some large subset that approximately forms a hierarchy and to somehow reduce to the case of Section 6.

Acknowledgements

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

References

[1] M.-F. Balcan, A. Blum, J. Hartline, and Y. Mansour. Mechanism design via machine learning. Manuscript, 2005.

[2] A. Goldberg, J. Hartline, A. Karlin, M. Saks, and A. Wright. Competitive auctions and digital goods. *Games and Economic Behavior*, 2002. Submitted for publication. An earlier version available as InterTrust Technical Report STAR-TR-99.09.01.

[3] A. Goldberg, J. Hartline, and A. Wright. Competitive Auctions and Digital Goods. In *Proc. 12th Symp. on Discrete Algorithms*, pages 735–744. ACM/SIAM, 2001.

[4] J. Hartline and V. Koltun. Near-optimal pricing in near-linear time. Manuscript, 2005.

[5] Jason Hartline and Robert McGrew. From optimal limited to unlimited supply auctions. In *EC*, 2005. To appear.

[6] V. Guruswami and J. Hartline and A. Karlin and D. Kempe and C. Kenyon and F. McSherry. On Profit-Maximizing Envy-Free Pricing. In *Proc. 16th Symp. on Discrete Alg.* ACM/SIAM, 2005.