Dynamic Pricing for Impatient Bidders

Nikhil Bansal* Ning Chen†‡ Neva Cherniavsky†§

Atri Rudra†¶ Baruch Schieber* Maxim Sviridenko*

Abstract

We study the following problem related to pricing over time. Assume there is a collection of bidders, each of whom is interested in buying a copy of an item of which there is an unlimited supply. Every bidder is associated with a time interval over which the bidder will consider buying a copy of the item, and a maximum value the bidder is willing to pay for the item. On every time unit the seller sets a price for the item. The seller's goal is to set the prices so as to maximize revenue from the sale of copies of items over the time period.

In the first model considered we assume that all bidders are *impatient*, that is, bidders buy the item at the first time unit within their bid interval that they can afford the price. To the best of our knowledge, this is the first work that considers this model. In the offline setting we assume that the seller knows the bids of all the bidders in advance. In the online setting we assume that at each time unit the seller only knows the values of the bids that have arrived before or at that time unit. We give a polynomial time offline algorithm and prove upper and lower bounds on the competitiveness of deterministic and randomized online algorithms, compared with the optimal offline solution. The gap between the upper and lower bounds is quadratic.

We also consider the *envy free* model in which bidders are sold the item at the minimum price during their bid interval, as long as it is not over their limit value. We prove tight bounds on the competitiveness of deterministic online algorithms for this model, and upper and lower bounds on the competitiveness of randomized algorithms with quadratic gap. The lower bounds for the randomized case in both models uses a novel general technique.

*IBM TJ Watson Research Center, Yorktown Heights, NY 10598. *Email:* {nikhil,sbar,sviri}@us.ibm.com

†University of Washington, Seattle, WA 98195. *Email:* {ning,nchernia,atri}@cs.washington.edu.

‡Supported in part by NSF CCR-0105406.

§Supported by the NSF graduate research fellowship.

¶This work was done when the author was visiting IBM TJ Watson Research Center, Yorktown Heights, NY.

1 Introduction

The problems considered in this paper are motivated by the application illustrated in following example. Consider a Video On Demand (VOD) service that multicasts a movie at different times and sets the subscription price dynamically. Suppose that the potential customers submit requests in which they specify an interval of time when they wish to watch the movie and a limit value for their subscription (similar to a "limit order" in the stock market). At each time unit, based on the information available to the VOD server, it sets a subscription price for the time unit. The customers are assumed to be "impatient": they subscribe to the service at the first time unit within their interval whose subscription price is no more than their limit value. The goal of the VOD server is to set the prices to maximize its revenue.

Note that, unlike the situation in the stock market, in our case the seller (the VOD server) has information on the limit orders when it sets the price. To "compensate" for this we also consider an "envy free" variant in which customers subscribe in the time unit within their interval with the *lowest* subscription price, as long as this price is no more than their limit value.

We consider both offline and online versions of the problem. In the offline version the VOD server has full information on current and future limit orders while in the online version it only knows the limit values of the active customers at each time unit.

The pricing problem described above is formalized as follows. Assume there is a collection of bidders, each of whom is interested in buying a copy of an item of which there is an unlimited supply. Every bidder i is associated with a tuple (s_i, e_i, b_i), where the range $[s_i, e_i]$ denotes the time interval over which bidder i will consider buying a copy of the item, and b_i is the maximum amount the bidder is willing to pay for a copy of the item. We refer to the tuple (s_i, e_i, b_i) as the *bid* of bidder i, the quantity b_i as her *bid value*, the interval $[s_i, e_i]$ as her *bid interval*, and s_i and e_i as the start and expiration time respectively.

From now on we assume that the time units in which s_i and e_i are specified are *days*. On every day $t = 1, 2, \ldots, T$, the seller (or the VOD server) sets a price $p(t)$ for the item. The seller's goal is to set the prices $\{p(1), \ldots, p(T)\}$ so as to maximize revenue from the sale of copies of items over the time period.

In the first model considered we assume that all bidders are *impatient*, that is, bidders buy the item on the first day within their bid interval they can afford the price. More formally, bidder i buys (a copy of) the item on the *first* day $t \in [s_i, e_i]$, such that $p(t) \leq b_i$. We call this model the IB-MODEL (where IB stands for impatient bidders). To the best of our knowledge, ours is the first work that considers this model.

In the *offline* setting we assume that the seller knows the bids of all the bidders in advance. In the more realistic *online* setting we assume that on day t, the seller only knows about the bids that have arrived before or on day t, i.e., all bids i such that $s_i \leq t$.[1] In fact, we assume that the seller only knows the value b_i of these bids, and does not necessarily know the expiration date e_i of bid i. We use the classical approach of competitive analysis to study the online model. That is, our aim is to design algorithms for setting prices that minimize the competitive ratio, which is the maximum ratio (over all possible bid sequences) of the revenue of the optimal offline solution to that of the online algorithm.

Our model is closely related to the *pricing over time* variant of the envy-free model first considered by Guruswami *et al.* [7]. Their setting is similar to ours, except that a bidder is sold the item at the minimum price during her bid interval, provided that she can afford it. That is, the bidder buys the item at the price $\min_{t \in [s_i, e_i]} p(t)$ (provided this price is less than b_i). In this model, a bidder is never "envious" of another bidder and the pricing is envy-free [7]. We call this model the EF-MODEL (where EF stands for envy free).

1.1 Notation and Preliminaries

We will assume that the bid values are in the interval $[1, h]$. In the online setting, the value of h is not known to the algorithm. The total number of bidders will be denoted by n. For every bidder i, the quantity $e_i - s_i + 1$ will sometimes be referred to as the *bid length* of bid i. We will say that a bid i is *alive* at time t if $t \in [s_i, e_i]$ and the bidder has not bought a copy of the item by day $t - 1$. For any set of bids B, $OPT(B)$ denotes the optimal offline revenue obtainable from B. OPT will denote the optimal revenue from the set of all input bids. For notational convenience, we will use OPT for both the EF-MODEL and the IB-MODEL, since the model under consideration will always be clear from the context.

For several pricing problems, randomized algorithms that have a logarithmic competitive ratio often follow trivially using the "classify and randomly select" technique. In particular, consider the algorithm that rounds down the bid values to the nearest powers of 2, randomly chooses one of these $O(\log h)$ bid values and sets this same price every day. The expected revenue obtained by this algorithm is at least $1/(2 \log h)$ fraction of the total bid values in the instance, and hence this algorithm is trivially $O(\log h)$ competitive for both the IB-MODEL and the EF-MODEL. For most pricing problems in the literature (including the EF-MODEL) these are essentially the best randomized algorithms known. Our focus in this paper will be to either give algorithms that improve on this straightforward guarantee, or show close to logarithmic lower bounds which suggest that the trivial "classify and randomly select" algorithm is essentially close to the best possible.

1.2 Our Results and Techniques

We show that the offline version of the IB-MODEL can be solved in polynomial time by a dynamic programming based algorithm. The rest of the results are in the online setting. For the EF-MODEL, we show an $\Omega(\sqrt{\log h / \log \log h})$ lower bound on the competitive ratio of any randomized algorithm. This may suggest that the trivial $O(\log h)$ competitive randomized "classify and select algorithm" is close to the best possible in this model. We also show that any deterministic algorithm must have a competitive ratio $\Omega(h)$. Note that the deterministic algorithm that sets the price of 1 every day is trivially h competitive, and hence the lower bound implies that this seemingly trivial algorithm is the best possible (up to a constant factor).

For the IB-MODEL, we give a randomized algorithm with competitive ratio $O(\log \log h)$. We also show that any randomized algorithm has a competitive ratio of $\Omega(\sqrt{\log \log h / \log \log \log h})$, which again may suggest that $O(\log \log h)$ is close to the best possible randomized guarantee in this model. For deterministic algorithms, we show that any deterministic algorithm must have a competitive ratio $\Omega(\sqrt{\log h})$, and present a simple greedy (and well known) deterministic algorithm that is $O(\log h)$-competitive.

Note that our results imply an exponential separation between the EF-MODEL and IB-MODEL in terms of competitive ratio for both deterministic and randomized algorithms. We summarize the results for online algorithms in Table 1.

Technically, the most interesting results of the

[1] Note that we assume that on day t, the seller knows the values of bids that arrive on day t in addition to the ones that have arrived before. This is required in our model to obtain non-trivial results (at least for deterministic algorithms). Otherwise, the adversary can only give bids of duration one, and make the performance arbitrarily bad since these bids expire before the online algorithm is even aware of their value.

	Deterministic		Randomized	
	Upper Bound	Lower Bound	Upper Bound	Lower Bound
IB-MODEL	$O(\log h)^{\dagger}$	$\Omega(\sqrt{\log h})$	$O(\log \log h)$	$\Omega(\sqrt{\frac{\log \log h}{\log \log \log h}})$
EF-MODEL	$O(h)^{\dagger}$	$\Omega(h)$	$O(\log h)^{\dagger}$	$\Omega(\sqrt{\frac{\log h}{\log \log h}})$

Table 1: Our results for online algorithms. (The upper bounds with † are previously known and/or trivial.)

paper are the lower bounds for randomized algorithms. Recall that the "classify and randomly select" algorithm achieves in expectation a revenue of at least a $1/(2\log h)$ fraction of total bid values in the instance. Thus to show, say an $\Omega(\log h)$ lower bound the instance must be such that the optimum can satisfy almost all bids at essentially their bid values, and yet any online algorithm must perform poorly. For any online algorithm to perform poorly, the bids in the instance must be such that their bid intervals have substantial overlap and dependence among each other. However, the goal is to do this without reducing the offline profit significantly.

It is instructive to consider the following "binary tree" like instance where bid intervals have a non-trivial dependence among each other. There is one bid with value h and interval $[0, T]$, two bids with value $h/2$ and intervals $[0, T/2-1]$ and $[T/2, T]$ respectively, four with value $h/4$ and intervals $[0, T/4-1], \ldots, [3T/4, T]$ respectively, and so on. We can view this instance as a binary tree in the natural way. The total value of bids in the instance is $O(h \log h)$ and each price level contains a value of h. While any reasonable algorithm can obtain a revenue of h (for example by setting the same price every day), it is a simple exercise to see that in the EF-MODEL no algorithm can achieve a revenue of more than $2h$. Intuitively, if the algorithm sets low prices at some time to gain some bids with low value, it loses all the high value bids that overlap with this time. Interestingly, we use a randomized version of this binary tree like instance to obtain our lower bound for the EF-MODEL. We show that if the instance is such that number of children of each node is an exponentially distributed random variable (instead of exactly two in the binary tree instance) then there are sufficiently many "disjoint" and "high value" regions in the tree such that the offline algorithm can obtain an expected revenue of $\Omega(h \cdot \sqrt{\log h/\log\log h})$. To show this, we analyze a natural branching process associated with this construction and carefully exploit the variance (second order effects) of the exponential distribution. We believe that this technique should be useful in other contexts. To get the bound for the IB-MODEL we define an intricate one-to-many mapping of the bids defined by the binary tree. The mapping ensures that also in this model it "does not pay" to set low prices.

1.3 Related Work Pricing and auctions have received a lot of attention in economics and recently also in computer science literature. In an auction, given the bids (in either offline or online fashion), the auctioneer has to decide on an allocation of items to the bidders and the price to charge them. (Note that in particular every bidder can be charged a different price while in our model *every* bidder is offered the same price on any given day.) Generally, the focus of these works is on one of the following: maximize the *social welfare* of all bidders or maximize the revenue of the seller. Our work falls in the latter category. In the rest of the section, we attempt to summarize a few previous works that are related to ours.

The work closest to ours is that of Guruswami *et al.* [7], which considers the EF-MODEL. They give a polynomial time algorithm to compute the optimal set of prices for the offline version of EF-MODEL, which is based on a dynamic program. In fact, our dynamic program-based algorithm for the offline IB-MODEL is similar in structure to theirs.

For unit bid length setting, Goldberg *et al.* [6] look at competitive *truthful* offline auctions for a single good with unlimited supply, where truthfulness requires that the bidders are best off not lying about their true values. The goal is to design truthful auctions that are still competitive. Online truthful auctions have also been considered for this model [2, 3]. We point out two key differences between this model and ours. First, in the truthful offline auctions model every bidder can be offered a different price. Second, the benchmark is not the best offline performance (as we consider in this work) but the revenue of the auction is compared to that of the best *fixed-price* auction. It is worthwhile to note that the requirement of truthfulness is important in this model as it is trivial to generate optimal revenue without this extra requirement (by selling the good to all bidders at their bid value). Note that we do not consider truthful auctions in this paper.

Auctions for the case when the bid intervals can be arbitrary have been considered in [9, 10, 8]. However, these are in some sense orthogonal to this work as they are either concerned with maximizing the social welfare or the items are available in limited supply. Again, every bidder can be offered a different price and these works deal with truthful auctions. We again note that maximizing the social welfare of an item with unlimited supply without the constraint on truthfulness is trivial by giving the items "for free".

The work of Guruswami *et al.* [7] also considered other envy-free pricing problems. Some progress has been made on those problems. For example, for subsequent work on *single minded pricing* see [1, 5].

2 Results for the EF-MODEL

We present lower bound results for the EF-MODEL. Recall that the algorithm that sets the price of 1 on each day is trivially an h competitive deterministic algorithm. Theorem 2.1 below shows that this is the best possible for any deterministic algorithm (up to a factor of $\phi \approx 1.618$). Next, we describe our randomized lower bound for the EF-MODEL.

THEOREM 2.1. *Any deterministic online algorithm $\mathcal{A}$ for* EF-MODEL *must have a competitive ratio of at least h/ϕ, where $\phi \approx 1.618$ is the "golden ratio".*

Proof. Consider the following game that the adversary plays with the online algorithm $\mathcal{A}$. On day 1, $\phi \cdot h^2$ bids $(1, h^2, h)$ arrive (i.e., each has value h and is valid until day h^2). In addition, on each day $t \geq 1$, h^2 bids $(t, t, 1)$ arrive.[2] These bids arrive either until $\mathcal{A}$ first sets $p(t) = 1$, or until $t = h^2$. At this point the game stops, that is, no more new bids are introduced by the adversary.

Let $t^* \leq h^2$ be the day the game stops. We consider two cases:

Case 1: Algorithm $\mathcal{A}$ never sets its price to 1. In this case $t^* = h^2$. The revenue of $\mathcal{A}$ consists only of the $\phi \cdot h^2$ bids with value h and hence is at most $\phi \cdot h^3$. The optimum sets its price to 1 on each day and thus its revenue is at least $t^* h^2 = h^4$, yielding a ratio of h/ϕ.

Case 2: Algorithm $\mathcal{A}$ sets its price to 1 at time $t^* \leq h^2$. In this case all the $\phi \cdot h^2$ bids with value h contribute only 1 to its revenue (by the property of the EF-MODEL). Additionally, it gets exactly a revenue of h^2 due to the h^2 unit value bids that arrive on day t^*, and hence total revenue is exactly $(1+\phi)h^2$. The optimum sets its price to h on each day until day h^2 and thus its revenue is at least $\phi \cdot h^2 \cdot h = \phi \cdot h^3$, yielding a ratio of $\phi \cdot h/(1+\phi) = h/\phi$. ■

[2] The quantities $\phi \cdot h^2$ and h^2 need not be integral– the correct numbers should be $\lfloor \phi \cdot h^2 \rfloor$ and $\lfloor h^2 \rfloor$. We however, use the expression without the floors for ease of notation– of course the calculation of the asymptotics are not affected. We make similar notational assumptions later in the paper when talking about number of bidders that depend on h.

In the remainder of this section we focus on proving the lower bound on the competitive ratio of any randomized algorithm. Our main result is as follows:

THEOREM 2.2. *Any randomized online algorithm for* EF-MODEL *has competitive ratio of* $\Omega\left(\sqrt{\frac{\log h}{\log\log h}}\right)$.

We use Yao's min-max theorem ([4]). To do this, we define a set of bid instances $I_1, I_2, \ldots$, and a probability distribution D on them. By Yao's principle [4], the quantity $\min_{\mathcal{A}} \frac{\mathbb{E}_{I \leftarrow D} OPT(I)}{\mathbb{E}_{I \leftarrow D} Rev_{\mathcal{A}}(I)}$ is a lower bound on the competitive ratio of any randomized algorithm. Here the minimum is taken over all possible deterministic online algorithms $\mathcal{A}$ and $Rev_{\mathcal{A}}(I)$ is the revenue of algorithm $\mathcal{A}$ on instance I.

Geometric distribution will be a key building block in our lower bound constructions. Let $G(p)$ denote the discrete distribution on positive integers $m = 1, 2, 3, \cdots$ such that $\Pr(m) = (1-p)^{m-1}p$. We need the following well known facts about this distribution.

FACT 2.1. *A random variable X drawn from $G(p)$ has the following properties:*

1. *The expectation is given by* $\mathbb{E}(X) = \frac{1}{p}$
2. $\Pr[X \leq m] = 1 - (1-p)^m$.
3. $\mathbb{E}(X|X > m) = m + \mathbb{E}(X)$, *that is, the geometric distribution is memoryless.*

We also need the following technical fact, the proof of which is deferred to the full version of the paper.

FACT 2.2. *Let k be a fixed positive integer and let c be a real such that $c > k$. Consider the sequence $x_k, x_{k-1}, \ldots, x_0$, where $x_k = 1$ and x_i is recursively defined in terms of x_{i+1} for $i = k-1, \ldots, 0$ as*

$$x_i = 1 + x_{i+1}\left(1 - \frac{1}{c}\right)^{\frac{c}{x_{i+1}}} - (1 + x_{i+1})\left(1 - \frac{1}{c}\right)^{c^2}.$$

Then $x_0 > \sqrt{k}/4$.

We now describe the set of instances for the lower bound. We will not describe the distribution explicitly but instead describe a procedure that will implicitly describe both the instances and the probability distribution over them.

We have the following $k+1$ distinct bid values: $h, h/\log h, h/(\log h)^2, \cdots, 1$. We say bids with bid value $p_i = h/(\log h)^i$ are at level i. Note that $p_0 > p_1 > \ldots >$

p_k and $k = \Theta(\log h / \log\log h)$. To simplify notations, let c denote the quantity $\log h$.

The instances have the property that for each i, the bids at level i are completely disjoint. Moreover, every bid at level i is completely contained inside a bid at level j for each $j < i$. Thus we can view each instance as a tree with k levels (with the root having level 0) where a bid b at level i is a child of a bid b' at level $i-1$ if and only if the bid interval of b is completely contained in the bid interval of b'.

Consider the following procedure for generating random trees. (We refer the reader to Figures 1 and 2 below for an example.) Each tree has k levels. Starting with the root, each node v at level i such that $0 \le i < k-1$, independently generates m_v children, where m_v is chosen from the geometric distribution $G(1/c)$. However, if m_v exceeds c^2 then it is truncated to c^2. Given such a tree instance, we associate an instance with bids as follows: each node at level i is a bid with bid length $(h/c^i)^2 = h^2/c^{2i}$ and bid value h/c^i. If u is the j^{th} child of node v (which is at level i), then the bid corresponding to node u is $(s_v + (j-1)\cdot h^2/(c^{2i+2}), s_v + (j \cdot h^2/c^{2i+2}) - 1, h/c^{i+1})$, where s_v is the start date of the bid corresponding to v. The root node has the bid $(1, h^2, h)$. We will refer to an instance from this distribution by I and use D to denote the induced distribution on the instances.

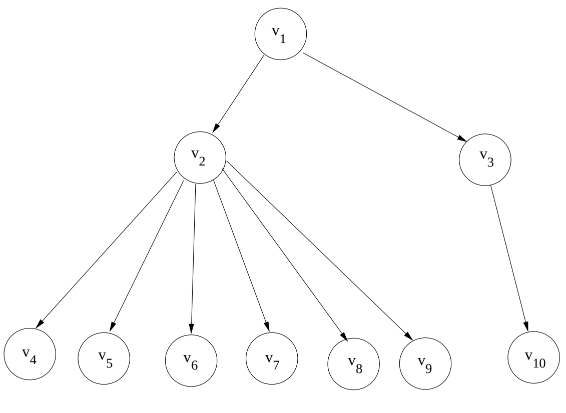

Figure 1: A tree structure which can be generated by the random process described above with $h = 16$, $c = \log h = 4$ and $k = 3$. The root is v_1. The bids corresponding to this example are in Figure 2.

Since the expected number of children of each node is at most c, it follows by a simple inductive argument that expected number of nodes (bids) at level i is at most c^i and hence expected total value of bids in level i is at most $c^i \cdot h/c^i = h$. Thus the expected total value of all the bids in the tree is at most $(k+1)h = \Theta(h \log h / \log\log h) = o(h \log h)$.

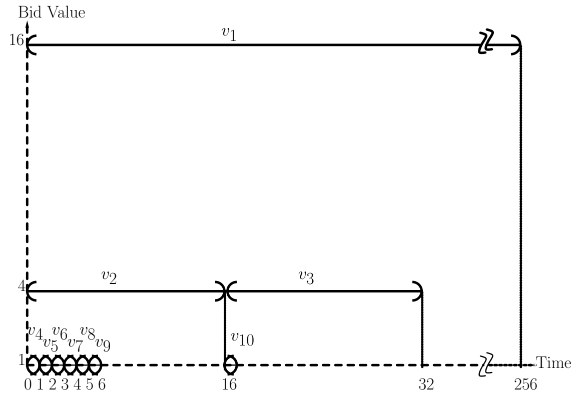

Figure 2: The bids corresponding to the tree structure in Figure 1. The figure is to scale except the time axis is broken between day 32 and day $256 = h^2$.

For technical convenience, we consider the following modified version of the EF-MODEL. For a bid b at level i, if the price is set to a value p_j strictly less than p_i during the duration of b, then b is lost and we obtain a revenue of 0 from b. Note that in the actual EF-MODEL, this bid might yield a revenue of p_j which could be as large as $p_i/\log h$. However, since the expected total value of the bids in the tree is at most $(k+1)h = o(h \log h)$ and the bid values between any two levels differ by at least $\log h$, for any setting of prices, the (additive) difference between the revenue of the EF-MODEL and modified model is at most $(1/\log h)\cdot(k+1)h = o(h)$, which will be insignificant for our purposes.

Our first lemma shows that the expected revenue of any deterministic online algorithm is $O(h)$. This essentially follows from the memoryless property of the geometric distribution.

LEMMA 2.1. *The expected revenue (w.r.t to the distribution D) of any deterministic online algorithm is $O(h)$.*

Proof. We show by induction on the number of levels in the tree that the optimum strategy for the online algorithm in the modified version of EF-MODEL is to set the highest fixed price h at all times and hence best achievable expected revenue is h. Clearly, this is true for the base case of $k = 1$. Inductively, assume that this is the best online strategy for all trees up to depth k. Consider an instance with $k+1$ levels. If the online algorithm decides not to set the (highest) price h at any time $t \in [1, h^2]$, then this bid is lost and yields revenue 0, no matter how prices are set at other times. So the algorithm might as well never set price to h at any time in this case. By the inductive hypothesis, the expected achievable revenue for each subtree of the root is no more than $h/\log h$ and since the expected number of subtrees is strictly less than $\log h$ (since the geometric distribution is truncated at $m_v = c^2$, and hence has mean strictly less than $c = \log h$), the expected revenue is no more than h. Thus the best possible strategy is to set the price to h at all times. ∎

We now show (the harder part) that the expected value of $OPT(I)$, where I is chosen according to D, is quite large. Clearly, Lemma 2.1 and Lemma 2.2 (below) imply Theorem 2.2 by Yao's principle.

LEMMA 2.2. *Let D be the distribution on instances as describe above, then*

$$\mathbb{E}_{I\leftarrow D}[OPT(I)] = \Omega\left(h\sqrt{\frac{\log h}{\log\log h}}\right)$$

Proof. Again, it is convenient to consider the modified EF-MODEL. In this model, given an instance I, $OPT(I)$ can be computed recursively starting from the leaves in a bottom up fashion. In particular, let $Rev(v)$ denote the optimal revenue obtainable from the subtree rooted at v at level i. Let $u_1, \ldots, u_{m_v}$ denote the children of v. Then, the algorithm can either set price p_i at all times during the duration of v, or else try to obtain optimum revenue from each of the subtrees rooted at $u_1, \ldots, u_{m_v}$. Thus we obtain that

$$Rev(v) = \max\left(\frac{h}{c^i}, \sum_{j=1}^{m_v} Rev(u_j)\right). \tag{2.1}$$

Note that given an instance I, $OPT(I) = Rev(r)$, where r is the root. Thus we have that $\mathbb{E}_{I\leftarrow D}[OPT(I)] = \mathbb{E}(Rev(r))$. By definition of expectation, for any node v and any positive real number α, $\mathbb{E}(Rev(v)) = \mathbb{E}\left(Rev(v) \mid m_v \le \alpha\right) \cdot \Pr\left[m_v \le \alpha\right] + \mathbb{E}\left(Rev(v) \mid m_v > \alpha\right) \cdot \Pr\left[m_v > \alpha\right]$. Thus, from (2.1) and the linearity of expectation,

$$\begin{aligned}\mathbb{E}(Rev(v)) \ge& (h/c^i) \cdot \Pr\left[m_v \le \alpha\right] + \\ &\mathbb{E}\left(\sum_{j=1}^{m_v} Rev(u_j) | m_v > \alpha\right) \cdot \Pr\left[m_v > \alpha\right]\end{aligned}$$

Further, note that since the random coin tosses in subtrees rooted at the children $u_1, \cdots, u_{m_v}$ are independent, $\mathbb{E}(Rev(u_1)) = \mathbb{E}(Rev(u_2)) = \cdots = \mathbb{E}(Rev(u_{m_v}))$ and hence,

(2.2)
$$\begin{aligned}\mathbb{E}(Rev(v)) \ge& (h/c^i) \cdot \Pr\left[m_v \le \alpha\right] + \\ &\mathbb{E}(Rev(u_1)) \cdot \mathbb{E}\left(m_v \mid m_v > \alpha\right) \cdot \Pr\left[m_v > \alpha\right]\end{aligned}$$

To simplify notation, we will use x_i to denote the expected optimal revenue generated from any node at level i when the bid values are normalized such that the bid value at level i is 1. That is, for any node v at level i,

$$x_i = \frac{c^i \mathbb{E}(Rev(v))}{h}.$$

Note that by the above definition, $\mathbb{E}(Rev(u_1)) = x_{i+1}h/c^{i+1}$. Thus equation (2.2) can be written as

(2.3)
$$x_i \ge \Pr[m_v \le \alpha] + \frac{x_{i+1}}{c}\mathbb{E}(m_v | m_v > \alpha) \cdot \Pr[m_v > \alpha]$$

Define q to be $(1 - 1/c)$. By Fact 2.1, $\Pr[m_v \le \alpha] = 1 - q^\alpha$. To bound $\mathbb{E}(m_v | m_v > \alpha) \cdot \Pr[m_v > \alpha]$, observe that $\mathbb{E}(m_v | m_v > \alpha) = \alpha + c$ for a geometric distribution. However, we need a slightly more careful accounting since we truncate our distribution at c^2. Thus,

$$\begin{aligned}\Pr[m_v > \alpha] \cdot& \mathbb{E}(m_v | m_v > \alpha) \\ &= \sum_{\alpha < j \le c^2} j q^{j-1}(1/c) + \sum_{j > c^2} c^2 q^{j-1}(1/c) \\ &= (\alpha + c)q^\alpha - (c^2 + c)q^{c^2} + q^{c^2}(c^2) \\ &= (\alpha + c)q^\alpha - c q^{c^2} \\ &\ge (\alpha + c)(q^\alpha - q^{c^2})\end{aligned}$$

Choosing $\alpha = c/x_{i+1}$, and plugging the values above, equation (2.3) can be written as

$$\begin{aligned}x_i &\ge (1 - q^\alpha) + \frac{x_{i+1}}{c}(\alpha + c)(q^\alpha - q^{c^2}) \\ &= (1 - q^\alpha) + (1 + x_{i+1})(q^\alpha - q^{c^2}) \\ &= 1 + x_{i+1}q^\alpha - (1 + x_{i+1})q^{c^2}\end{aligned}$$

As $q = (1 - 1/c)$ and $\alpha = c/x_{i+1}$, the above recursion is exactly that in Fact 2.2. Thus, we have $x_0 > \frac{\sqrt{k}}{4}$ or $\mathbb{E}(Rev(r)) = hx_0 > h \cdot \sqrt{k}/4$ (where r is the root), which proves the lemma. ∎

3 Results for the IB-MODEL

We now consider the IB-MODEL. Recall that in this model, the bidders are impatient and buy the item at the earliest time they can afford it.

3.1 Optimal Offline Algorithm

As in the EF-MODEL, the pricing problem in the offline IB-MODEL can be solved by dynamic programming. Our solution is similar in spirit to that of [7].

THEOREM 3.1. *The optimal set of prices for the offline IB-MODEL can be computed in polynomial time.*

Proof. We describe a dynamic program to compute the optimal revenue (the set of prices will be a by-product). Let the bids be numbered such that the bid values $b_1 \geq b_2 \geq \cdots \geq b_n$ are in decreasing order. Let $p_1 > p_2 > \ldots > p_L$ denote the distinct bid values where $L \leq n$. Note that any optimum algorithm sets prices from the set $\{p_1, \ldots, p_L\}$. (Otherwise, the solution can be trivially improved by increasing the price to the nearest larger element in the set $\{p_1, \ldots, p_L\}$.)

The idea of the dynamic program is the following: consider the optimum solution subject to the constraint that all prices used are at least p_k. If we consider the times where the price is exactly set to p_k, then the solution between every two such consecutive time steps has prices that are at least p_{k-1}. Thus, given precomputed pieces of the solution where the prices are constrained to be at least p_{k-1}, we can stitch these together to obtain a solution where the prices are at least p_k. We now give the details.

For any pair of days s and e, where $s \leq e$, and parameters $\ell \in \{0, 1, 2, \cdots, n\}$ and $k \in \{1, 2, \cdots, L\}$, let $A_k(s, e, \ell)$ denote the optimal revenue obtainable under the following constraints: (1) the subset of bids considered consists only of bids i such that $b_i \geq p_k$ and $s_i \in [s, e]$, (2) $\min_{t \in [s,e]} p(t) \geq p_k$, and (3) ℓ bids with bid value at least p_k are still alive on day $e + 1$. We also define $C_k(s, e)$ to be the optimal revenue obtainable under the following constraints: (1) the subset of bids considered consists only of bids i such that $b_i \geq p_k$ and $s_i \in [s, e]$, (2) $\min_{t \in [s,e-1]} p(t) > p_k$, and (3) $p(e) = p_k$; that is, $C_k(\cdot, \cdot)$ is like $A_k(\cdot, \cdot, 0)$ with the additional constraint that the price p_k is used on the last day.

Let $n^k_{s,t}$ denote the number of bids i with $s_i \in [s, t]$, $e_i \geq t$ and $b_i = p_k$, and let $m^k_{s,t}$ denote the number of bids i with $s_i \in [s, t]$, $e_i \geq t + 1$ and $b_i = p_k$.

We now spell out the recurrence relation for $A_k(s, e, \ell)$ (assuming $\ell > 0$).

$$A_k(s, e, \ell) = \max(A_{k-1}(s, e, \ell - m^k_{s,e}), \max_{t' \in [s,e-1]} (C_k(s, t') + A_k(t' + 1, e, \ell)))$$

Note that for the optimal revenue $A_k(s, e, \ell)$ there are two options: either only use prices greater than or equal to p_{k-1} or use the price p_k somewhere in the time interval $[s, e]$. The first case is captured by the term $A_{k-1}(s, e, \ell - m^k_{s,e})$, we subtract out $m^k_{s,e}$ from ℓ because, by definition, the last argument in $A_{k-1}(\cdot, \cdot, \cdot)$ is the number of bids with value greater than p_{k-1} that are still alive on day $e + 1$. In the second case when the price p_k is used, let t' be the first time it is used. This implies that for days in $[s, t'-1]$ the price is at least p_{k-1}. Then by definition, the revenue obtained on the first t' days is $C_k(s, t')$. Note that any bid with value at least p_k that was alive on day t' cannot be alive on day $t'+1$. This implies that the optimal revenue obtainable from days $[t' + 1, e]$ such that ℓ bids with bid value greater than p_k are alive on day $e + 1$ is $A_k(t' + 1, e, \ell)$. Of course, for the optimal revenue $A_k(s, e, \ell)$ one has to pick the best possible value of t'. This is obtained by the expression $\max_{t' \in [s,e-1]}(C_k(s, t') + A_k(t' + 1, e, \ell))$.

Using similar reasoning and defining for any $\ell < 0$, $A_k(s, e, \ell) = 0$, we get the following recurrence relation:

$$A_k(s, e, 0) = \max(A_{k-1}(s, e, -m^k_{s,e}), \max_{t' \in [s,e-1]} (C_k(s, t') + A_k(t' + 1, e, 0)), C_k(s, e))$$

We now give the recurrence relation for $C_k(s, e)$. Note that in this case the minimum price used in the time range $(s, e - 1)$ is at least p_{k-1}. If there are ℓ' many bids with value greater than p_{k-1} that are alive on day e, then the maximum revenue obtainable from the days $(s, e - 1)$, by definition, is $A_{k-1}(s, e - 1, \ell')$. Further, on day e, $\ell' + n^k_{s,e}$ copies of items are sold at price p_k. Finally optimizing over the choice of ℓ', we get

$$C_k(s, e) = \max_{\ell' \in \{0,1,\cdots,n\}} \left(A_{k-1}(s, e - 1, \ell') + (\ell' + n^k_{s,e})p_k\right)$$

The base cases of the recurrences are pretty simple. For any $s \leq e$ and ℓ

$$A_0(s, e, \ell) = 0$$

$$A_1(s, s, \ell) = 0 \text{ if } \ell \neq 0$$

$$C_1(s, e) = n^1_{s,e} \cdot p_1$$

We are interested in the quantity $A_L(1, \max_{i=1..n} e_i, 0)$. The optimality of the above follows from considering the prices set and the days in non-increasing order. Further, it is easy to see that the prices set by the optimal algorithm have to be among the bid values.

We finally need to show that the dynamic program runs in polynomial time. The number of days considered in the above recurrence relations is $\max_{i=1}^n e_i$ which need not be polynomial in n. However, one can assume w.l.o.g. that $\min_i\{s_i\} = 1$ and $\max_i\{e_i\} \leq n + 1$. To see this note that we may consider only "efficient" algorithms, i.e., algorithms for which $p(t)$, for every time t, is no more than the maximum bid value of the bidders at this time (if such exist). This implies that if there are bidders at day t, at least one of them buys a copy of the item at this day. It follows that by a simple preprocessing the bid intervals can be "shortened" in such a way that either $\max_i\{e_i\} = n + 1$ or there exists $t < \max_i\{e_i\}$ such that t is not contained in any bid interval in which case the problem can be broken into two subproblems. In the preprocessing we scan the bid intervals $[s_i, e_i)$ in increasing order of their start day, and set $e_i = s_i + \ell$, where ℓ is the minimum index such that ℓ bid intervals intersect the interval $[s_i, s_i + \ell)$. Since there are at most n different price levels that are considered by the dynamic program, at most n^3 entries need to be considered for $A_k(\cdot,\cdot,\cdot)$ and at most n^2 entries for $C_k(\cdot,\cdot)$. Further, for each level k, only entries in the level $k-1$ need to be accessed. Thus, the above dynamic program runs in polynomial time. ∎

3.2 Deterministic Online Algorithms Next we focus on online algorithms. In this section we study deterministic algorithms and Section 3.3 contains our results for randomized algorithms. To simplify the analysis we round down each bid value to the closest power of 2. This may decrease the revenue by no more than a factor of 2, which is insignificant since all our bounds are not constants. Thus, from now on we assume that the bid values are powers of 2 and hence lie in the set $\{1, 2, 4, \cdots, h/2, h\}$.

We first show a trivial (and well-known) $O(\log h)$ competitive deterministic algorithm, and then show that any deterministic algorithm has a competitive ratio of $\Omega(\sqrt{\log h})$.

THEOREM 3.2. *The algorithm that on day t only considers the bids that arrive on that day and sets the price that yields the maximum revenue among these bids is $O(\log h)$ competitive.*

Proof. Let $b_{i,t}$ denote the sum of bid values for bids that have bid value 2^i each and arrive at time t. Clearly the optimum is upper bounded by the sum of all bid values, i.e. $OPT \leq \sum_t \sum_{i=0}^{\log h} b_{i,t}$. On the other hand, on each day t the online algorithm obtains a revenue of at least $\sum_i b_{i,t}/(\log h+1)$ (by the pigeon hole principle) on the bids that arrive on day t. Since the bidders are impatient the bids sold on day t are not affected by the prices set on days after t. Thus the online algorithm obtains a revenue of at least $\sum_t \sum_i b_{i,t}/(\log h + 1)$. ∎

THEOREM 3.3. *Any deterministic online algorithm $\mathcal{A}$ for* IB-MODEL *must have a competitive ratio of $\Omega(\sqrt{\log h})$.*

Proof. Consider the following game that the adversary plays with the online algorithm $\mathcal{A}$. On day 1, 2^i bids $(1, \log h, h/2^i)$ arrive, for every $i = 0, 1, 2, \cdots, \log h - 1$. In addition, on each $t \geq 1$, $h\sqrt{\log h}$ bids $(t, t, 1)$ arrive. These bids arrive either until $\mathcal{A}$ first sets $p(t) = 1$, or until $t = \log h$. At this point the game stops, that is, no more new bids are introduced by the adversary.

Let $t^* \leq \log h$ be the day that the game stops. The revenue of the offline algorithm is lower bounded by the revenue obtained using two possible algorithms. The first algorithm is to set price 1 on each day and obtain a revenue of at least $t^* \cdot h\sqrt{\log h}$. The second algorithm sets price $p(t) = 2^{\log h+1-t}$ on day t, for $t = 1, 2, \ldots, \log h$. On each day $t = 1, \ldots, \log h$, this algorithm gets a revenue of h due to the 2^{t-1} bids with value $h/2^{t-1}$, and thus $h \log h$ overall. Thus, $OPT \geq \max\left\{h \log h, t^* \cdot h\sqrt{\log h}\right\}$.

Note that by the way the bids are set up, setting price $p(t) = 2^i$ for $i \geq 1$ results in a revenue of $2^i \cdot \left(\sum_{j\geq i} h/2^j\right) \leq 2h$. It follows that on each day before t^* algorithm $\mathcal{A}$ gets a revenue of at most $2h$ since the price it sets is at least 2. In case $\mathcal{A}$ sets $p(t^*) = 1$ it gets additional revenue of $h\sqrt{\log h}$ from the unit value bids arriving at day t^*. Thus, the competitive ratio of the algorithm is at least

$$\frac{\max\{h \log h, t^* \cdot h\sqrt{\log h}\}}{t^* \cdot 2h + h\sqrt{\log h}} = \Omega(\sqrt{\log h})$$

∎

3.3 Randomized Online Algorithms We first give a randomized $O(\log \log h)$-competitive algorithm for IB-MODEL, and then show that any randomized online algorithm has a competitive ratio $\Omega(\sqrt{\log \log h / \log \log \log h})$.

The randomized algorithm is a "classify and randomly select" algorithm. However, here the classification is according to bid lengths. The following lemmas imply the classification by showing that the bid lengths can be partitioned into $\log \log h + 3$ groups such that there exists an $O(1)$ competitive algorithm if the lengths are limited to be from a single group.

LEMMA 3.1. *Let $k \leq \log h$ be a fixed integer, and consider instances in which the length of every bid lies between $2k$ and $4k$. If k is known in advance, then there is an $O(1)$-competitive randomized algorithm.*

Proof. We divide time into intervals of size k. In particular, for $i \geq 1$, let T_i denote the interval $[(i-1)k + 1, ik]$. Let $V_j(i)$ denote the sum of all bid values for bids with value 2^j that arrive during T_i. Let $j_1(i), j_2(i), \ldots, j_k(i)$ be the k indices with the k highest values of $V_j(i)$. Order these indices such that $j_1(i) > j_2(i) > \ldots > j_k(i)$. Let $\mathcal{V}(i)$ denote the set of these k indices $j_1(i), \ldots, j_k(i)$. Finally, let $R(i)$ denote the value $V_{j_1}(i) + V_{j_2}(i) + \ldots + V_{j_k}(i)$.

Consider the following algorithm that we call $\mathsf{Alg}_{even}(k)$. During T_i, for $i = 2, 4, 6, \ldots$, algorithm $\mathsf{Alg}_{even}(k)$ sets the prices to be 2 to the power of the indices in the set $\mathcal{V}(i-1)$ in decreasing order. Specifically, on the ℓ^{th} day of interval T_i (i.e., day $(i-1)k+\ell$), it sets the price to $2^{j_\ell(i-1)}$. On the other days during intervals $T_1, T_3, T_5, \ldots$, the prices are set to infinity.

Note that $\mathsf{Alg}_{even}(k)$ is a well-defined online algorithm, as $\mathcal{V}(i-1)$ is known at the start of T_i. Also, as each bid has length at least $2k$, every T_i has length k and as the prices during T_{i-1} are set to infinity, the bids that arrive during T_{i-1} are all alive at the start of T_i (and have expiration days outside T_i). Finally, since the prices set during T_i are in a decreasing order, the algorithm $\mathsf{Alg}_{even}(k)$ collects a revenue of at least $R(i-1)$ during T_i. Thus the total revenue of this algorithm is at least $R(1)+R(3)+\ldots$. Analogously, define the algorithm $\mathsf{Alg}_{odd}(k)$ that sets infinite prices during $T_2, T_4, \ldots$ and sets prices in $\mathcal{V}(i-1)$ during T_i, for odd i. It is easy to see that the total revenue of $\mathsf{Alg}_{odd}(k)$ is at least $R(2)+R(4)+\ldots$. Note that both algorithms do not get any revenue for bids that arrive in the last interval of size k. However, by the assumption on the bid length there are no such bids.

Our randomized online algorithm simply tosses one coin at the beginning and either executes $\mathsf{Alg}_{odd}(k)$ or $\mathsf{Alg}_{even}(k)$. We call this algorithm $\mathsf{Alg}(k)$. Clearly, the expected revenue of this algorithm is at least $1/2\sum_{i\geq 1} R(i)$.

We now show that any offline algorithm can get a total revenue of at most $\sum_{i\geq 1} 10R(i)$. Consider the period T_i for some $i \geq 1$. Since each bid has length at most $4k$, the revenue obtained during T_i can only be due to bids that arrived during $T_{i-4}, .., T_i$. Thus it suffices to show that for $q = i-4, \ldots, i$, the revenue that can be obtained during T_i due to bids that arrive during T_q is at most $2R(q)$. Without loss of generality we assume that the prices are also powers of 2. Let $j'_1 > j'_2 > \ldots > j'_\ell$, where $\ell \leq k$, denote the distinct base 2 logarithms of the prices that the offline algorithm sets during T_i. The revenue obtained from bids that arrive during T_q when the price is set to 2^j is at most $\sum_{s\geq 0} V_{j+s}(q)/2^s$. Thus, the total revenue due to bids that arrive during T_q is at most

$$\sum_{r=1}^{\ell}\sum_{s\geq 0}\frac{V_{j'_r+s}(q)}{2^s} = \sum_{s\geq 0}\frac{1}{2^s}\left(\sum_{r=1}^{\ell} V_{j'_r+s}(q)\right) \leq \sum_{s\geq 0}\frac{1}{2^s}R(q) \leq 2R(q).$$

The inequality follows since $R(q)$, by definition, is the sum of the k highest values of the sum of all bids from one level in interval T_q. ∎

Our next observation implies that the problem is easy for instances with bid lengths at least $2\log h + 2$.

Lemma 3.2. *If all bid durations are at least $2\log h+2$, then there is a 2-competitive randomized algorithm.*

Proof. The proof is similar to the proof of Lemma 3.1 the only additional observation is that when $k = \log h + 1$ the revenue obtained in each interval equals the total value of the bids that arrived in the previous interval. Specifically, consider the following two algorithms. The first sets its prices to $\{h, h/2, \ldots, 1\}$ during the first $\log h+1$ time slots, sets price to infinity during the next $\log h+1$ time steps and repeats this pattern forever. The second algorithm sets its price to infinity during the first $\log h + 1$ time slots, sets the prices to $\{h, h/2, \ldots, 1\}$ during the next $\log h + 1$ time slots and repeats this pattern forever. Consider the time partitioned into consecutive intervals of length $\log h + 1$. The profit obtained by the first algorithm is the total value of bids arriving in the even intervals, and the profit obtained by the second algorithm is the total value of bids arriving in the odd intervals. Thus choosing one of these randomly obtains at least half of all bid values. Following previous notation we denote this algorithm $\mathsf{Alg}(\log h + 1)$. ∎

Finally, if all bids have duration 1, the bids arriving on different days do not overlap and hence the instance can be solved optimally, by simply setting the revenue maximizing price on each day. We call this algorithm $\mathsf{Alg}(0)$.

Theorem 3.4. *There is a randomized online algorithm for the* IB-model *with a competitive ratio $O(\log\log h)$.*

Proof. Divide the bids into $\log\log h + 3$ groups according to their bid lengths: group 0 consists of all bids of length 1, group k, for $k = 1, 2, \ldots (\log h)/2, \log h$, consists of all bids whose length lies between $2k$ and $4k$, and group $2\log h$ consists of all bids of length at least $4\log h$. By Lemmas 3.1 and 3.2 and the discussion above if the bid lengths are taken from a single group k then the algorithm $\mathsf{Alg}(k)$ is $O(1)$ competitive. Consider the "classify and randomly select" algorithm that chooses k uniformly at random from the set $S = \{0, 1, 2, 4, \ldots, (\log h)/2, \log h, 2\log h\}$ of cardinality $\log\log h + 3$ and executes the algorithm $\mathsf{Alg}(k)$. Thus, this algorithm is $O(\log\log h)$ competitive. ∎

The algorithm as stated above requires prior knowledge of h. However, this requirement can be removed using rather standard techniques. In particular, the algorithm that begins afresh whenever the current value of h changes by more than a factor of 2, can be shown (rather simply) to yield the same guarantee. We defer the details to the full version.

We now show the lower bound of $\Omega(\sqrt{\log\log h/\log\log\log h})$ on the competitive ratio of any randomized algorithm.

Theorem 3.5. *Any randomized online algorithm for* IB-model *has competitive ratio of $\Omega\left(\sqrt{\frac{\log\log h}{\log\log\log h}}\right)$.*

Proof. (Sketch) We modify the construction in proof of Theorem 2.2. We use a tree construction similar to the one used in the proof of Theorem 2.2, but replace a bid of value v and duration t (say its interval is $(\tau, \tau + t - 1)$), that corresponds to a node in the tree, with t levels of bids such that level i has 2^i bids, each of which has value $v/(t2^i)$ and interval $(\tau, \tau + t - 1)$. Thus, the value v is distributed over t different price levels. The main observation relating the EF-MODEL to the IB-MODEL is the following. Suppose that the algorithm for EF-MODEL obtains the value v from the bid in the interval $(\tau, \tau + t - 1)$, recall that in this case this algorithm sets the price to v during the entire interval $(\tau, \tau+t-1)$. Since in the IB-MODEL bidders are impatient there is no need to fix the price for the entire interval to obtain the value of the bid. Instead, the corresponding algorithm for IB-MODEL sets the price to $v/(t2^i)$ at time $\tau+i$ for $0 \le i \le t-1$ to collect the entire value v that is now distributed over t levels. Clearly, the addition of the levels for each node reduces the depth of the tree and this is the reason for the smaller lower bound. Below, we give the details of the construction.

As in the construction in the proof of Theorem 2.2, we consider the bid instances as a tree. There will be $k+1$ tree levels with the root being at level 0. Define $d = \log\log h$. A node v at level i consists of bids at $d^{2(k-i)}$ distinct bid values that are exponentially decreasing. Let $S_i = \sum_{j=0}^{i} d^{2(k-j)} = (d^{2(k+1)} - d^{2(k-i)})/(d^2 - 1)$ be the total number of bid values from the root to tree level i (inclusive). Let k be such that $2^{S_k} = 2^{(d^{2k+2}-1)/(d^2-1)} = h$. Note that $k = \Omega\left(\frac{\log\log h}{\log\log\log h}\right)$.

For each node v at tree level i, for $\ell = 0, \cdots, d^{2(k-i)} - 1$, there are 2^ℓ many bids

$$\left(s_u + (j-1)d^{2(k-i)}, s_u + jd^{2(k-i)} - 1, \frac{h}{d^i 2^{S_{i-1}+\ell}}\right)$$

where s_u is the start time of the parent u of v (and v is the j^{th} child of u for $1 \le j \le m_u$). Further, each v at level i has m_v children where m_v is chosen from $G(1/d)$. However, if m_v exceeds d^2, then it is truncated to d^2. Finally, the level 0 node has d^{2k} distinct price levels. For every $\ell = 0, \cdots, d^{2k} - 1$, there are 2^ℓ bids $(0, d^{2k} - 1, h/2^\ell)$.

Using arguments similar to the ones used in Theorem 2.2, the optimum strategy for the online algorithm is the analog of staying at the root (which is to visit all the d^{2k} levels in decreasing order during the time $t = 0, \ldots, d^{2k} - 1$, and get a revenue of $O(h)$).

Repeating the argument in the proof of Lemma 2.2 (and considering the analog in the IB-MODEL, as discussed above), one can show that the expected value of the optimal revenue is $\Omega(h\sqrt{k}) = \Omega\left(h\sqrt{\frac{\log\log h}{\log\log\log h}}\right)$ which proves Theorem 3.5. ∎

3.4 *Acknowledgments The authors would like to thank Anna Karlin and Tracy Kimbrel for helpful discussions.

References

[1] M.-F. Balcan, and A. Blum. Approximation algorithms and online mechanisms for item pricing. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 29–35, 2006.

[2] A. Blum, and J. Hartline. Near-optimal online auctions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, 1156–1163, 2005.

[3] A. Blum, V. Kumar, A. Rudra, and F. Wu. Online learning in online auctions. *Theoretical Computer Sciences*, 324(2-3):137–146, 2004.

[4] A. Borodin, and R. El-Yaniv. *On-Line Computation and Competitive Analysis*. Cambridge University Press, 1998.

[5] E. D. Demaine, U. Feige, M. T. Hajiaghayi, and M. R. Salavatipour. Combination can be hard: approximability of the unique coverage problem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, 162–171, 2006.

[6] A. Goldberg, J. Hartline, A. Karlin, M. Saks, and A. Wright. Competitive auctions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, 735–744, 2001. Journal version to appear in Games and Economic Behavior.

[7] V. Guruswami, J. Hartline, A. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On profit-maximizing envy-free pricing. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, 1164–1173, 2005.

[8] M. T. Hajiaghayi, R. D. Kleinberg, M. Mahdian, and D. C. Parkes. Online auctions with re-usable goods. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 165–174, 2005.

[9] M. T. Hajiaghayi, R. D. Kleinberg, and D. C. Parkes. Adaptive limited-supply auctions. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 71–80, 2004.

[10] R. Lavi, and N. Nisan. Online ascending auctions for gradually expiring items. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, 1146–1155, 2005.