

Multi-unit Auctions with Unknown Supply

[Extended Abstract]

Mohammad Mahdian
Microsoft Research
Redmond, WA, USA
mahdian@microsoft.com

Amin Saberi
Stanford University
Stanford, CA, USA
saberi@stanford.edu

ABSTRACT

We study multi-unit auctions for perishable goods, in a setting where the supply arrives online. This is motivated by its application in advertisement auctions on the internet.

We give a $\frac{1}{4}$ -competitive algorithm for computing the optimal single price auction assuming that all the agents report their bids truthfully. We use that algorithm to develop a truthful auction with a constant competitive ratio compared to the optimum offline single-price auction.

1. INTRODUCTION

The main source of revenue for search engines such as Google, Yahoo!, or MSN is a simple auction through which they allocate the advertisement space. In fact search engines run a very busy auction house. Every time a user searches on a keyword, an auction takes place to choose the advertisements that will be showed to the user along with the search results.

The goods that are being sold in these auctions – the search queries – have very interesting characteristics: they should be allocated to the buyers in a fraction of a second or they will perish instantly. Moreover, they arrive in an online fashion and their total supply – the number of times the users search for a specific keyword – is unknown.

The online nature of these auctions gives rise to several interesting and challenging theoretical problems (see, for example, [12]). In this paper, we study the problem of finding optimum reserve prices for keywords. In many situations, having a reserve price, or equivalently reducing the supply of a good, can potentially increase the revenue. In our context, reducing the supply could be particularly useful to increase the competition for keywords with a few number of interested bidders.

The problem of optimizing the revenue of auctions using competitive analysis has received a lot of attention in the computer science community [5, 6, 7, 8, 10]. These mechanisms were initially motivated by their application in selling digital goods. They can also be used in scenarios in which

the total supply of goods is fixed. In this paper, We will extend the framework of those models to the cases where the supply of goods is a priori unknown.

We will consider the following simple multi-unit auction: There are n agents each of them interested in a single unit of a good. The i th agent has a value of u_i for receiving one unit of this good. At any time a new unit of the good might arrive. This unit must be allocated to an agent immediately, or otherwise it will perish. The end of the auction is declared at the end of the day, and the agents are charged at that moment. Notice that the online nature of our model is due to the fact that multiple units of the good arrive online. This is in contrast to other online multi-unit auction problems previously considered in the literature (see, for example, [1, 2, 3]) where the number of units (usually infinite) is known in advance, but the buyers arrive online.

Here we have made the simplifying assumption that each agent is only interested in a single copy of the good. In practice, agents usually either specify a daily budget limit, or a limit on the number of units (often more than one) they are interested in acquiring. We believe that our results can be generalized to such cases, as long as the portion of revenue contributed by any single agent is small compared to the total revenue.

We first study this problem assuming that the agents announce their utilities to the algorithm truthfully. The objective is to give an algorithm that charges each winning agent the same amount, and maximizes the revenue at the end of the day subject to this constraint. The single-price requirement is a natural constraint and provides a reasonable benchmark in settings where the seller does not have any knowledge about the *attributes* of the buyers that can be used for price discrimination (see [2] for a definition of *attribute auctions*).

This problem is in fact the online version of optimal omniscient mechanism. Assuming $u_1 \geq u_2 \geq \dots \geq u_n$, the algorithm can achieve a revenue of $R(i) = iu_i$ by allocating i units of the good. When a new unit arrives, the algorithm should decide whether to allocate the unit to the next person and have the total revenue of $R(i+1) = (i+1)u_{i+1}$ or let the good perish and maintain the same revenue. It is easy to see that $R(i)$ could have several local maxima or minima with very different values. Using that, we can show that any deterministic algorithm will have a very small (in particular, subconstant) competitive ratio.

Interestingly, it is possible to give a randomized algorithm with a constant competitive ratio for this problem. In Section 2, we will give a randomized $\frac{1}{4}$ -competitive algorithm

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

EC'06, June 11–15, 2006, Ann Arbor, Michigan, USA.
Copyright 2006 ACM 1-59593-236-4/06/0006 ...\$5.00.

for this problem. In our algorithm, we flip a fair coin to decide whether or not to allocate the new unit to the next bidder in certain critical points at which the revenue is at a local maximum. Roughly speaking, this will ensure that the algorithm is making the right decision about half of the times! A more careful analysis shows that the competitive ratio of our algorithm is $\frac{1}{4}$ and it is tight. Using Yao's principle, we show an upper bound of $\epsilon/(\epsilon+1)$ for the competitive ratio of any online algorithm.

The more difficult problem is to design a truthful mechanism in this setting that has a revenue competitive to that of the optimal single-price revenue. In Section 3, we use the idea of random sampling optimal price auctions in [7, 5] to solve this problem. The idea is to partition the bids into two subsets A and B randomly and use each subset to compute the reserve price for the other. The arriving goods will be offered to the bidders in each subset alternately using the reserve price calculated by keeping track of the fictitious run of our online algorithm (described in the above paragraph) on the other subset. In case the number of interested bidders is more than one, we have to adopt a delicate tie breaking rule and a non-uniform pricing scheme to maintain the truthfulness.

2. COMPUTING THE OPTIMAL SINGLE PRICE AUCTION

In this section, we give a competitive algorithm for computing the optimal single price revenue, assuming that the utility of the agents are known.

The model. We consider a setting with n buyers, each interested in a single copy of a perishable good. The utility of the i th buyer for this good is denoted by u_i . In this section, we assume that these utilities are publicly known. Also, we assume utilities are sorted in non-increasing order ($u_1 \geq u_2 \geq \dots \geq u_n$). Multiple units of the good arrive online. As each unit arrives, we must decide to either allocate or discard this unit. The algorithm is constrained to sell the good at a single price, and is allowed to change the price of previously sold units over the course of the game. Therefore, at any point in time, the revenue of the algorithm is ku_k , where k is the number of units sold so far.

Notation. Let $\text{OPT}(m)$ denote $\text{argmax}_{1 \leq i \leq m} \{iu_i\}$ and $\text{OPTR}(m)$ denote $\max_{1 \leq i \leq m} \{iu_i\}$. In other words, $\text{OPT}(m)$ is the number of units sold by an optimal single-price auctioneer when the total number of units available is m and $\text{OPTR}(m)$ is the revenue of such an auction.

We use the framework of online competitive analysis to analyze our algorithms: A (randomized) algorithm is c -competitive if for any set of utilities and any value of m , the (expected) revenue of the algorithm after exactly m units of the good arrive is at least $c\text{OPTR}(m)$. The following example illustrates the difficulty in solving this problem, and can be used to show that no deterministic algorithm can achieve a constant competitive ratio for this problem.

EXAMPLE 1. Consider a scenario with one buyer with utility $u_1 = 100$ and 999 buyers with utility $u_2 = u_3 = \dots = u_{1000} = 1$. Any competitive algorithm must allocate the first unit of the good. When the second unit of the good arrives, the algorithm must decide whether or not to allocate this unit. If the algorithm decides to allocate it, then the revenue

Preprocessing:

Sort buyers in decreasing order of their utility ($u_1 \geq \dots \geq u_n$).

Let $m' := 0$ and $F := \text{true}$.

On the arrival of the m th unit:

If $m' < \text{OPT}(m)$ or $m' > \text{OPT}(m)$,

Let $m' := m' + 1$.

Let $p := u_{m'}$.

Sell the newly-arrived unit to buyer m' at price p .

Set the price of all previously-sold units to p .

Let $F := \text{true}$.

else if $F = \text{true}$,

flip a fair coin; if the outcome is head, then

Let $m' := m' + 1$.

Let $p := u_{m'}$.

Sell the newly-arrived unit to buyer m' at price p .

Set the price of all previously-sold units to p .

else

Let $F := \text{false}$.

Figure 1: The Algorithm BestSinglePrice

suddenly drops from 100 to 2. However, if it decides not to allocate this unit, then the algorithm has to face the same decision when the third unit arrives, and so on. If the algorithm never decides to allocate a unit, then after 1000th unit arrives, the revenue of the algorithm is 100, while the revenue of the optimal algorithm is 1000.

The algorithm. We now propose a randomized algorithm (which we call BestSinglePrice). The algorithm keeps track of three parameters: m , the total number of units of the good arrived so far; m' , the number of units allocated so far; and a boolean variable F initialized to **true**. Any time a new unit arrives, we do the following: If $m' < \text{OPT}(m)$ or $m' > \text{OPT}(m)$, then the newly-arrived unit will be allocated to the next agent (i.e., the agent with the highest utility that has not received the good yet), and set F to **true**. If $m' = \text{OPT}(m)$, then if F is **false**, we do nothing (i.e., the newly-arrived unit will not be allocated to anyone), and if F is **true**, we flip a fair coin; if it comes head, we allocate the newly-arrived unit to the next agent, and if it comes tail, we do not allocate it and set F to **false**. The pseudo-code for this algorithm is presented in Figure 1.

THEOREM 1. The above algorithm is $\frac{1}{4}$ -competitive, i.e., for every m , the expected revenue of the solution produced by the algorithm after m units of the good arrive is at least $\text{OPTR}(m)/4$.

PROOF. Let $a_1 < \dots < a_k$ be an ordered list of all numbers in the set $\{x : \text{OPT}(x) = x, 1 \leq x \leq m\}$. Note that this definition implies that $\text{OPT}(m) = a_k$, and more generally, for any $x \in [a_i, a_{i+1})$, $\text{OPT}(x) = a_i$. We define $D := \max_{1 \leq i < k} \{a_{i+1} - a_i\}$. We index the time steps by the

number of units arrived so far, i.e., time t is the time step at which the t 'th unit arrives.

We start by proving the following lemma.

LEMMA 1. *With probability 1, the algorithm sells at least $a_k - D$ units of the good.*

PROOF. We prove by induction on i that at the time that the algorithm has seen exactly a_i units, it has sold at least $a_i - D$ units (with probability 1). The basis of induction is trivial since $a_1 - D = 1 - D \leq 0$. Assume at the time that the algorithm has seen a_{i-1} units, it has sold $x \geq a_{i-1} - D$ units. If $x < a_{i-1}$, by the description of the algorithm, it will sell the next $a_{i-1} - x$ units. Therefore, by the time the algorithm sees a_i units, it has sold at least $x + \min(a_{i-1} - x, a_i - a_{i-1})$ units. Since $D \geq a_i - a_{i-1}$ and $x \geq a_{i-1} - D$, the above number is at least $a_i - D$. \square

We now consider two cases based on whether $D \leq a_k/2$ or $D > a_k/2$.

- Case I: $D \leq a_k/2$. In this case, by the above lemma, the algorithm always sells at least $a_k/2$ units, no matter what random choices it makes. If the algorithm sells less than a_k units, then the price at which it sells these units is at least the price at which the optimal algorithm sells, and therefore, the revenue of the algorithm conditioned on it selling less than a_k units is at least $\text{OPTR}(m)/2$.

On the other hand, any sequence of random choices that causes the algorithm to sell more than a_k units of the good can be matched to a different sequence with the same probability that causes the algorithm to sell exactly a_k units. This is done by changing the last random choice in the sequence. This means that the probability that the algorithm sells more than a_k units is less than or equal to the probability that it sells exactly a_k units. In other words, conditioned on the algorithm selling at least a_k units, the probability that it sells exactly a_k units is at least $1/2$. If this event happens, the revenue of the algorithm will be precisely $\text{OPTR}(m)$. Hence, the expected revenue of the algorithm conditioned on it selling at least a_k units is at least $\text{OPTR}(m)/2$ as well. Therefore, the overall expected revenue of the algorithm is at least $\text{OPTR}(m)/2$.

- Case II: $D > a_k/2$. First, notice that there can be at most one i^* such that $D = a_{i^*+1} - a_{i^*}$. We claim that every run of the algorithm (independent of the outcome of random coin flips) will at some time step $t \leq \min(2a_{i^*}, a_{i^*+1} - 1)$ sell exactly a_{i^*} units. The reason is that in every step after a_{i^*} units have arrived, if the algorithm still has not sold a_{i^*} units, it will sell the newly arrived unit with probability 1. Therefore, at time $t = \min(2a_{i^*}, a_{i^*+1} - 1)$, the algorithm has sold at least $\min(a_{i^*}, t - a_{i^*}) = \min(a_{i^*}, D - 1) \geq \min(a_{i^*}, a_k - D) \geq a_{i^*}$ units.

Now, consider the first time $t \leq \min(2a_{i^*}, a_{i^*+1} - 1)$ at which the algorithm sells exactly a_{i^*} units. In the next time step, the algorithm flips a coin to decide whether to sell the $t + 1$ 'st unit. With probability $1/2$, the algorithm decides to sell this unit. In this case, at the end, the algorithm will sell at least $\min(a_{i^*+1}, a_{i^*} + m - t) \geq$

$\min(a_{i^*+1}, a_{i^*} + a_k - 2a_{i^*}) = \min(a_{i^*+1}, a_k - a_{i^*+1} + D) \geq a_k/2$ units. If the algorithm actually reaches a point where it has sold exactly a_k units, then in the next step, with probability $1/2$ it will not sell the next unit, hence resulting in the optimal revenue. Therefore, the expected revenue of the algorithm conditioned on reaching the point where it sells a_k units is at least $\text{OPTR}(m)/2$. On the other hand, if it does not reach this point, then the selling price will be at least that of the optimal algorithm, and since the algorithm sells at least $a_k/2$ units, it achieves at least half the optimal revenue. Therefore, the overall expected revenue of the algorithm in case II is at least $\text{OPTR}(m)/4$ (the extra factor of 2 comes from the probability $1/2$ with which the algorithm decides to sell the $t + 1$ st unit).

The following example shows that our analysis of this algorithm is tight.

EXAMPLE 2. *The following example shows that the competitive ratio of the above algorithm is not better than $\frac{1}{4}$. Assume the utilities of the bidders are given as follows: $u_1 = 1000$, $u_i = 1$ for $2 \leq i \leq 1001$, and $u_i = 0$ for $i > 1001$. Assume that the number of units is 1002. When the second unit arrives, the algorithm sells this unit with probability $1/2$. If it decides not to sell this unit, it won't sell any unit before the 1001st unit, and it will sell units 1001 and 1002, ending up with a revenue of 3. On the other hand, if it decides to sell the second unit, then at time 1002, it will again flip a coin and sells the 1002th unit with probability $1/2$. If this happens, the revenue is 0; otherwise, the revenue is 1001. Therefore, the overall expected revenue is $1001/4 + 3/2$, whereas the optimum is 1001.*

3. A COMPETITIVE TRUTHFUL AUCTION

In the previous section, we showed how we can approximate the maximum revenue in a setting where the true utilities of the buyers are known, and the good must be sold at the same price to different buyers. In situations where the true utilities of the buyers are not known in advance, auctions provide a suitable way to price the good. In such cases, the challenge is to design a competitive truth-revealing mechanism for the auction, i.e., a mechanism which incentivizes the buyers to bid their true utilities, and then computes a solution which (in expectation) achieves a revenue at least a constant fraction of the optimal single-price auction. In this section, we present such a mechanism and prove that it is truthful and competitive.

The mechanism is as follows: First, randomly partition the set of agents into two sets A and B . Also, fix an arbitrary (bid-independent) ordering of the agents in each set. The units will be allocated alternately to A and B , i.e., odd-numbered units can only be allocated to A and even-numbered units can only be allocated to B . The algorithm also keeps track of two fictitious runs of the algorithm **BestSinglePrice** of the previous section on A and B (run in parallel). As the $(2k - 1)$ 'st unit arrives, we use the fictitious run of the algorithm on B (by fictitious run, we mean a run of **BestSinglePrice** that does not actually allocate the good, but calculates the price at which the good would have been allocated) to compute a price p_k for this unit. Notice that since the algorithm **BestSinglePrice** is randomized, p_k is actually a random variable, and also depends on previous p_i 's. We

then find the first agent of A in the fixed ordering that has utility greater than or equal to p_k , and has not received the good so far. If such an agent i exists, allocate the unit to this agent at price p_k (this price is subject to change in subsequent steps), and reset the price of every agent *before* i in the ordering that has received the good in previous rounds to p_k .¹ If no such i exists, do not allocate the unit, and reset the price of every agent that is allocated the good previously to p_k . The pseudo-code for this mechanism, denoted by `CompMechanism`, is shown in Figure 2.

THEOREM 2. *The mechanism in Figure 2 is truthful and has a constant competitive ratio, i.e., for any set of utilities and any m , the expected revenue of this mechanism after the arrival of m units of the good is at least a constant fraction of `OPTR`(m).*

PROOF. First, we show that the mechanism is truthful. Assume, for contradiction, that there is a scenario with a bidder i that can benefit by mis-reporting her utility. Assume, without loss of generality, that $i \in A$. Notice that the price sequence (p_1, p_2, \dots, p_k) computed from the fictitious run of the algorithm on the set B is entirely independent of the bid of i . Since the mechanism does not use the information about the bid of i in its decisions about bidders that come before i in the ordering, the allocation and the price of these bidders must be the same in the truthful and non-truthful scenarios. This implies that bidder i receives a sequence of offers (p'_1, p'_2, \dots) that is independent of her bid. She receives the good as soon as she accepts one offer (i.e., she sees one offer with p'_j less than or equal to her utility), and her price is determined by the last offer she sees. Thus, if i wins a unit of the good, she has to pay a fixed amount, independent of her bid. Hence, i cannot benefit by mis-reporting her utility.

We now prove that the mechanism has a constant competitive ratio. We analyze the expected revenue from bidders in A . By symmetry, the overall expected revenue will be twice this amount. Also, without loss of generality, we assume that all bids are distinct (We can always add a small symbolic value ϵ_i to the bid of agent i to make the values distinct. These values will not change the payment of an agent, and will only change the allocation if the price is equal to the bid, hence not impacting the incentive properties of our algorithm).

Fix any price vector (p_1, p_2, \dots, p_k) computed from the fictitious run of the algorithm on the set B (i.e., p_i is computed when the $(2i - 1)$ st unit arrives, or in other words when the fictitious run of the algorithm on B has seen i units). We define a_i (b_i , respectively) as the number of elements of A (B , respectively) with utility at least p_i . Also, let k_A and k_B denote the number of units sold to A and B , respectively, assuming that the price offered for the i 'th unit is p_i (Hence, k_A is the actual number of sales to bidders in A , and k_B is the final value of the variable m'_B in

¹Notice that in order to achieve truthfulness, it is essential *not* to reduce the price for agents that are after i in the ordering. To see this, consider a scenario with two agents of utility 4 in A and two agents of utility 5 and 3 in B . If the number of units that arrive is 4 (i.e., 2 for each set), then our mechanism allocates only one unit of the good to A . Thus, the second agent in A (in the fixed ordering) has an incentive to bid a higher value, say 5, to receive the first unit of the good and not let it perish, knowing (or hoping) that the price will later drop to 3.

Preprocessing:

Partition the set of bidders randomly into two sets A and B by putting each bidder in one of the sets independently with probability $1/2$.

Let (π_1, \dots, π_n) be an arbitrary bid-independent ordering of buyers.

Let $m'_A = m'_B = 0$ and $F_A = F_B = \text{true}$.

On the arrival of the m th unit:

If m is odd,

Let $C := A$ and $D := B$.

else

Let $C := B$ and $D := A$.

// compute the sale price p from D :

If $m'_D < \text{OPT}(\lceil m/2 \rceil)$ or $m'_D > \text{OPT}(\lfloor m/2 \rfloor)$,

Let $m'_D := m'_D + 1$.

Let p be the m'_D th largest utility in D .

Let $F_D := \text{true}$.

else if $F_D = \text{true}$,

flip a fair coin; if the outcome is head, then

Let $m'_D := m'_D + 1$.

Let p be the m'_D th largest utility in D .

else

Let $F_D := \text{false}$.

// offer to C at price p :

for $i = 1$ to n do if $\pi_i \in C$,

if π_i has already received the good,

reduce the price of π_i to p .

else

if $u_{\pi_i} \geq p$, then

sell the newly-arrived unit to π_i at price p .

break out of the “for” loop.

Figure 2: The mechanism `CompMechanism`

the algorithm in Figure 2). Notice that a_i 's, b_i 's, k_A , and k_B are all functions of the sequence (p_1, \dots, p_k) . We have the following two lemmas.

LEMMA 2. *If (p_1, p_2, \dots, p_k) is an arbitrary price vector computed from the fictitious run on B (i.e., the fictitious run outputs this sequence of prices with non-zero probability), then $k_B = b_k$.*

PROOF. This follows from the fact that the algorithm `BestSinglePrice` never decreases the price to anything below the “next” bidder’s utility. In other words, b_{i+1} is at most $b_i + 1$. (Notice that here we are using the assumption that bids are distinct). Therefore, at the end, every agent with bid at least p_k receives a unit of the good in the fictitious run. \square

LEMMA 3. *We have $k_A = k - \max_{0 \leq i \leq k} \{i - a_i\}$ and $k_B = k - \max_{0 \leq i \leq k} \{i - b_i\}$.*

PROOF. Construct a bipartite graph with bidders in A on the first side, and units of the good on the second side. Connect a bidder to a unit if the price offered for that unit at the time of its arrival is at most the utility of the bidder. It is easy to see that k_A is the size of the maximum matching in this bipartite graph. By Hall's condition, this value is equal to $k - \max_S \{|S| - |N(S)|\}$, where the maximum is taken over all subsets of vertices of the second side (i.e., units of the good), and $N(S)$ is the set of neighbors of vertices in S . Because of the special structure of this graph, this maximum is always achieved when S is the set of i most expensive units, for some i . This gives $|S| = i$ and $|N(S)| = a_i$. Hence, $k_A = k - \max_{0 \leq i \leq k} \{i - a_i\}$. The equation for B can be proved similarly. \square

We also use the following lemma, which is a consequence of the main lemma proved in [5].

LEMMA 4. For any price p , let a_p (b_p , respectively) denote the number of agents in A (B , respectively) with bid greater than or equal to p . Then with probability at least a constant, for every price p , $|a_p - b_p|$ is at most $\min(a_p, b_p)/10$.

Let \mathcal{E} denote the event that for every price p , $|a_p - b_p| \leq \min(a_p, b_p)/10$. The previous lemma guarantees that this event happens with constant probability. Notice that the event \mathcal{E} only depends on the random partitioning of the set of bidders into $A \cup B$, and does not depend on other random choices of the algorithm. Now, we fix a partitioning $A \cup B$ of the bidders satisfying \mathcal{E} , and compute the expectation of the revenue of the mechanism, where the expectation is taken over other random choices of the mechanism (other than choosing A and B). For such A and B , we have

$$\begin{aligned} |k_A - k_B| &= \left| \max_{0 \leq i \leq k} \{i - a_i\} - \max_{0 \leq i \leq k} \{i - b_i\} \right| \\ &\leq \max_{0 \leq i \leq k} |(i - a_i) - (i - b_i)| \\ &\leq \max_{0 \leq i \leq k} \left\{ \frac{\min(a_i, b_i)}{10} \right\} \\ &= \min(a_k, b_k)/10 \\ &\leq k_B/10. \end{aligned}$$

Now, notice that if the total number of units is $2k - 1$ or $2k$, and the sequence of prices computed by the fictitious run on B is (p_1, \dots, p_k) , then our mechanism sells k_A units to bidders in A each at a price greater than or equal to p_k . Therefore, the expected revenue derived from bidders in A is $E[k_A p_k] \geq \frac{9}{10} E[k_B p_k]$. But $E[k_B p_k]$ is the expected revenue of the fictitious run on B , which by the result of the previous section, is at least $1/4$ times the optimal offline revenue for bidders in B , assuming that k units arrive. Therefore, for any fixed A and B satisfying the event \mathcal{E} , the expected revenue from bidders in A is at least a constant fraction of the optimal single-price revenue of selling k units to the bidders in B .

Now, assume that the optimal solution (for all bidders) sells k^* units and has revenue OPT . If event \mathcal{E} happens, the number of units sold to bidders in B in the optimal solution is at least $\frac{9}{20}k^*$ and at most $\frac{11}{20}k^*$. Therefore, for any A and B satisfying \mathcal{E} , the revenue of selling at most $k^*/2$ units to bidders in B is at least $10/11$ of the revenue derived from the bidders in B in the optimal solution, which

in turn is at least $9/20$ of OPT . Therefore, with constant probability, our algorithm derives a revenue of at least

$$\frac{9}{40} \times \frac{10}{11} \times \frac{9}{20} OPT > OPT/20.$$

Hence the competitive ratio of this mechanism is a constant. \square

4. AN UPPER BOUND USING LINEAR PROGRAMMING

In this section, we study upper bounds on the competitive ratio of the maximum single-price revenue problem studied in Section 2, and give a linear programming-based approach for computing the optimal competitive ratio on a given instance of the problem. Our upper bound for this problem is $e/(e+1)$, leaving a gap with the lower bound of $1/4$ given in Theorem 1. We suspect that the upper bound of $e/(e+1)$ is indeed the right answer.

Assume we are given the utilities $u_1 \geq \dots \geq u_n$ of the buyers. For every $m \leq n$, we let $R(m) = mu_m$ be the revenue of selling exactly m units. Let \mathcal{A} be an algorithm for solving the problem. For $j \leq i \leq n$, we define x_{ij} as the probability that \mathcal{A} sells exactly j units after it sees the i 'th unit of the good. If the number of units that arrive is m , then the expected revenue achieved by \mathcal{A} is precisely

$$\sum_{j=0}^m R(j)x_{mj}.$$

Furthermore, for x_{ij} 's to correspond to the probabilities defined above, there must be y_{ij0} and y_{ij1} such that for every i and j ,

$$x_{ij} = y_{ij0} + y_{ij1}$$

and

$$x_{ij} = y_{i-1,j,0} + y_{i-1,j-1,1}.$$

Therefore, the competitive ratio of any algorithm \mathcal{A} on the instance u_1, \dots, u_n is at most the solution of the following linear program.

$$\begin{aligned} &\text{maximize} && z \\ &\text{subject to} && \forall m : \sum_{j=0}^m R(j)x_{mj} \geq \text{OPT}(m) \times z \\ & && \forall 0 \leq j \leq i \leq n : x_{ij} = y_{ij0} + y_{ij1} \\ & && \forall 1 \leq j \leq i \leq n : x_{ij} = y_{i-1,j,0} + y_{i-1,j-1,1} \\ & && \forall 0 \leq i < j : x_{ij} = 0 \\ & && \forall 1 \leq i : x_{i,0} = y_{i-1,0,0} \\ & && x_{0,0} = 1. \end{aligned}$$

In fact, it is not hard to see that for any feasible solution of the above program, there is an online algorithm that achieves a competitive ratio of z on the given set of utilities. Therefore, the optimal competitive ratio of any online algorithm on the instance u_1, \dots, u_n is precisely the solution of the above linear program. We call this program the *factor-revealing linear program* (see [11] for a discussion about such programs).

Using this approach, we can compute an upper bound on the best competitive ratio achievable for this problem by solving the program 4 on a particular instance. In particular, if we solve this program on an instance consisting of one bidder with value $M = 3, 4$ and a large number of bidders with value 1, we obtain the upper bounds of $6/7$ and $9/11$. In fact, in order to get an upper bound on the competitive ratio, instead of solving the above linear program, it is enough to find a feasible solution for its dual. It is not hard to see that this is essentially the same as using Yao's principle [13]. Using this idea, we are able to prove the following upper bound on the best competitive ratio for this problem.

THEOREM 3. *Any algorithm for computing the maximum single-price revenue with unknown supply has a competitive ratio of at most $e/(e + 1)$, where e is the base of natural logarithm.*

PROOF. Consider an instance that consists of one bidder with utility M and an infinite number of bidders with utility 1. The number of units of the good is a random variable with the following distribution: the probability that this number is i is proportional to $e^{-i/M} \max(1, \frac{i}{M})$. In other words, this probability is equal to

$$p_i = c e^{-i/M} \max(1, \frac{i}{M}), \quad (1)$$

where c is chosen so that

$$\sum_{i=1}^{\infty} p_i = 1. \quad (2)$$

By Yao's principle, the competitive ratio of the best *deterministic* algorithm on this instance is an upper bound on the competitive ratio of the best randomized algorithm for the problem. It is easy to see that the best deterministic algorithm on this instance is of the following form: the algorithm sells the first unit of the good, does not sell the next $k - 1$ units, and sells every unit after that. The expected competitive ratio of this algorithm can be written as follows:

$$\begin{aligned} ALG_k &= \sum_{i=1}^k p_i \cdot \frac{M}{\max(M, i)} + \sum_{i=k+1}^{\infty} p_i \cdot \frac{i - k + 1}{\max(M, i)} \\ &= \sum_{i=1}^k c e^{-i/M} + \sum_{i=k+1}^{\infty} c e^{-i/M} \frac{i - k + 1}{M} \end{aligned}$$

Therefore, we need to compute the maximum of ALG_k over all k , where c is defined using equations (1) and (2). Letting M tend to infinity, this value can be approximated by the following integral expression:

$$\gamma := \max_{t \geq 0} \left(\int_0^t c e^{-x} dx + \int_t^{\infty} c e^{-x} (x - t) dx \right),$$

where c is given by

$$c \cdot \int_0^{\infty} e^{-x} \max(1, x) dx = 1.$$

Simple calculation gives $\gamma = c = e/(e + 1)$. Therefore, the expected competitive ratio of any deterministic algorithm on this instance is at most $e/(e + 1)$. Thus, no randomized algorithm can guarantee a competitive factor larger than $e/(e + 1)$ on every instance. \square

5. CONCLUSION

In this paper, we studied multi-unit auctions for perishable goods, when the supply arrives online. We derived a competitive algorithm for this problem when the utilities are known, and a truthful mechanism when the utilities are private information. We showed lower and upper bounds of $1/4$ and $e/(e + 1)$ on the best competitive factor for this problem (in the case of known utilities). It would be interesting to close this gap. Our conjecture is that the upper bound is the right answer.

Another interesting direction is to generalize this result to the setting where there are multiple types of goods. Unfortunately, even defining a benchmark similar to the optimal single-price revenue in this case becomes a challenge, as it is related to the envy-free pricing problem defined in [9].

Finally, it is curious to notice the similarity between the problem studied in Section 2 and the classical ski rental problem [4]. It would be interesting to find a common generalization of both problems.

Acknowledgments. We would like to thank Nicole Immorlica for helpful discussions.

6. REFERENCES

- [1] Z. Bar-Yosef, K. Hildrum, and F. Wu. Incentive-compatible online auctions for digital goods. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002)*, 2002.
- [2] A. Blum and J.D. Hartline. Near-optimal online auctions. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005)*, 2005.
- [3] A. Blum, V. Kumar, A. Rudra, and F. Wu. Online learning in online auctions. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2003)*, 2003.
- [4] R. El-Yaniv, R. Kaniel, and N. Linial. Competitive optimal online leasing. *Algorithmica*, 25(1):116-140, 1999.
- [5] U. Feige, A. Flaxman, J.D. Hartline, and R.D. Kleinberg. On the competitive ratio of the random sampling auction. In *Proceedings of the 1st Workshop on Internet and Network Economics (WINE 2005)*, 2005.
- [6] A. Fiat, A.V. Goldberg, J.D. Hartline, and A. Karlin. Competitive auctions and digital goods. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002)*, 2002.
- [7] A.V. Goldberg, J.D. Hartline, and A. Wright. Competitive auctions for multiple digital goods. Technical report, Technical Report STAR-TR-00.05-02, InterTrust Technologies Corp, 2000.
- [8] A.V. Goldberg, J.D. Hartline, and A. Wright. Competitive auctions and digital goods. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001)*, 2001.
- [9] V. Guruswami, J.D. 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 (SODA 2005)*, 2005.

- [10] J.D. Hartline and R. McGrew. From optimal limited to unlimited supply auctions. In *Proceedings of the 6th ACM conference on Electronic commerce (EC 2005)*, pages 175–182, New York, NY, USA, 2005. ACM Press.
- [11] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V.V. Vazirani. Approximation algorithms for facility location via dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, November 2003.
- [12] A. Mehta, A. Saberi, U.V. Vazirani, and V.V. Vazirani. Adwords and generalized on-line matching. In *Proceedings of the 46th Annual Symposium on Foundations of Computer Science (FOCS 2005)*, pages 264–273, 2005.
- [13] A.C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science (FOCS 1977)*, 1977.