Competitive Algorithms for Online Pricing

Yong Zhang[1,2,⋆], Francis Y.L. Chin[2,⋆⋆], and Hing-Fung Ting[2,⋆⋆⋆]

[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
[2] Department of Computer Science, The University of Hong Kong, Hong Kong
{yzhang,chin,hfting}@cs.hku.hk

Abstract. Given a seller with m amount of items, a sequence of users $\{u_1, u_2, ...\}$ come one by one, the seller must set the unit price and assign some amount of items to each user on his/her arrival. Items can be sold fractionally. Each u_i has his/her value function $v_i(\cdot)$ such that $v_i(x)$ is the highest unit price u_i is willing to pay for x items. The objective is to maximize the revenue by setting the price and amount of items for each user. In this paper, we have the following contributions: if the highest value h among all $v_i(x)$ is known in advance, we first show the lower bound of the competitive ratio is $O(\log h)$, then give an online algorithm with competitive ratio $O(\log h)$; if h is not known in advance, we give an online algorithm with competitive ratio $O(h^{3\log^{-1/2} h})$.

1 Introduction

Pricing is an interesting problem from Economics. In computer science, researchers often build theoretical models for some economic events, then solve the problems by using methods from combinatorial optimization. In this paper, we consider the online pricing problem, which is formally defined as follows: Given a seller with m amount of items, a sequence of users $\{u_1, u_2, ...\}$ come one by one, the seller must set the unit price and assign some amount of items to each user on his/her arrival. Items can be sold fractionally. Each u_i has his/her value function $v_i(\cdot)$ such that $v_i(x)$ is the highest unit price u_i is willing to pay for x amount of items. Generally, the more items a user buys, the lower unit price he excepts. Thus, $v_i(x)$ is non-increasing. Let h be the highest value among all $v_i(x)$, i.e., $v_i(x) \leq h$ for all i and x. When user u_i comes, assuming that the seller sets unit price p_i and assigns m_i items to u_i. If $p_i > v_i(m_i)$, user u_i cannot accept this price, thus, no item is bought by u_i. Otherwise, $p_i \leq v_i(m_i)$, u_i will accept this price and pay $m_i \cdot p_i$ to the seller. If there exist $m_i' > m_i$ such that $p_i \leq v_i(m_i')$, user u_i is *partially satisfied*. Otherwise, user u_i is *totally satisfied*. The objective is to maximize the revenue by setting the price and amount of items for each user.

⋆ Research supported by Shenzhen New Industry Development Fund under grant No.CXB201005250021A. and Shanghai Key Laboratory of Intelligent Information Processing, China. Grant No. IIPL-2010-010.
⋆⋆ Research supported by HK RGC grant HKU 7117/09E.
⋆⋆⋆ Research supported by HK RGC grant HKU-7171/08E.

B. Fu and D.-Z. Du (Eds.): COCOON 2011, LNCS 6842, pp. 391–401, 2011.

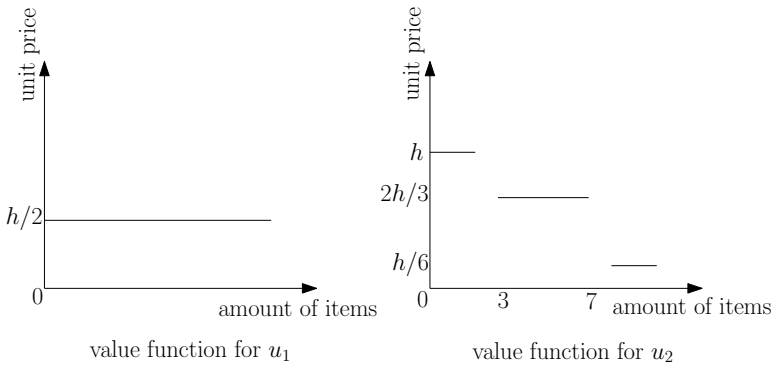

Fig. 1. Example of two value functions

For example, consider the two value functions shown in Figure 1. u_1 is willing to buy any amount of items with unit price $h/2$. In another words, if the unit price of items is no more than $h/2$, u_1 wants to buy as much as possible. If the amount of assigned items is less than 3, the highest unit price u_2 is willing to pay is h; if the amount of assigned items is between 3 and 7, u_2 wants to pay at most $2h/3$ for the unit price; if the amount of assigned items is more than 7, u_2 can afford at most $h/6$ for the unit price. Assuming that the seller assigns 4 items with unit price $h/2$ to u_1 and assigns 6 items with unit price $2h/3$ to u_2, the total revenue of the seller is $4 \cdot h/2 + 6 \cdot 2h/3 = 6h$. In this case, u_1 and u_2 are both partially satisfied.

We consider the online version of the problem, i.e., before the i-th user come, the seller has no information of the j-th user for $j \geq i$. To measure the performance of online algorithms, the competitive analysis is generally used, i.e., compare the outputs between the online algorithm and the optimal offline algorithm, which knows the whole information in advance. In the online pricing problem, the output is the total revenue returned from the algorithm. Given the seller has m amount of items and a user sequence σ, let $A(m, \sigma)$ and $O(m, \sigma)$ denote the total revenue of an online algorithm A and the optimal offline algorithm O, respectively. The competitive ratio of algorithm A is

$$R_A = \sup_{m,\sigma} \frac{O(m, \sigma)}{A(m, \sigma)}.$$

Pricing for items has been well studied during these years. Given n items with unlimited supply, if users are unit-demanded or single-minded, Guruswami et al. [10] showed that a randomized pricing scheme can achieve the revenue within a logarithmic factor of the social welfare. Balcan et al. [3] considered the problem of pricing n items for some unknown users with complex preferences. In some pricing problem, items can be regarded as vertices in a hyper-graph. Each user is interested in some set of items (subgraph of the hyper-graph), and the value function is based on the combination of those items (vertices). The seller

sets price for each item (vertices), if the total price of the interested set of a user is no more than his threshold, the user will buy the items in his interested set. Elbassioni et al. [7] and Grigoriev et al. [9] have studied the profit maximizing pricing in graphs. To maximize the seller's revenue, Balcan et al. [1] gave an $O(k)$-approximation algorithm for single-minded users who wants at most k items. Envy-free is an important property in computational economy, we say a pricing is envy-free if the utility of each user is maximized. Envy-free pricing has been studied in [2,5,6,8,11].

This paper is organized as follows: Section 2 consider the variant in which the highest value h of the value functions is known in advance, we first prove the lower bound of the competitive ratio for this variant is $O(\log h)$, then give a deterministic online algorithm whose competitive ratio matches this lower bound. In Section 3, without knowing h in advance, we give a deterministic online algorithm with competitive ratio $O(h^{3\log^{-1/2} h})$.

2 h Is Known in Advance

2.1 Lower Bound

In this subsection, we show the lower bound of the competitive ratio is at least $O(\log h)$ for the online pricing problem.

Generally, the lower bound of the competitive ratio of an online problem is proved by measuring the performance against an adversary, who knows all information and adjusts the input sequence according to the decisions made by the online algorithms. For this problem, we analyze the pricing procedure step by step. In each step, the adversary decides whether or not sending a user with some fixed value function to the seller. If a user comes, the seller sets the unit price and assigns some amount of items to him. Comparing with the optimal offline pricing scheme, if the total revenue at any step below some threshold, the adversary stops sending users to the seller and the procedure terminates.

- step 1: the adversary sends u_1 to the seller, such that u_1 is willing to pay $2^0 = 1$ for each item.
  If the seller assigns $x_1 \leq m/\lfloor \log h \rfloor$ items to u_1, the adversary stops the procedure and no user comes. The revenue of the online algorithm is at most $m/\lfloor \log h \rfloor$, while the optimal strategy assigns all items to the first user with the total revenue m. Thus, in this case, the competitive ratio is at least $\lfloor \log h \rfloor$.
  Otherwise, the seller assigns $x_1 > m/\lfloor \log h \rfloor$ items to u_1, then the adversary sends u_2 to the seller.
- step 2: u_2 is willing to pay $2^1 = 2$ for each item.
  If the seller assigns x_2 items to u_2 such that $x_1 + x_2 \leq 2m/\lfloor \log h \rfloor$, since $x_1 > m/\lfloor \log h \rfloor$, we have $x_2 < m/\lfloor \log h \rfloor$. The adversary stops the arrival of the following users. The total revenue of the online algorithm is at most $x_1 + 2x_2 \leq 3m/\lfloor \log h \rfloor$, while the optimal strategy assigns all items to u_2 with the total revenue $2m$. Thus, in this case, the competitive ratio is $2\lfloor \log h \rfloor/3$.

Otherwise, the seller assigns x_2 amount of items to u_2 such that $x_1 + x_2 > 2m/\lfloor \log h \rfloor$.
...

– step i: user i is willing to pay 2^{i-1} for item.
If the seller assigns x_i items to u_i such that $\sum_{j=1}^{i} x_j \leq im/\lfloor \log h \rfloor$, the adversary stops the arrival of the following users.

Fact 1. *The total revenue is at most* $2^i m/\lfloor \log h \rfloor$.

Proof. From previous steps, $\sum_{j=1}^{k} x_j > km/\lfloor \log h \rfloor$ $(k < i)$. Thus, we have $\sum_{j=k+1}^{i} x_j < (i-k)m/\lfloor \log h \rfloor$ for $(0 \leq k < i)$. Therefore,

$$\sum_{j=1}^{i} 2^{j-1} x_j \leq \sum_{j=1}^{i} 2^{j-1} m/\lfloor \log h \rfloor < 2^i m/\lfloor \log h \rfloor \qquad \square$$

The optimal strategy assigns all items to u_i with the total revenue $2^{i-1}m$. Thus, in this case, the competitive ratio is at least $\lfloor \log h \rfloor/2$.
Otherwise, the seller assigns x_i items to u_i such that $\sum_{j=1}^{i} x_i > im/\lfloor \log h \rfloor$.
...

– step $\lfloor \log h \rfloor$: user $u_{\lfloor \log h \rfloor}$ is willing to pay $2^{\lfloor \log h \rfloor - 1}$ to each item.
Suppose the seller assigns $x_{\lfloor \log h \rfloor}$ to $u_{\lfloor \log h \rfloor}$. Note that the amount of assigned items cannot larger than the total amount, i.e., $\sum_{j=1}^{\lfloor \log h \rfloor} x_j \leq m$. Similar to step i, we can say that the competitive ratio in this case is also at least $O(\lfloor \log h \rfloor)$.

From the above analysis, we have the following conclusion for the lower bound of the competitive ratio.

Theorem 1. *The competitive ratio of the online pricing problem is at least* $O(\log h)$.

Proof. Since $\sum_{j=1}^{\lfloor \log h \rfloor} x_j$ cannot be larger than m, the procedure must be terminated at some step. From the above analysis, the competitive ratio is at least $O(\log h)$ if the adversary terminates the procedure at any step. Thus, this theorem is true. $\square$

2.2 Online Algorithm

In this subsection, we give an online algorithm for this problem and prove that the competitive ratio of our algorithm matches the lower bound $O(\log h)$ in Section 2.1.

Firstly, we consider a simple example, which give us some heuristic for handling the online pricing problem.

Suppose the seller has h items. When the first user comes with the value function $v_1(x) = 1$ for any x. If the seller sets the unit price to be 1, and assigns all these h items to him, the revenue is h. Then the second user comes with the value function $v_2(x) = h$ for any x. There is no item left for the second user, and the total revenue is h. In this case, the optimal scheme assigns no item to the

first user, assigns all items to the second user with unit price h. The maximal revenue is h^2. However, if the seller doesn't assign any item to the first user, no user comes any more and the revenue is zero. In this case, the optimal scheme assigns h items with unit price 1 to the first user and get the revenue h.

The above example shows that the online strategy which set the price and the amount of assigned items to the extreme value is not good, the competitive ratio of such online strategies is at least h. From the above example, we have the following idea:

Heuristic: when setting the price to some lower value, the amount of assigned items must be bounded by some threshold.

In our online algorithm, we set the unit price for each user to the value of some power of 2, say 2^i. Associate $m/(\lfloor \log h \rfloor + 1)$ items to each price level 2^i. If all items in price level 2^i is assigned to some users, the seller may use the remaining quota from lower price levels to satisfy the user with unit price 2^i. When using the remaining quota from lower price levels, the order must be strictly decreasing, i.e., first try price level 2^{i-1}, then 2^{i-2}, ...

Fact 2. *The unit price for any item in price level 2^i is at least 2^i.*

The amount of items for price level 2^i is $w_i = m/(\lfloor \log h \rfloor + 1)$. Since user with higher price level may use the quota from lower price levels, the maximal amount of items which can be assigned to level $2^{\lfloor \log h \rfloor}$ is

$$\sum_{i=0}^{\lfloor \log h \rfloor} \frac{m}{\lfloor \log h \rfloor + 1} = m$$

The following algorithm describes how to set the unit price and the amount of items when a user comes. In the initial step, compute the maximal possible amount of items which can be used for each price level 2^i:

$$x_i = \frac{(i+1) \cdot m}{\lfloor \log h \rfloor + 1}$$

In the algorithm Pricing, the values of x_i $(0 \leq i \leq \lfloor \log h \rfloor)$ are updated step by step. At any step, x_i is the current maximal amount of items which can be used for price level 2^i. Suppose at some step, we assign m_i items with price 2^i, after that step, x_i will be modified to $x_i - m_i$. If we modify x_i, the values of x_j $(j < i)$ may be updated too. In Algorithm 2, the strategy for modifying available amount of items, we use the quota for higher price first, thus, the quota for lower price may be left. Note that the quota of items for lower price level can be also used for higher price levels. For example, the amount of available items for price levels are: $x_0 = 1$, $x_1 = 6$, $x_2 = 8$, $x_3 = 20$, and $x_4 = 24$. If we set $m_i = 10$ items with price 2^4 to some user, the updated amount will be: $x_0 = 1$, $x_1 = 6$, $x_2 = 8$, $x_3 = 14$, and $x_4 = 14$.

Algorithm 1. Pricing

Let y_j be the largest amount that user i is willing to buy given price 2^j and satisfying $y_j \leq m$.
Let $k = \arg\max_j y_j \cdot 2^j$
if $x_k \neq 0$ **then**
    Set unit price to be $p_i = 2^k$.
    Assign $m_i = \min\{x_k, y_k\}$ items to user i.
    Modify Available Amount of Items.
else $x_k = 0$
    Let $k' = \arg\max_{j>k} y_j \cdot 2^j$ such that $x_{k'} > 0$
    Set the unit price to be $p_i = 2^{k'}$.
    Assign $m_i = \min\{x_{k'}, y_{k'}\}$ items to user i.
    Modify Available Amount of Items.
end if

Algorithm 2. Modify Available Amount of Items: ($p_i = 2^k$ and $m_i = \min\{x_k, y_k\}$)

if $m_i = x_k$ **then**
    $x_j = 0$ for $0 \leq j \leq k$
else $m_i = y_k$
    Let $\ell = \arg\max_j x_k - x_j \geq y_k$.
    for $j = \ell + 1$ to k **do**
        $x_j = x_k - y_k$
    end for
end if

From the algorithm **Pricing**, we say that the price level 2^i is full if $x_i = 0$. After handling the sequence of users by the algorithm Pricing, let k be the highest value such that $x_j = 0$ for any $j \leq k$, i.e., all price levels from 2^0 to 2^k are full but price levels $2^{>k}$ are not full, i.e., $x_j > 0$ for any $j \geq k+1$.

Let ALG be the total revenue of the algorithm Pricing, OPT be the optimal revenue of the optimal offline algorithm. Partition OPT into two parts: OPT1 and OPT2. OPT1 denotes the revenue for users with assigned price no more than 2^k by the optimal offline algorithm, OPT2 denotes the revenue for users with assigned price more than 2^k by the optimal offline algorithm.

Lemma 3. *The ratio between OPT1 and ALG is at most* $O(\log h)$.

Proof. The total amount of assigned items is at most m, thus, OPT1 is at most $m \cdot 2^k$.

Since the price levels from 2^0 to 2^k are all full, from Fact 2, the total revenue for users with price no more than 2^k is at least

$$\sum_{i=0}^{k} \frac{m \cdot 2^i}{\lfloor \log h \rfloor + 1} = \frac{m}{\lfloor \log h \rfloor + 1} \sum_{i=0}^{k} 2^i \geq \frac{m \cdot 2^k}{\lfloor \log h \rfloor + 1}$$

Thus, this lemma is true. □

Lemma 4. *The ratio between OPT2 and ALG is at most* $O(\log h)$.

Proof. We prove this lemma by considering the following two cases.

- User u_i is assigned with unit price $p' > 2^k$ by the optimal offline algorithm and with unit price $2^p > 2^k$ by our algorithm Pricing.
  Let $2^\ell \le p' < 2^{\ell+1}$. From the description of the algorithm Pricing, we have $y_p \cdot 2^p \ge y_\ell \cdot 2^\ell$, where y_p and y_ℓ are the largest amounts of items user u_i is willing to buy given unit price 2^p and 2^ℓ respectively. Since the price level 2^p is not full, that means y_p items are all assigned to this user. Thus, the revenue of the algorithm Pricing on user u_i is $y_p \cdot 2^p$. Since the highest unit price is non-increasing in the value function of any user, we know the largest amount of items u_i is willing to buy given unit price p' is at most y_ℓ. Thus, the ratio between the revenues on u_i by the optimal offline algorithm and our algorithm Pricing is at most 2.
- User u_i is assigned with unit price $p' > 2^k$ by the optimal offline algorithm and with unit price $2^p \le 2^k$ by our algorithm Pricing.
  Let $2^\ell \le p' < 2^{\ell+1}$. Suppose the optimal offline algorithm assigns m' items to u_i. Similar to the above analysis, we have $m' \le y_\ell$, where y_ℓ is the largest amount of items user u_i is willing to buy given unit price 2^ℓ. Thus, the optimal revenue on u_i is $p' \cdot m' \le 2 \cdot 2^\ell \cdot y_\ell$.
  From our algorithm, we know that $y_p \cdot 2^p \ge y_\ell \cdot 2^\ell$. If all these y_p items are assigned to u_i, the revenue of our algorithm on user u_i is no less than half of the optimal revenue on this user. Otherwise, algorithm Pricing assigns x_p items to u_i such that $x_p < y_p$. In this case, algorithm Pricing can only assign x_p items to this user, after that, the price level 2^p is full. In price level 2^p, the amount of items is $m/(\lfloor \log h \rfloor + 1)$. Since

$$\frac{m \cdot 2^p}{\lfloor \log h \rfloor + 1} \ge \frac{y_p \cdot 2^p}{\lfloor \log h \rfloor + 1} \ge \frac{y_\ell \cdot 2^\ell}{\lfloor \log h \rfloor + 1} \ge \frac{m' \cdot p'}{2(\lfloor \log h \rfloor + 1)}$$

  the ratio between the revenues on this user from the optimal offline algorithm and the total revenue in price level 2^p from algorithm Pricing is at most $2(\lfloor \log h \rfloor + 1)$.

Fact 5. *For each level* $2^p \le 2^k$*, there is at most one user with assigned amount* $x_p < y_p$ *by the online algorithm Pricing.*

Proof. From Algorithm Pricing, when the seller sets the unit price 2^p, the remaining amount for this price level is strictly larger than 0. If $x_p < y_p$, the seller will assign all remaining amount of items, say x_p, to this user, and then $x_p = 0$. Thus, the seller will not assign items to some user with unit price 2^p. □

From previous analysis, we know that each item in level $2^p \le 2^k$ is counted at most twice, one for the totally satisfied users and the other for the only one partially satisfied user at price level 2^p.

We may further partition OPT2 into two parts: OPT21 and OPT22. OPT21 is the revenue on those users who are totally satisfied by the algorithm Pricing. OPT22 is the revenue on those users who are partially satisfied by the algorithm Pricing. From above analysis, we know $OPT21 \leq 2 \cdot ALG$ and $OPT22 \leq 2 \cdot ALG \cdot (\lfloor \log h \rfloor + 1)$.

Thus,

$$OPT2 = OPT21 + OPT22 \leq 2 \cdot ALG + 2 \cdot ALG \cdot (\lfloor \log h \rfloor + 1) = O(\log h) \cdot ALG.$$

This lemma is true. □

Combining the above two lemmas, we have the next result.

Theorem 2. *The competitive ratio of the algorithm Pricing is* $O(\log h)$.

3 h Is Not Known in Advance

In Section 2, we assume that the highest unit price among all value functions from all users is known in advance. However, in many realities, the information of a user will be revealed on his arrival. Let h_i denote the highest unit price in the value function $v_i(\cdot)$. That means user u_i is willing to pay unit price h_i if the amount of assigned items is less than some value. In this section, we consider the case in which h_i can be only known to the seller on the arrival of u_i.

Since the seller does not know $h = \max_i h_i$ in advance. When the user with the highest unit price comes, if there is no item for this user, the performance of the seller's strategy may be not good. For example, in a sequence of some coming users, except the last one, all other users have the value functions with quite low unit price. If the seller assigns all items to these users, the last user with very high unit price can not be satisfied. If the total revenue in this case is $O(1)$, the revenue of the strategy that assigns some items to the last user can be $O(h)$. Thus, the performance ratio is achieving $O(h)$.

Similar to Section 2.2, group the prices into levels according to their values: $p \in L_j$ if $2^{j^2} \leq p < 2^{(j+1)^2}$, where $j = 0, 1, ...$ Associate the item set S_i to level L_i such that $|S_0| = m/2$, and $|S_i| = m \cdot 2^{-i-1}$ for $i > 0$.

In our algorithm, when assigning items with unit price 2^{i^2}, we will first use the items from S_i. If all items in S_i are used up, we will use items from lower level sets in decreasing order, first from S_{i-1}, then S_{i-2}, and so on.

Fact 6. *(1) The price of any assigned item in* S_i *is at least* 2^{i^2}. *(2) The amount of items which can be only assigned to users with unit price no less than* $2^{(i+1)^2}$ *is at least* $m \cdot 2^{-i-1}$.

Now we give our algorithm for the online pricing without knowing the highest unit price h in advance. Initially, let $x_i = m \cdot (1 - 2^{-i-1})$, which is the largest amount of items for price 2^{i^2}.

From the above algorithm, we say that the price level 2^{i^2} is full if $x_i = 0$. After handling the sequence of users by the algorithm Pricing2, let k be the highest

Algorithm 3. Pricing2

Let y_j be the largest amount that user i is willing to buy given price 2^{j^2} and satisfying $y_j \leq m$.
Let $k = \arg\max_j y_j \cdot 2^{j^2}$.
if $x_k \neq 0$ **then**
    Set unit price to be $p_i = 2^{k^2}$.
    Assign $m_i = \min\{x_k, y_k\}$ items to user i.
    Modify Available Amount of Items (Algorithm 2).
else $x_k = 0$
    Let $k' = \arg\max_{j>k} y_j \cdot 2^{j^2}$ such that $x_{k'} > 0$
    Set the unit price to be $p_i = 2^{k'^2}$.
    Assign $m_i = \min\{x_{k'}, y_{k'}\}$ items to user i.
    Modify Available Amount of Items (Algorithm 2).
end if

value such that $x_j = 0$ for any $j \leq k$, i.e., all price levels from 2^0 to 2^{k^2} are full but price levels $2^{>k^2}$ are not full, i.e., $x_j > 0$ for any $j \geq k+1$.

Let ALG be the total revenue of the algorithm Pricing2, OPT be the optimal revenue of the optimal offline algorithm. Partition OPT into two parts: OPT1 and OPT2. OPT1 denotes the revenue for users with assigned price no more than 2^{k^2} by the optimal offline algorithm, OPT2 denotes the revenue for users with assigned price more than 2^{k^2} by the optimal offline algorithm.

Lemma 7. *The ratio between OPT1 and ALG is at most* $O(h^{\log^{-1/2} h})$.

Proof. The total amount of assigned items is at most m, thus, OPT1 is at most $m \cdot 2^{k^2}$.

In price level 2^{i^2}, the amount of items is $m \cdot 2^{-i-1}$. Note that the price levels from 2^0 to 2^{k^2} are all full, the total revenue for users with price no more than 2^{k^2} is at least

$$\sum_{i=0}^{k} 2^{i^2} \cdot m \cdot 2^{-i-1} \geq m \cdot 2^{k^2-k-1}$$

Thus, $OPT1/ALG \leq 2^{k+1}$. Since $h \geq 2^{k^2}$, we have $k \leq \log^{1/2} h$. Therefore,

$$\frac{OPT1}{ALG} \leq 2^{k+1} \leq 2^{\log^{1/2} h + 1} = 2 \cdot 2^{\log h \log^{-1/2} h} = 2 \cdot h^{\log^{-1/2} h}.$$

This lemma is true. □

Lemma 8. *The ratio between OPT2 and ALG is at most* $O(h^{3\log^{-1/2} h})$.

Proof. We prove this lemma by considering the following two cases.

- u_i is assigned with unit price $p' > 2^{k^2}$ by the optimal offline algorithm and with unit price $2^{p^2} > 2^{k^2}$ by our algorithm Pricing2.

Let $2^{\ell^2} \le p' < 2^{(\ell+1)^2}$. From the description of the algorithm Pricing2, we have $y_p \cdot 2^{p^2} \ge y_\ell \cdot 2^{\ell^2}$, where y_p and y_ℓ are the largest amount of items user u_i is willing to buy given unit price 2^{p^2} and 2^{ℓ^2} respectively. Since the price level 2^{p^2} is not full, that means y_p items are all assigned to this user. Thus, the revenue of the algorithm Pricing2 on user u_i is $y_p \cdot 2^{p^2}$. Since the highest unit price is non-increasing in the value function of any user, we know the largest amount of items u_i is willing to buy given unit price p' is at most y_ℓ. The optimal revenue on u_i is at most $y_\ell \cdot 2^{(\ell+1)^2}$. Thus, the ratio between the revenues on u_i by the optimal offline algorithm and our algorithm Pricing2 is at most $2^{2\ell+1}$. Since $h \ge 2^{\ell^2}$, we have $\ell \le \log^{1/2} h$. Thus, the ratio is at most

$$2^{2\ell+1} \le 2^{2\log^{1/2} h+1} \le 2 \cdot h^{2\log^{-1/2} h}$$

– u_i is assigned with unit price $p' > 2^{k^2}$ by the optimal offline algorithm and with unit price $2^{p^2} \le 2^{k^2}$ by our algorithm Pricing2.
Let $2^{\ell^2} \le p' < 2^{(\ell+1)^2}$. Suppose the optimal offline algorithm assigns m' items to u_i. Similar to the above analysis, we have $m' \le y_\ell$, where y_ℓ is the largest amount of items user u_i is willing to buy given unit price 2^{ℓ^2}. Thus, the optimal revenue on u_i is $p' \cdot m' \le 2^{(\ell+1)^2} \cdot y_\ell = 2^{2\ell+1} \cdot 2^{\ell^2} \cdot y_\ell$. From our algorithm, we know that $y_p \cdot 2^{p^2} \ge y_\ell \cdot 2^{\ell^2}$.
If all these y_p items are assigned to u_i, the ratio between the revenue by our algorithm on user u_i and the optimal revenue on this user is at most $2^{2\ell+1}$. From previous analysis, this ratio is at most $O(h^{2\log^{-1/2} h})$.
Otherwise, algorithm Pricing2 assigns x_p items to u_i such that $x_p < y_p$. In this case, algorithm Pricing2 can only assign x_p items to this user, after that, the price level 2^{p^2} is full. Since in price level 2^{p^2}, the amount of items is $m \cdot 2^{-p-1}$, we have

$$\begin{aligned} & 2^{p^2} \cdot m \cdot 2^{-p-1} \\ \ge\ & 2^{p^2} \cdot y_p \cdot 2^{-p-1} \\ \ge\ & 2^{\ell^2} \cdot y_\ell \cdot 2^{-p-1} \\ =\ & 2^{(\ell+1)^2} \cdot 2^{-2\ell-1} \cdot y_\ell \cdot 2^{-p-1} \\ \ge\ & p' \cdot m' \cdot 2^{-2\ell-1} \cdot 2^{-p-1} \end{aligned} \tag{1}$$

The ratio between the optimal offline algorithm on those m' amount of items total revenue by Pricing2 on item set S_p is at most

$$2^{2\ell+p+2} \le 2^{3\ell+2} \le 4 \cdot h^{3\log^{-1/2} h}.$$

Fact 9. *For each level $2^{p^2} \le 2^{k^2}$, there is at most one user with assigned amount $x_p < y_p$ by the online algorithm Pricing2.*

Proof. Similar to the proof of Fact 5. □

From previous analysis, we know that each item in level $2^{p^2} \le 2^{k^2}$ is counted at most twice: one for the totally satisfied users; the other for the only one partially satisfied user at price level 2^{p^2}, see equation (1).

We may further partition OPT2 into two parts: OPT21 and OPT22. OPT21 is the revenue of those users who are totally satisfied by the algorithm Pricing2. OPT22 is the revenue of those users who are partially satisfied by the algorithm Pricing2. From above analysis, we know $OPT21 \leq O(h^{2\log^{-1/2}h}) \cdot ALG$ and $OPT22 \leq O(h^{3\log^{-1/2}h}) \cdot ALG$.

Thus,

$$OPT2 = OPT21 + OPT22 \leq O(h^{3\log^{-1/2}h}) \cdot ALG.$$

This lemma is true. □

Combining the above two lemmas, we have the next result.

Theorem 3. *The competitive ratio of the algorithm Pricing2 is* $O(h^{3\log^{-1/2}h})$.

References

1. Balcan, M.-F., Blum, A.: Approximation Algorithms and Online Mechanisms for Item Pricing. Thoery of Computing 3, 179–195 (2007)
2. Cheung, M., Swamy, C.: Approximation Algorithms for Single-minded Envy-free Profit-maximization Problems with Limited Supply. In: Proc. of 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008), pp. 35–44 (2008)
3. Balcan, N., Blum, A., Mansour, Y.: Item pricing for revenue maximization. In: Proc. of the 9th ACM Conference on Electronic Commerce (EC 2008), pp. 50–59 (2008)
4. Bansal, N., Chen, N., Cherniavsky, N., Rurda, A., Schieber, B., Sviridenko, M.: Dynamic pricing for impatient bidders. ACM Transactions on Algorithms 6(2) (March 2010)
5. Chen, N., Deng, X.: Envy-Free Pricing in Multi-item Markets. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 418–429. Springer, Heidelberg (2010)
6. Chen, N., Ghosh, A., Vassilvitskii, S.: Optimal envy-free pricing with metric substitutability. In: Proc. of the 9th ACM Conference on Electronic Commerce (EC 2008), pp. 60–69 (2008)
7. Elbassioni, K., Sitters, R., Zhang, Y.: A quasi-PTAS for profit-maximizing pricing on line graphs. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 451–462. Springer, Heidelberg (2007)
8. Fiat, A., Wingarten, A.: Envy, Multi Envy, and Revenue Maximization. In: Leonardi, S. (ed.) WINE 2009. LNCS, vol. 5929, pp. 498–504. Springer, Heidelberg (2009)
9. Grigoriev, A., van Loon, J., Sitters, R., Uetz, M.: How to Sell a Graph: Guidelines for Graph Retailers. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 125–136. Springer, Heidelberg (2006)
10. Guruswami, V., Hartline, J., Karlin, A., Kempe, D., Kenyon, C., McSherry, F.: On Profit-Maximizing Envy-Free Pricing. In: Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005), pp. 1164–1173 (2005)
11. Im, S., Lu, P., Wang, Y.: Envy-Free Pricing with General Supply Constraints. In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 483–491. Springer, Heidelberg (2010)