A Dynamic Near-Optimal Algorithm for Online Linear Programming

Shipra Agrawal
Department of Computer Science, Stanford University, Stanford, CA
email: shipra@cs.stanford.edu

Zizhuo Wang
Department of Management Science and Engineering, Stanford University, Stanford, CA
email: zzwang@stanford.edu

Yinyu Ye
Department of Management Science and Engineering, Stanford University, Stanford, CA
email: yinyu-ye@stanford.edu

A natural optimization model that formulates many online resource allocation and revenue management problems is the online linear program (LP) where the constraint matrix is revealed column by column along with the objective function. We provide a near-optimal algorithm for this surprisingly general class of online problems under the assumption of random order of arrival and some mild conditions on the size of the LP right-hand-side input. Our learning-based algorithm works by dynamically updating a threshold price vector at geometric time intervals, where the dual prices learned from revealed columns in the previous period are used to determine the sequential decisions in the current period. Our algorithm has a feature of "learning by doing", and the prices are updated at a carefully chosen pace that is neither too fast nor too slow. In particular, our algorithm doesn't assume any distribution information on the input itself, thus is robust to data uncertainty and variations due to its dynamic learning capability. Applications of our algorithm include many online multi-resource allocation and multi-product revenue management problems such as online routing and packing, online combinatorial auctions, adwords matching, inventory control and yield management.

Key words: online algorithms; linear programming; primal-dual; dynamic price update

MSC2000 Subject Classification: Primary: 68W27, 90B99; Secondary: 90B05, 90B50, 90C05

OR/MS subject classification: Primary: Linear Programming; analysis of algorithms

---

1. Introduction Online optimization is attracting an increasingly wide attention in computer science, operations research, and management science communities because of its wide applications to electronic markets and dynamic resource allocation problems. In many practical problems, data does not reveal itself at the beginning, but rather comes in an online fashion. For example, in the online revenue management problem, consumers arrive sequentially requesting a subset of goods (multi-leg flights or a period of stay in a hotel), each offering a certain bid price for his demand. On observing a request, the seller needs to make an irrevocable decision for that consumer with the overall objective of maximizing the revenue while respecting the resource constraints. Similarly, in the online routing problem [8, 3], the central organizer receives demands for subsets of edges in a network from the users in a sequential manner, each with a certain utility and bid price for his demand. The organizer needs to allocate the network capacity online to those bidders to maximize social welfare. A similar format also appears in online auctions [4, 10], online keyword matching problems [13, 20, 23, 16], online packing problems [9], and various other online revenue management and resource allocation problems [22, 11, 6].

In all these examples mentioned above, the problem can be formulated as an online linear programming problem[1]. In an online linear programming problem, the constraint matrix is revealed column by column with the corresponding coefficient in the objective function. After observing the input arrived so far, the online algorithm must make the current decision without observing the future data. To be precise, consider the linear program

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{n} \pi_j x_j & \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j \le b_i, & i = 1, \ldots, m \\ & 0 \le x_j \le 1 & j = 1, \ldots, n \end{array} \tag{1}$$

where $\forall j$, $\pi_j \ge 0$, $\boldsymbol{a}_j = (a_{ij})_{i=1}^m \in [0,1]^m$, and $\boldsymbol{b} = \{b_i\}_{i=1}^m \in \mathbb{R}^m$. In the corresponding online problem, at time t, the coefficients $(\pi_t, \boldsymbol{a}_t)$ are revealed, and the algorithm must make a decision x_t. Given the

[1]In fact, many of the problems mentioned above take the form of an integer program. While we discuss our ideas and results in terms of linear relaxation of these problems, our results naturally extends to integer programs. See Section 5.4 for the detailed discussion.

previous $t-1$ decisions $x_1, \ldots, x_{t-1}$, and input $\{\pi_j, \boldsymbol{a}_j\}_{j=1}^t$ until time t, the t^{th} decision is to select x_t such that

$$\begin{array}{l} \sum_{j=1}^t a_{ij} x_j \le b_i, \quad i = 1, \ldots, m \\ 0 \le x_t \le 1. \end{array} \tag{2}$$

The goal of the online algorithm is to choose x_t's such that the objective function $\sum_{t=1}^n \pi_t x_t$ is maximized.

To evaluate an online algorithm, one could consider various kinds of input models. One approach, which is completely robust to input uncertainty, is the worst-case analysis, that is, to evaluate the algorithm based on its performance on the worst-case input [23, 9]. However, this leads to very pessimistic bounds for the above online problem: no online algorithm can achieve better than $O(1/n)$ approximation of the optimal offline solution [4]. The other approach, popular among practitioners with domain knowledge, is to assume certain distribution on the inputs and consider the expected objective value achieved by the algorithm. In many cases, a priori input distribution can simplify the problem to a great extent, however, the choice of distribution is very critical and the performance can suffer if the actual input distribution is not as assumed. In this paper, we take an intermediate path. While we do not assume any knowledge of the input distribution, we relax the worst-case model by making the following two assumptions:

ASSUMPTION 1.1 *The columns $\boldsymbol{a}_j$ (with the objective coefficient π_j) arrive in a random order, i.e., the set of columns can be adversarily picked at the start. However, after they are chosen, $(\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n)$ and $(\boldsymbol{a}_{\sigma(1)}, \boldsymbol{a}_{\sigma(2)}, ..., \boldsymbol{a}_{\sigma(n)})$ have same chance to happen for all permutation σ.*

This assumption says that we consider the average behavior of the online algorithm over random permutations. This assumption is reasonable in practical problems, since the order of columns usually appears to be independent of the content of the columns. We like to emphasize that this assumption is strictly weaker than assuming an input distribution. In particular, it is automatically satisfied if the input columns are generated independently from some common (but unknown) distributions. This is also a standard assumption in many existing literature on solving online problems [4, 13, 20, 17].

ASSUMPTION 1.2 *We know the total number of columns n a priori.*

The second assumption is needed since we will use quantity n to decide the length of history used to learn the threshold prices in our algorithm. It can be relaxed to an approximate knowledge of n (within at most $1 \pm \epsilon$ multiplicative error), without affecting the final results. Note that this assumption is also standard in many existing algorithms for online problems [4] in computer science. For many practical problems, the knowledge of n can be implied approximately by the length of time horizon T and the arrival rates, which is usually available. As long as the time horizon is long enough, the total number of arrivals in the LP problem will be highly accurate. This is justified in [11] and [19] when they use the expected number of arrivals for constructing a pricing policy in a revenue management problem. Moreover, this assumption is necessary from the algorithmic point of view. In [13], the authors showed that if Assumption 2 doesn't hold, then the worst-case competitive ratio is bounded away from 1 even we admit Assumption 1.

In this paper, we present an almost optimal solution for online linear program (2) under the above two assumptions and a lower bound condition on the size of $\boldsymbol{b}$. We also extend our results to the following more general online linear optimization problems:

- Problems with multi-dimensional decisions at each time step. More precisely, consider a sequence of n non-negative vectors $\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_n \in \mathbb{R}^k$, mn non-negative vectors

  $$\boldsymbol{g}_{i1}, \boldsymbol{g}_{i2}, \ldots, \boldsymbol{g}_{in} \in [0,1]^k, \quad i = 1, \ldots, m,$$

  and $(k-1)$-dimensional simplex $K = \{\boldsymbol{x} \in \mathbb{R}^k : \boldsymbol{x}^T \boldsymbol{e} \le 1, \boldsymbol{x} \ge 0\}$. In this problem, given the previous $t-1$ decisions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$, each time we make a k-dimensional decision $\boldsymbol{x}_t \in \mathbb{R}^k$, satisfying:

  $$\begin{array}{c} \sum_{j=1}^t \boldsymbol{g}_{ij}^T \boldsymbol{x}_j \le b_i, \quad i = 1, \ldots, m \\ \boldsymbol{x}_t \in K \end{array} \tag{3}$$

  where decision vector $\boldsymbol{x}_t$ must be chosen only using the knowledge up to time t. The objective is to maximize $\sum_{j=1}^n \boldsymbol{f}_j^T \boldsymbol{x}_j$ over the whole time horizon. Note that Problem (2) is a special case of Problem (3) with $k = 1$.

- Problem (2) with both buy-and-sell orders, that is,

$$\pi_j \text{ either positive or negative, and } \boldsymbol{a}_j = (a_{ij})_{i=1}^m \in [-1,1]^m. \tag{4}$$

1.1 Our key ideas and main results In the following, let OPT denote the optimal objective value for the offline problem (1).

DEFINITION 1.1 *An online algorithm A is c-competitive in random permutation model if the expected value of the online solution obtained by using A is at least c factor of the optimal offline solution. That is,*

$$\mathbb{E}_\sigma[\textstyle\sum_{t=1}^n \pi_t x_t(\sigma, A)] \geq c OPT$$

where the expectation is taken over uniformly random permutations σ of $1,\ldots,n$, and $x_t(\sigma, A)$ is the t^{th} decision made by algorithm A when the inputs arrive in the order σ.

Our algorithm is based on the observation that the optimal solution $\boldsymbol{x}^*$ for the offline linear program is almost entirely determined by the optimal dual solution $\boldsymbol{p}^* \in \mathbb{R}^m$ corresponding to the m inequality constraints. The optimal dual solution acts as a *threshold price* so that $x_j^* > 0$ only if $\pi_j \geq \boldsymbol{p}^{*T}\boldsymbol{a}_j$. Our online algorithm works by learning a threshold price vector from the input received so far. The price vector then determines the decision for the next period. However, instead of computing a new price vector at every step, the algorithm initially waits until $n\epsilon$ steps or arrivals, and then computes a new price vector every time the history doubles. That is, at time steps $n\epsilon, 2n\epsilon, 4n\epsilon, \ldots$ and so on. We show that our algorithm is $1 - O(\epsilon)$-competitive in random permutation model under a size condition of the right-hand-side input. Our main results are precisely stated as follows:

THEOREM 1.1 *For any $\epsilon > 0$, our online algorithm is $1 - O(\epsilon)$ competitive for the online linear program (2) in random permutation model, for all inputs such that* [2]

$$B = \min_i b_i \geq \Omega\left(\frac{m \log(n/\epsilon)}{\epsilon^2}\right) \tag{5}$$

Note that the condition in Theorem 1.1 depends on $\log n$, which is far from satisfying everyone's demand when n is large. In [21], the author proves that $k \geq 1/\epsilon^2$ is necessary to get $1-O(\epsilon)$ competitive ratio in k-secretary problem, which is the single dimensional case of our problem with $m = 1, B = k$ and $a_t = 1$ for all t. Thus, dependence on ϵ in the above theorem is near-optimal. In the next theorem, we show that a dependence on m is necessary as well for any online algorithm to obtain a near-optimal solution. Its proof will appear in Section 4.

THEOREM 1.2 *For any online algorithm for linear program (2) in random permutation model, there exists an instance such that its competitive ratio is less than $1 - \Omega(\epsilon)$ when*

$$B = \min_i b_i \leq \frac{\log(m)}{\epsilon^2}$$

We also extend our results to more general online linear programs as introduced in (3) and (4):

THEOREM 1.3 *For any $\epsilon > 0$, our algorithm is $1-O(\epsilon)$ competitive for the general online linear problem (3) or (4) in random permutation model, for all inputs such that:*

$$B = \min_i b_i \geq \Omega\left(\frac{m \log(nk/\epsilon)}{\epsilon^2}\right). \tag{6}$$

REMARK 1.1 *Our condition to hold the main result is independent of the size of OPT or objective coefficients, and is also independent of any possible distribution of input data. If the largest entry of constraint coefficients does not equal to 1, then our both theorems hold if the condition (5) or (6) is replaced by:*

$$\frac{b_i}{\bar{a}_i} \geq \Omega\left(\frac{m \log(nk/\epsilon)}{\epsilon^2}\right), \ \forall i,$$

[2] For any two functions $f(n), g(n)$, $f(n) = O(g(n))$ iff there exists some constant c_1 such that $f(n) \leq c_1 g(n)$; and $f(n) = \Omega(g(n))$ iff there exists some constant c_2 such that $f(n) \geq c_2 g(n)$. The precise constants required here will be illustrated later in the text.

where, for each row i, $\bar{a}_i = \max_j\{|a_{ij}|\}$ of (2) and (4), or $\bar{a}_i = \max_j\{\|\boldsymbol{g}_{ij}\|_\infty\}$ of (3). Note that this bound is proportional only to $\log(n)$ so that it is way below to satisfy everyone's demand.

It is apparent that our generalized problem formulation should cover a wide range of online decision making problems. In the next section, we discuss the related work and some sample applications of our model. As one can see, our result in fact improves the best competitive ratios for many online problems studied in the literature and presents the first non-asymptotic analysis for solving many online resource allocation and revenue management problems.

1.2 Related work Online decision making has been a topic of wide recent interest in the computer science, operations research, and management science communities. Various special cases of the general problem presented in this paper have been studied extensively in the computer science literature as "secretary problems". A comprehensive survey of existing results on the secretary problems can be found in [4]. In particular, constant factor competitive ratios have been proven for k-secretary and knapsack secretary problems under random permutation model. Further, for many of these problems, a constant competitive ratio is known to be optimal if no additional conditions on input are assumed. Therefore, there has been recent interest in searching for online algorithms whose competitive ratio approaches 1 as the input parameters become large. The first result of this kind appears in [21], where a $1 - O(1/\sqrt{k})$-competitive algorithm is presented for k secretary problem under random permutation model. Recently, the authors in [13, 15] presented a $1 - O(\epsilon)$-competitive algorithm for online adwords matching problem under assumptions of certain lower bounds on OPT and $k = \min_i b_i$ in terms of ϵ and other input parameters. Our work is closely related to the works of Kleinberg [21], Devanur et al. [13], and Feldman et al. [15].

In [21], the author presented a recursive algorithm for single dimensional multiple-choice secretary problem, and proved that $1-O(1/\sqrt{k})$ competitive ratio achieved by his algorithm is the best possible for this problem. We present a $(1-O(\sqrt{m\log(n)/k}))$ competitive algorithm for multi-dimensional multiple-choice secretary problem ($k = \min_i b_i$), which involves recursive pricing similar to [21], however, due to the multi-dimensional structure of the problem, significantly different techniques are required for analysis than those used in [21]. We also prove that no online algorithm can achieve a competitive ratio better than $(1 - \Omega(\sqrt{\log(m)/k}))$ for the multi-dimensional problem. To our knowledge, this is the first result that shows a dependence on dimension m, of the best competitive ratio achievable for this problem.

In [13], the authors study a one-time pricing approach (as opposed to the recursive or dynamic pricing approach) for a special case (adwords matching) of our problem. Also, the authors prove a competitive ratio that depends on the optimal value OPT, rather than the right hand side $k = \min_i b_i$. To be precise, they prove a competitive ratio of $(1 - O(\sqrt[3]{m^2\log(n)/\mathrm{OPT}}))$. Due to the use of a dynamic pricing approach, we show that even in terms of OPT, our algorithm has a strictly better competitive ratio of $(1 - O(\sqrt{m^2\log(n)/\mathrm{OPT}}))$ (see Section 5.1 and Appendix F). We would like to emphasize here that (a) a competitive ratio in terms of OPT does not imply a corresponding competitive ratio in terms of k and vice-versa, as we illustrate by examples in Appendix F, there are cases where one may be better than the other, and (b) significantly different techniques are used in this paper to achieve a bound that depends only on k, and not on the value of OPT, in particular refer to the analysis in Lemma 3.2 and Lemma 3.3, and its implications on the competitive ratio analysis. We believe having a dependence on k rather than OPT may be more attractive in many practical settings, since the value of k is known and can be checked.

More recently, and subsequent to the submission of an earlier version of our paper [2], Feldman et al. [15] (independently) obtained bounds for an online packing problem which depend both on the right hand side k and OPT. In addition to requiring lower bounds on both k and OPT, their lower bound on k depends on ϵ^3, thus their result is much weaker than the result in this paper. Their algorithm is based on a one-time pricing approach.

Table 1 provides a comparison of these related results, clearly illustrating that our dynamic pricing algorithm provides significant improvement over the past results. Here "Condition" refers to the lower bound condition required on the input to achieve $(1 - O(\epsilon))$ competitive ratio, and $k = \min_i b_i$.

In the operations research and management science community, a dynamic and optimal pricing strategy for various online resource allocation problems has always been an important research topic, some

	Condition	Technique	Target problem
Kleinberg et al., 2005 [21]	$k \geq \frac{1}{\epsilon^2}$, for $m = 1$	Dynamic pricing	k-secretary problem
Devanur et al., 2009 [13]	$\frac{\text{OPT}}{\pi_{max}} \geq \frac{m^2 \log(n)}{\epsilon^3}$	One-time pricing	Adwords problem
Feldman et al., 2010 [15]	$k \geq \frac{m \log(n)}{\epsilon^3}$ **and** $\frac{\text{OPT}}{\pi_{max}} \geq \frac{m \log(n)}{\epsilon}$	One-time pricing	online packing
This paper	$k \geq \frac{m \log(n)}{\epsilon^2}$ **or** $\frac{\text{OPT}}{\pi_{max}} \geq \frac{m^2 \log(n)}{\epsilon^2}$	Dynamic pricing	general online LP

Table 1: Comparison of some existing results

literatures include [14, 18, 19, 26, 22, 11, 6]. In [19, 18, 6], the arrival process are assumed to be price sensitive. However, as commented in [11], this model can be reduced to a price independent arrival process with availability control under Poisson arrivals. Our model can be further viewed as a discrete version of the availability control model which is also used as an underlying model in [26] and discussed in [11]. The idea of using a threshold - or "bid" - price is not new. It is initiated in [28, 25] and investigated further in [26]. In [26], the authors show that the bid price is asymptotically optimal. However, they assume the knowledge on the arrival process and therefore the price is obtained by "forecasting" the future using the distribution information rather than "learning" from the past observations as we do in our paper. The idea of using linear programming to find dual optimal bid price is discussed in [11] where asymptotic optimality is also achieved. But again, the arrival process is assumed to be known which made their analysis relatively simple.

Our work improves upon these existing works in various manners. First, we provide a common near-optimal solution for a wide class of online linear programs which encompasses many special cases of secretary problems and resource allocation problems discussed above. Moreover, due to its dynamic learning capability, our algorithm is *distribution free*–no knowledge on the input distribution is assumed except for the random order of arrival. The techniques proposed in this paper may also be considered a step forward in the threshold price learning kind of approaches. A common element in the techniques used in existing works on secretary problems [4] (with the exception of [21]), online combinatorial auction problems [1], and adwords matching problem [13, 15], has been *one-time learning* of threshold price(s) from first few ($n\epsilon$) customers, which is then used to determine the decision for remaining customers. However, in practice one would expect some benefit from dynamically updating the prices as more and more information is revealed. Dynamic pricing has been a topic of wide attention in many of the management science literature [14], and a question of increasing importance is: how often and when to update the price? In [11], the authors demonstrate with a specific example that updating the price too frequently may even hurt the results. In this paper, we propose a dynamic pricing algorithm that updates the prices at geometric time intervals–not too sparse nor too often. In particular, we present an improvement from a factor of $1/\epsilon^3$ to $1/\epsilon^2$ in the lower bound requirement on B by using dynamic price updating instead of one-time learning. Thus we present, for the first time, a precisely quantified strategy for dynamic price update.

In our analysis, we apply many standard techniques from PAC-learning[3], in particular, concentration bounds and covering arguments. These techniques were also heavily used in [5] and [13]. In [5], price learned from one half of bidders is used for the other half to get an incentive compatible mechanism for combinatorial auctions. Their approach is closely related to the idea of *one-time learning* of price in online auctions, however, their goal is *offline revenue maximization* and an *unlimited supply* is assumed. And [13], as discussed above, considers a special case of our problem. Part of our analysis is inspired by some ideas used there, as will be pointed out in the text.

Further comparison of our work with the related work on specific applications appears in the next section.

1.3 Specific Applications In the following, we show some of the specific applications of our algorithm and compare our work to the previous results for those applications. It is worth noting that

[3]Probably Approximately Correct learning: which is a framework for mathematical analysis of machine learning

for many of the problems we discuss below, our algorithm is the first near-optimal algorithm under the distribution-free model.

1.3.1 Online routing problems The most direct application of our online algorithm is the online routing problem. In this problem, there are m edges in a network, each edge i has a bounded capacity b_i. There are n customers arriving online, each with a request of certain path $\boldsymbol{a}_t \in \{0,1\}^m$, where $a_{it} = 1$, if the path of request t contains edge i, and a utility π_t for his request. The offline problem for the decision maker is given by the following integer program:

$$\begin{array}{ll} \text{maximize} & \sum_{t=1}^{n} \pi_t x_t \\ \text{subject to} & \sum_{t=1}^{n} a_{it} x_t \le b_i \quad i = 1, \ldots, m \\ & x_t \in \{0,1\} \end{array} \tag{7}$$

By Theorem 1.1, and its natural extension to integer programs as will be discussed in Section 5.4, our algorithm gives a $1-O(\epsilon)$ competitive solution to this problem in the random permutation model as long as the edge capacity is reasonably large. Earlier, a best of $\log(m\frac{\pi_{max}}{\pi_{min}})$ competitive algorithm was known for this problem under the worst case input model [8].

1.3.2 Online single-minded combinatorial auctions In this problem, there are m goods, b_i units of each good i are available. There are n bidders arriving online, each with a bundle of items $\boldsymbol{a}_t \in \{0,1\}^m$ that he desires to buy, and a limit price π_t for his bundle. The offline problem of maximizing the social utility is the same as the routing problem formulation given in (7). Due to the use of a threshold price mechanism, where threshold price for t^{th} bidder is computed from the input of previous bidders, it is easy to show that our $1-O(\epsilon)$ competitive online mechanism also supports incentive compatibility and voluntary participation. Also one can easily transform this model to revenue maximization. A $\log(m)$-competitive algorithm for this problem in random permutation setting can be found in recent work [1].

1.3.3 The online adwords problems The online adwords allocation problem is essentially the online matching problem. In this problem, there are n queries arriving online. And, there are m bidders each with a daily budget b_i, and bid π_{ij} on query j. For j^{th} query, the decision vector $\boldsymbol{x}_j$ is an m-dimensional vector, where $x_{ij} \in \{0,1\}$ indicates whether the j^{th} query is allocated to the i^{th} bidder. Also, since every query can be allocated to at most one bidder, we have the constraint $\boldsymbol{x}_j^T \boldsymbol{e} \le 1$. Therefore, the corresponding offline problem can be stated as:

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{n} \boldsymbol{\pi}_j^T \boldsymbol{x}_j \\ \text{subject to} & \sum_{j=1}^{n} \pi_{ij} x_{ij} \le b_i, \quad i = 1, \ldots, m \\ & \boldsymbol{x}_j^T \boldsymbol{e} \le 1 \\ & \boldsymbol{x}_j \in \{0,1\}^m \end{array}$$

The linear relaxation of the above problem is a special case of the general linear optimization problem (3) with $\boldsymbol{f}_j = \boldsymbol{\pi}_j$, $\boldsymbol{g}_{ij} = \pi_{ij}\boldsymbol{e}_i$ where $\boldsymbol{e}_i$ is the ith unit vector of all zeros except 1 for the ith entry. By Theorem 1.3 (and remarks below the theorem), and extension to integer programs discussed in Section 5.4, our algorithm gives a $1-O(\epsilon)$ approximation for this problem given the lower bound condition that for all i, $\frac{b_i}{\pi_i^{max}} \ge \frac{m\log(mn/\epsilon)}{\epsilon^2}$, where $\pi_i^{max} = \max_j\{\pi_{ij}\}$ is the largest bid by bidder i among all queries. Richer models incorporating other aspects of sponsored search such as multiple slots, can be formulated by redefining $\boldsymbol{f}_j, \boldsymbol{g}_{ij}, K$ to obtain similar results.

There were several previous studies on this problem under the worst case input model [23, 7, 16], for which an $1-1/e$ competitive algorithm exists, and this factor is tight. Under the random permutation assumption, Goel et al [20] showed that the greedy algorithm yields a $1-1/e$ approximation. Later Feldman et al [17] improved this ratio to 0.67.

A $1-O(\epsilon)$ algorithm was provided for this problem by [13] with certain lower bound conditions on OPT. As we pointed out earlier, we can guarantee the same ratio with a weaker (see Table 1) condition on OPT than that obtained by [13]. Later, we show that this improvement is a result of dynamically learning the price at geometric intervals, instead of one-time learning in [13]. [15] also presents a near-optimal algorithm for this problem concurrent (and independent) with our work. However, their results are also much weaker than ours, the comparison can be found in Table 1.

1.3.4 Yield management problems Online yield management problem is to allocate perishable resources to demands in order to increase the revenue by best online matching the resource capacity and demand in a given time horizon T. It has wide applications including airline booking, hotel reservation, media and the internet resource allocation problems. In these problems, there are several types of product j, $j = 1, 2, ..., J$, and several resources b_i, $i = 1, ..., m$. To sell a unit demand of product j, it requires to consume certain units r_{ij} of resource i, for all i. Buyers, each demanding certain type of product, come in a stationary Poisson process and each offers a price π for his or her unit product demand. The objective of the seller is to maximize his or her revenue in the time horizon T while respecting given resource constraints. The offline problem can be written as follows:

$$\begin{array}{ll} \text{maximize} & \sum_t \pi_t x_t \\ \text{subject to} & \sum_t \boldsymbol{a}_t x_t \le \boldsymbol{b} \\ & x_t \in \{0,1\}, \quad \forall t, \end{array} \tag{8}$$

where $\boldsymbol{a}_t$ is a type of product demanded by t^{th} buyer, i.e. $a_{it} = r_{ij}$, if the t^{th} buyer demands product of type j, $j = 1, ..., J$. In practice, we may not know the exact number n of the total buyers in advance. However, by discretizing the time horizon T to sufficiently small time periods such that there is at most one arrival in each time period and setting $a_t = \mathbf{0}$ for period t if there is no arrival in that period, the above model well approximates the arrival process under the assumption of stationary Poisson arrivals [26]. Furthermore, the arrivals can be viewed as randomly ordered. Therefore, our online linear programming model encompasses the above revenue management problem and our algorithm will provide a near-optimal solution to this problem without any knowledge on the distribution of the demands. In this case, our dynamic price updating algorithm will update the threshold prices at time periods $\epsilon T, 2\epsilon T, 4\epsilon T, \ldots$ till T.

1.3.5 Inventory control problems with replenishment This problem is similar to the yield management problem discussed in the previous subsection but has multiple time period. The sellers have m items to sell. At each time step j, a bidder comes and requests a certain bundle of items $\boldsymbol{a}_j$ and offers a price π_j. In each period, the seller has to choose an inventory $\boldsymbol{b}$ at the beginning, and then allocate the demand of the buyers during this period. Each unit of b_i costs capital c_i and the total investment $\sum_i b_i c_i$ in each period is limited by budget C. There are T periods and the objective is to maximize the total revenue in these periods. The offline problem of each period, for all bidders who arrive in that period, is as follows:

$$\begin{array}{lll} \text{maximize}_{\boldsymbol{x},\boldsymbol{b}} & \sum_t \pi_t x_t & \\ \text{subject to} & \sum_t \boldsymbol{a}_t x_t - \boldsymbol{b} \le \mathbf{0} & \\ & \sum_{i=1}^m c_i b_i \le C & \\ & 0 \le x_t \le 1, & \forall t. \end{array} \tag{9}$$

Note that given $\boldsymbol{b}$, the problem for one period is exactly as we discussed before. Given the bids come in a random permutation over the total time horizon, our analysis will show that the itemized demands $\boldsymbol{b}$ learned from the previous period, together with its itemized dual prices, will be approximately the same for the remaining periods. Thus, installing inventory $\boldsymbol{b}$ and online pricing the bids for each of the remaining periods based on the learned demand and dual prices would give a revenue that is close to the optimal revenue of the offline problem over the entire time horizon:

$$\begin{array}{lll} \text{maximize}_{\boldsymbol{x},\boldsymbol{b}} & \sum_j \pi_j x_j & \\ \text{subject to} & \sum_j \boldsymbol{a}_j x_j - \boldsymbol{b} \le \mathbf{0} & \\ & \sum_{i=1}^m c_i b_i \le T \cdot C & \\ & 0 \le x_j \le 1, & \forall j. \end{array} \tag{10}$$

1.4 Organization The rest of the paper is organized as follows. In Section 2 and 3, we present our online algorithm and prove that it achieves $1 - O(\epsilon)$ competitive ratio under mild conditions on the input. To keep the discussion clear and easy to follow, we start in Section 2 with a simpler one-time learning algorithm. While the analysis for this simpler algorithm will be useful to demonstrate our proof techniques, the results obtained in this setting are weaker than those obtained by our dynamic price update algorithm, which is discussed in Section 3. In Section 4, we give a detailed proof of Theorem 1.2 regarding the necessity of lower bound condition used in our main theorem. In Section 5, we present many extensions, including an alternative bound in terms of optimal offline value OPT instead of B, the extension to multi-dimensional online linear programs, linear program with negative coefficients, and

integer programs, and the applicability of our model to solving large static linear programs. Then we conclude our paper in Section 6.

2. One-time learning algorithm For the linear program (1), we use $\boldsymbol{p} \in \mathbb{R}^m$ to denote the dual variable associated with the first set of constraints $\sum_t \boldsymbol{a}_t x_t \le \boldsymbol{b}$. Let $\hat{\boldsymbol{p}}$ denote the optimal dual solution to the following partial linear program defined only on the input until time $s = \lceil n\epsilon \rceil$:

$$\begin{array}{lll} \text{maximize} & \sum_{t=1}^{s} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{s} a_{it} x_t \le (1-\epsilon)\frac{s}{n} b_i, & i = 1, \ldots, m \\ & 0 \le x_t \le 1, & t = 1, \ldots, s \end{array} \tag{11}$$

Also, for any given dual price vector $\boldsymbol{p}$, define the allocation rule $x_t(\boldsymbol{p})$ as:

$$x_t(\boldsymbol{p}) = \begin{cases} 0 & \text{if } \pi_t \le \boldsymbol{p}^T \boldsymbol{a}_t \\ 1 & \text{if } \pi_t > \boldsymbol{p}^T \boldsymbol{a}_t \end{cases} \tag{12}$$

Our one-time learning algorithm can now be stated as follows:

Algorithm 1 One-time Learning Algorithm (OLA)

(i) Initialize $s = \lceil n\epsilon \rceil$, $x_t = 0$, for all $t \le s$. And $\hat{\boldsymbol{p}}$ is defined as above.

(ii) Repeat for $t = s+1, s+2, \ldots$

- If $a_{it} x_t(\hat{\boldsymbol{p}}) \le b_i - \sum_{j=1}^{t-1} a_{ij} x_j$, set $x_t = x_t(\hat{\boldsymbol{p}})$; otherwise, set $x_t = 0$. Output x_t.

This algorithm learns a dual price vector using the first $\lceil n\epsilon \rceil$ arrivals. Then, at each time $t > \lceil n\epsilon \rceil$, it uses this price vector to decide the current allocation, and executes this decision as long as it doesn't violate any of the constraints. An attractive feature of this algorithm is that it requires to solve only one small linear program, defined on $\lceil n\epsilon \rceil$ variables. In the next subsection, we prove the following proposition regarding the competitive ratio of this algorithm, which relies on a stronger condition than Theorem 1.1:

PROPOSITION 2.1 *For any $\epsilon > 0$, the one-time learning algorithm is $1 - 6\epsilon$ competitive for the online linear program (2) in random permutation model, for all inputs such that*

$$B = \min_i b_i \ge \frac{6m \log(n/\epsilon)}{\epsilon^3} \tag{13}$$

2.1 Competitive ratio analysis Observe that the one-time learning algorithm waits until time $s = \lceil n\epsilon \rceil$, and then sets the solution at time t as $x_t(\hat{\boldsymbol{p}})$, unless there is a constraint violation. To prove the competitive ratio of this algorithm, we first prove that with high probability, $x_t(\hat{\boldsymbol{p}})$ satisfies all the constraints of the linear program. Then, we show that the expected value $\sum_t \pi_t x_t(\hat{\boldsymbol{p}})$ is close to the optimal offline objective value. For simplicity of the discussion, we assume $s = n\epsilon$ in the following.

To start with, we observe that if $\boldsymbol{p}^*$ is the optimal dual solution to (1), then $\{x_t(\boldsymbol{p}^*)\}$ is close to the primal optimal solution $\boldsymbol{x}^*$, i.e., learning the dual price is sufficient to determine the primal solution. We make the following simplifying technical assumption:

ASSUMPTION 2.1 *The inputs of this problem are in general position, namely for any price vector $\boldsymbol{p}$, there can be at most m columns such that $\boldsymbol{p}^T \boldsymbol{a}_t = \pi_t$.*

The assumption is not necessarily true for all inputs. However, one can always randomly perturb π_t by arbitrarily small amount η through adding a random variable ξ_t taking uniform distribution on interval $[0, \eta]$. In this way, with probability 1, no $\boldsymbol{p}$ can satisfy $m+1$ equations simultaneously among $\boldsymbol{p}^T \boldsymbol{a}_t = \pi_t$, and the effect of this perturbation on the objective can be made arbitrarily small [4]. Given this assumption, we can use complementary conditions of linear program (1) to observe that:

LEMMA 2.1 *$x_t(\boldsymbol{p}^*) \le x_t^*$, and under Assumption 2.1, x_t^* and $x_t(\boldsymbol{p}^*)$ differ on at most m values of t.*

[4] This technique for resolving ties was also used in [13].

The lemma basically tells that, if the optimal dual solution $\boldsymbol{p}^*$ to (1) is known, then the (integer) solution $x_t(\boldsymbol{p}^*)$ used by the online pricing algorithm is close to the optimal offline solution. However, in the online algorithm, we use the price $\hat{\boldsymbol{p}}$ learned from first few inputs, instead of the optimal dual price. The remaining discussion attempts to show that the learned price will be sufficiently accurate for our purpose. Note that the random order assumption can be interpreted to mean that the first s inputs are a uniform random sample without replacement of size s from the n inputs. Let S denote this sample set of size s, and N denote the complete set of size n. Consider the sample linear program (11) defined on the sample set S with right hand side set as $(1-\epsilon)\epsilon\boldsymbol{b}$. Then, $\hat{\boldsymbol{p}}$ was constructed as the optimal dual price of the sample linear program, which we refer to as the sample dual price. In the following two lemmas, we show that with high probability, the primal solution $x_t(\hat{\boldsymbol{p}})$ constructed from this sample dual price is feasible and near-optimal:

LEMMA 2.2 *The primal solution constructed using sample dual price is a feasible solution to the linear program*(1) *with high probability. More precisely, with probability* $1-\epsilon$*,*

$$\sum_{t=1}^{n} a_{it} x_t(\hat{\boldsymbol{p}}) \le b_i, \quad \forall i = 1, \ldots, m$$

given $B \ge \frac{6m\log(n/\epsilon)}{\epsilon^3}$.

PROOF. The proof will proceed as follows: Consider any fixed price $\boldsymbol{p}$. We say a random sample S is "bad" for this $\boldsymbol{p}$ if and only if $\boldsymbol{p} = \hat{\boldsymbol{p}}(S)$, but $\sum_{t=1}^{n} a_{it} x_t(\boldsymbol{p}) > b_i$ for some i. First, we show that the probability of bad samples is small for every fixed $\boldsymbol{p}$. Then, we take union bound over all "distinct" prices to prove that with high probability the learned price $\hat{\boldsymbol{p}}$ will be such that $\sum_{t=1}^{n} a_{it} x_t(\hat{\boldsymbol{p}}) \le b_i$ for all i.

To start with, we fix $\boldsymbol{p}$ and i. Define $Y_t = a_{it} x_t(\boldsymbol{p})$. If $\boldsymbol{p}$ is an optimal dual solution for sample linear program on S, then by the complementary conditions, we have:

$$\textstyle\sum_{t\in S} Y_t = \sum_{t\in S} a_{it} x_t(\boldsymbol{p}) \le (1-\epsilon)\epsilon b_i \tag{14}$$

Therefore, the probability of bad samples is bounded by:

$$\begin{aligned} P(\textstyle\sum_{t\in S} Y_t \le (1-\epsilon)\epsilon b_i, \sum_{t\in N} Y_t \ge b_i) &\le P(|\textstyle\sum_{t\in S} Y_t - \epsilon \sum_{t\in N} Y_t| \ge \epsilon^2 b_i | \textstyle\sum_{t\in N} Y_t = b_i) \\ &\le 2\exp\left(\tfrac{-\epsilon^3 b_i}{2+\epsilon}\right) \le \delta \end{aligned} \tag{15}$$

where $\delta = \frac{\epsilon}{m\cdot n^m}$. The last step follows from Hoeffding-Bernstein's Inequality (Lemma A.1 in appendix A), and the condition made on B.

Next, we take a union bound over all "distinct" $\boldsymbol{p}$'s. We call two price vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ distinct if and only if they result in distinct solutions, i.e., $\{x_t(\boldsymbol{p})\} \ne \{x_t(\boldsymbol{q})\}$. Note that we only need to consider distinct prices, since otherwise all the Y_t's are exactly the same. Thus, each distinct $\boldsymbol{p}$ is characterized by a unique separation of n points $(\{\pi_t, \boldsymbol{a}_t\}_{t=1}^n)$ in m-dimensional space by a hyperplane. By results from computational geometry [24], the total number of such distinct prices is at most n^m. Taking union bound over n^m distinct prices, and $i = 1, \ldots, m$, we get the desired result. □

Above we showed that with high probability, $x_t(\hat{\boldsymbol{p}})$ is a feasible solution. In the following, we show that it actually is a near-optimal solution.

LEMMA 2.3 *The primal solution constructed using sample dual price is a near-optimal solution to the linear program* (1) *with high probability. More precisely, with probability* $1-\epsilon$*,*

$$\sum_{t\in N} \pi_t x_t(\hat{\boldsymbol{p}}) \ge (1-3\epsilon)\,OPT \tag{16}$$

given $B \ge \frac{6m\log(n/\epsilon)}{\epsilon^3}$.

PROOF. The proof of this lemma is based on two observations. First, $\{x_t(\hat{\boldsymbol{p}})\}_{t=1}^n$ and $\hat{\boldsymbol{p}}$ satisfies all the complementarity conditions, and hence is an optimal primal-dual solution of the following linear program

$$\begin{array}{lll} \text{maximize} & \sum_{t\in N} \pi_t x_t & \\ \text{subject to} & \sum_{t\in N} a_{it} x_t \le \hat{b}_i, & i = 1, \ldots, m \\ & 0 \le x_t \le 1, & t = 1, \ldots, n \end{array} \tag{17}$$

where $\hat{b}_i = \sum_{t\in N} a_{it}x_t(\hat{\boldsymbol{p}})$ if $\hat{p}_i > 0$, and $\hat{b}_i = \max\{\sum_{t\in N} a_{it}x_t(\hat{\boldsymbol{p}}), b_i\}$, if $\hat{p}_i = 0$.

Second, one can show that if $\hat{p}_i > 0$, then with probability $1-\epsilon$, $\hat{b}_i \geq (1-3\epsilon)b_i$. To see this, observe that $\hat{\boldsymbol{p}}$ is optimal dual solution of sample linear program on set S, let $\hat{x}$ be the optimal primal solution. Then, by complementarity conditions of the linear program, if $\hat{p}_i > 0$, the i^{th} constraint must be satisfied by equality. That is, $\sum_{t\in S} a_{it}\hat{x}_t = (1-\epsilon)\epsilon b_i$. Then, given the observation made in Lemma 2.1, and that $B = \min_i b_i \geq \frac{m}{\epsilon^2}$, we get:

$$\sum_{t\in S} a_{it}x_t(\hat{\boldsymbol{p}}) \geq \sum_{t\in S} a_{it}\hat{x}_t - m \geq (1-2\epsilon)\epsilon b_i. \tag{18}$$

Then, using the Hoeffding-Bernstein's Inequality, in a manner similar to the proof of Lemma 2.2, we can show that (the proof is given in Appendix A.3.) given the lower bound on B, with probability at least $1-\epsilon$:

$$\hat{b}_i = \sum_{t\in N} a_{it}x_t(\hat{\boldsymbol{p}}) \geq (1-3\epsilon)b_i \tag{19}$$

Lastly, observing that whenever (19) holds, given an optimal solution x^* to (1), $(1-3\epsilon)x^*$ will be a feasible solution to (17). Therefore, the optimal value of (17) is at least $(1-3\epsilon)$OPT, which is equivalently saying that

$$\sum_{t=1}^{n} \pi_t x_t(\hat{\boldsymbol{p}}) \geq (1-3\epsilon)\text{OPT}$$

□

Therefore, the objective value for online solution taken over entire period $\{1,\ldots,n\}$ is near optimal. However, the online algorithm does not make any decision in the learning period S, and only the decisions from period $\{s+1,\ldots,n\}$ contribute to the objective value. The following lemma that relates sample optimal to the optimal value of a linear program will bound the contribution from the learning period:

LEMMA 2.4 *Let OPT(S) denote the optimal value of the linear program* (11) *over sample S, and OPT(N) denote the optimal value of the offline linear program* (1) *over N. Then,*

$$\mathbb{E}[OPT(S)] \leq \epsilon OPT(N)$$

PROOF. Let $(\boldsymbol{x}^*, \boldsymbol{p}^*, \boldsymbol{y}^*)$ and $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{p}}, \hat{\boldsymbol{y}})$ denote the optimal primal-dual solution of linear program (1) on N, and sample linear program on S, respectively.

$$(\boldsymbol{p}^*, \boldsymbol{y}^*) = \begin{array}{ll} \arg\min & \boldsymbol{b}^T\boldsymbol{p} + \sum_{t\in N} y_t \\ \text{s.t.} & \boldsymbol{p}^T\boldsymbol{a}_t + y_t \geq \pi_t \quad t \in N \\ & \boldsymbol{p}, \boldsymbol{y} \geq 0 \end{array} \qquad (\hat{\boldsymbol{p}}, \hat{\boldsymbol{y}}) = \begin{array}{ll} \arg\min & \epsilon\boldsymbol{b}^T\boldsymbol{p} + \sum_{t\in S} y_t \\ \text{s.t.} & \boldsymbol{p}^T\boldsymbol{a}_t + y_t \geq \pi_t, \ t \in S \\ & \boldsymbol{p}, \boldsymbol{y} \geq 0 \end{array}$$

Since $S \subseteq N$, $\boldsymbol{p}^*, \boldsymbol{y}^*$ is a feasible solution to the dual of linear program on S, by weak duality theorem:

$$\text{OPT}(S) \leq \epsilon\boldsymbol{b}^T\boldsymbol{p}^* + \sum_{t\in S} y_t^*$$

Therefore,

$$\mathbb{E}[\text{OPT}(S)] \leq \epsilon\boldsymbol{b}^T\boldsymbol{p}^* + \mathbb{E}[\sum_{t\in S} y_t^*] = \epsilon(\boldsymbol{b}^T\boldsymbol{p}^* + \sum_{t\in N} y_t^*) = \epsilon\text{OPT}(N)$$

□

Now, we are ready to prove Proposition 2.1:

Proof of Proposition 2.1: Using Lemma 2.2 and Lemma 2.3, with probability at least $1-2\epsilon$, the following event happen:

$$\sum_{t=1}^{n} a_{it}x_t(\hat{\boldsymbol{p}}) \leq b_i, \quad i = 1,\ldots,m$$

$$\sum_{t=1}^{n} \pi_t x_t(\hat{\boldsymbol{p}}) \geq (1-3\epsilon)OPT$$

That is, the decisions $x_t(\hat{\boldsymbol{p}})$ are feasible and the objective value taken over the entire period $\{1,\ldots,n\}$ is near optimal. Denote this event by $\mathcal{E}$, where $\Pr(\mathcal{E}) \geq 1-2\epsilon$. We have by Lemma 2.2, 2.3 and 2.4:

$$\begin{aligned} \mathbb{E}[\textstyle\sum_{t\in N\setminus S} \pi_t x_t(\hat{\boldsymbol{p}}) \mid \mathcal{E}] &\geq (1-3\epsilon)\mathrm{OPT} - \mathbb{E}[\textstyle\sum_{t\in S} \pi_t x_t(\hat{\boldsymbol{p}}) \mid \mathcal{E}] \\ &\geq (1-3\epsilon)\mathrm{OPT} - \tfrac{1}{1-2\epsilon}\mathbb{E}[\textstyle\sum_{t\in S} \pi_t x_t(\hat{\boldsymbol{p}})] \\ &\geq (1-3\epsilon)\mathrm{OPT} - \tfrac{\epsilon \mathrm{OPT}}{1-2\epsilon} \end{aligned} \tag{20}$$

Therefore,

$$\mathbb{E}[\sum_{t=s+1}^{n} \pi_t x_t(\hat{\boldsymbol{p}})] \geq \mathbb{E}[\sum_{t\in N\setminus S} \pi_t x_t(\hat{\boldsymbol{p}}) \mid \mathcal{E}] \cdot Pr(\mathcal{E}) \geq (1-6\epsilon)\mathrm{OPT}$$

□

3. Dynamic price update algorithm The algorithm discussed in the previous section uses the first $n\epsilon$ inputs to learn the price, and then applies it in the remaining time horizon. While this one-time learning algorithm has its own merits, in particular, requires solving only a small linear problem defined on $n\epsilon$ variables, the lower bound required on B is stronger than that claimed in Theorem 1.1 by an ϵ factor.

In this section, we discuss an improved dynamic price update algorithm that will achieve the result in Theorem 1.1. Instead of computing the price only once, the algorithm will update the price every time the history doubles, that is, it will learn a new price at time $t = n\epsilon, 2n\epsilon, 4n\epsilon, \ldots$. To be precise, let $\hat{\boldsymbol{p}}^{\ell}$ denote the optimal dual solution for the following partial linear program defined only on the input until time ℓ:

$$\begin{array}{lll} \text{maximize} & \sum_{t=1}^{\ell} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \leq (1-h_\ell)\frac{\ell}{n} b_i, & i = 1,\ldots,m \\ & 0 \leq x_t \leq 1, & t = 1,\ldots,\ell \end{array} \tag{21}$$

where the set of numbers h_ℓ are defined as follows:

$$h_\ell = \epsilon\sqrt{\tfrac{n}{\ell}} \quad \forall \ell \tag{22}$$

Also, for any given dual price vector $\boldsymbol{p}$, define the allocation rule $x_t(\boldsymbol{p})$ as earlier in (12). Then, our dynamic price update algorithm can be stated as follows:

Algorithm 2 Dynamic Pricing Algorithm (DPA)

(i) Initialize $t_0 = \lceil n\epsilon \rceil$, $x_t = \hat{x}_t = 0$, for all $t \leq t_0$.

(ii) Repeat for $t = t_0+1, t_0+2, \ldots$

(a) Set $\hat{x}_t = x_t(\hat{\boldsymbol{p}}^{\ell})$, where $\ell = \lceil 2^r n\epsilon \rceil$ for largest integer r such that $\ell < t$.

(b) If $a_{it}\hat{x}_t \leq b_i - \sum_{j=1}^{t-1} a_{ij}x_j$, set $x_t = \hat{x}_t$; otherwise, set $x_t = 0$. Output x_t.

Note that we update the price $\lceil \log_2(1/\epsilon) \rceil$ times during the whole time horizon. Thus, the algorithm requires more computation, but as we show next it requires a weaker lower bound on B for proving the same competitive ratio. The intuition behind this improvement is as follows. Note that initially, at $\ell = n\epsilon$, $h_\ell = \sqrt{\epsilon} > \epsilon$. Thus, more slack is available, and so the large deviation argument for constraint satisfaction (as in Lemma 2.2) requires a weaker condition on B. The numbers h_ℓ decreases as ℓ increases. However, for large values of ℓ, sample size is larger, making the weaker condition on B sufficient for our purpose. Also, h_ℓ decreases rapidly enough, so that the overall loss on the objective value is not significant. The careful choice of numbers h_ℓ will play a key role in proving our results.

3.1 Competitive ratio analysis The analysis for the dynamic algorithm proceeds in a manner similar to that for the one-time learning algorithm. However, stronger results for the price learned in each period need to be proven here. In this proof, for simplicity of discussion, we assume that $\epsilon = 2^{-E}$ for some integer E, and that the numbers $\ell = 2^r n\epsilon$ for $r = 0, 1, 2, ..., E-1$ are all integers. Let $L = \{n\epsilon, 2n\epsilon, \ldots, 2^{E-1}\epsilon\}$.

Lemma 3.1 and 3.2 are in parallel to Lemma 2.2 and 2.3, in the previous section, however require a weaker condition on B:

LEMMA 3.1 *For any $\epsilon > 0$, with probability $1-\epsilon$:*

$$\sum_{t=\ell+1}^{2\ell} a_{it} x_t(\hat{\boldsymbol{p}}^\ell) \le \frac{\ell}{n} b_i, \qquad \text{for all } i \in \{1,\dots,m\},\ \ell \in L$$

given $B = \min_i b_i \ge \frac{10m\log(n/\epsilon)}{\epsilon^2}$.

PROOF. The proof is similar to the proof of Lemma 2.2 but a more careful analysis is needed. We provide a brief outline here with a detailed proof in Appendix B.1. First, we fix $\boldsymbol{p}$, i and ℓ. This time, we say a random order is "bad" for this $\boldsymbol{p}$, i and ℓ if and only if $\boldsymbol{p} = \hat{\boldsymbol{p}}^l$ but $\sum_{t=\ell+1}^{2\ell} a_{it} x_t(\hat{\boldsymbol{p}}^l) > \frac{l}{n} b_i$. By using the Hoeffding-Berstein's Inequality, we show that the probability of "bad" orders is less than $\delta = \frac{\epsilon}{m \cdot n^m \cdot E}$ for any fixed $\boldsymbol{p}$, i and ℓ under the condition on B. Then by taking union bound over all distinct prices, all items i and periods ℓ, the lemma is proved. □

In the following, we will use some notations. Let $\texttt{LP}_s(\boldsymbol{d})$ denote the partial linear program that is defined on variables till time s, i.e. $(x_1,\dots,x_s)$, with right hand side in the inequality constraints set as $\boldsymbol{d}$. That is,

$$\texttt{LP}_s(\boldsymbol{d}): \quad \begin{array}{lll} \text{maximize} & \sum_{t=1}^{s} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{s} a_{it} x_t \le d_i, & i = 1,\dots,m \\ & 0 \le x_t \le 1, & t = 1,\dots,s \end{array}$$

And let $\text{OPT}_s(\boldsymbol{d})$ denote the optimal objective value for $\texttt{LP}_s(\boldsymbol{d})$.

LEMMA 3.2 *With probability at least $1-\epsilon$, for all $\ell \in L$:*

$$\sum_{t=1}^{2\ell} \pi_t x_t(\hat{\boldsymbol{p}}^\ell) \ge (1 - 2h_\ell - \epsilon) OPT_{2\ell}(\frac{2\ell}{n}\boldsymbol{b})$$

given $B = \min_i b_i \ge \frac{10m\log(n/\epsilon)}{\epsilon^2}$.

PROOF. Let $\hat{b}_i = \sum_{j=1}^{2\ell} a_{ij} x_j(\hat{\boldsymbol{p}}^\ell)$ for i such that $p_i^\ell > 0$, and $\hat{b}_i = \max\{\sum_{j=1}^{2\ell} a_{ij} x_j(\hat{\boldsymbol{p}}^\ell), \frac{2\ell}{n} b_i\}$, otherwise. Then, note that the solution pair $(\{x_t(\hat{\boldsymbol{p}}^\ell)\}_{t=1}^{2\ell}, \hat{\boldsymbol{p}}^\ell)$, satisfies all the complementarity conditions, and therefore is an optimal primal-dual solution for the linear program $\texttt{LP}_{2\ell}(\hat{\boldsymbol{b}})$:

$$\begin{array}{lll} \text{maximize} & \sum_{t=1}^{2\ell} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{2\ell} a_{it} x_t \le \hat{b}_i, & i = 1,\dots,m \\ & 0 \le x_t \le 1, & t = 1,\dots,2\ell \end{array} \tag{23}$$

This means

$$\sum_{t=1}^{2\ell} \pi_t x_t(\hat{\boldsymbol{p}}^\ell) = \text{OPT}_{2\ell}(\hat{\boldsymbol{b}}) \ge \left(\min_i \frac{\hat{b}_i}{b_i \frac{2\ell}{n}}\right) \text{OPT}_{2\ell}(\frac{2\ell}{n}\boldsymbol{b}) \tag{24}$$

Now, let us analyze the ratio $\frac{\hat{b}_i}{2\ell b_i/n}$. By definition, for i such that $p_i^\ell = 0$, $\hat{b}_i \ge 2\ell b_i/n$. Otherwise, using techniques similar to the proof of Lemma 3.1, we can prove that with probability $1-\epsilon$,

$$\hat{b}_i = \sum_{t=1}^{2\ell} a_{it} x_t(\hat{\boldsymbol{p}}^\ell) \ge (1 - 2h_\ell - \epsilon)\frac{2\ell}{n} b_i \tag{25}$$

A detailed proof of inequality (25) appears in appendix B.2. And the lemma follows from (25). □

Next, similar to Lemma 2.4 in previous section, we prove the following lemma relating the sample optimal to the optimal value of the offline linear program:

LEMMA 3.3 *For any ℓ,*

$$\mathbb{E}\left[OPT_\ell(\frac{\ell}{n}\boldsymbol{b})\right] \le \frac{\ell}{n} OPT$$

PROOF. The proof is exactly the same as the proof for Lemma 2.4. □

Now we are ready to prove Theorem 1.1.

Proof of Theorem 1.1: Observe that the output of the online solution at time $t \in \{\ell+1, \ldots, 2\ell\}$ is $x_t(\hat{\boldsymbol{p}}^\ell)$ as long as the constraints are not violated. By Lemma 3.1 and Lemma 3.2, with probability at least $1-2\epsilon$:

$$\sum_{t=\ell+1}^{2\ell} a_{it} x_t(\hat{\boldsymbol{p}}^\ell) \le \frac{\ell}{n} b_i, \qquad \text{for all } i \in \{1,\ldots,m\},\ \ell \in L$$

$$\sum_{t=1}^{2\ell} \pi_t x_t(\hat{\boldsymbol{p}}^\ell) \ge (1 - 2h_\ell - \epsilon)\mathrm{OPT}_{2\ell}(\frac{2\ell}{n}\boldsymbol{b})$$

Denote this event by $\mathcal{E}$, where $\Pr(\mathcal{E}) \ge 1-2\epsilon$. Given this event, the expected objective value achieved by the online algorithm can be bounded as follows:

$$\begin{aligned}
& \mathbb{E}[\sum_{\ell\in L}\sum_{t=\ell+1}^{2\ell} \pi_t x_t(\hat{\boldsymbol{p}}^\ell)|\mathcal{E}] \\
\ge\ & \sum_{l\in L}\mathbb{E}\left[\sum_{t=1}^{2\ell}\pi_t x_t(\hat{\boldsymbol{p}}^l)|\mathcal{E}\right] - \sum_{\ell\in L}\mathbb{E}\left[\sum_{t=1}^{\ell}\pi_t x_t(\hat{\boldsymbol{p}}^\ell)|\mathcal{E}\right] \\
\ge\ & \sum_{\ell\in L}(1-2h_l-\epsilon)\mathbb{E}\left[\mathrm{OPT}_{2\ell}(\frac{2\ell}{n}\boldsymbol{b})|\mathcal{E}\right] - \sum_{\ell\in L}\mathbb{E}\left[\mathrm{OPT}_{\ell}(\frac{\ell}{n}\boldsymbol{b})|\mathcal{E}\right] \\
\ge\ & \mathrm{OPT} - \sum_{\ell\in L} 2h_l\mathbb{E}\left[\mathrm{OPT}_{2\ell}(\frac{2\ell}{n}\boldsymbol{b})|\mathcal{E}\right] - \epsilon\sum_{\ell\in L}\mathbb{E}\left[\mathrm{OPT}_{2\ell}(\frac{2\ell}{n}\boldsymbol{b})|\mathcal{E}\right] - \mathbb{E}\left[\mathrm{OPT}_{\epsilon n}(\epsilon\boldsymbol{b})|\mathcal{E}\right] \\
\ge\ & \mathrm{OPT} - \frac{1}{Pr(\mathcal{E})}\sum_{\ell\in L} 2h_l\mathbb{E}\left[\mathrm{OPT}_{2\ell}(\frac{2\ell}{n}\boldsymbol{b})\right] - \frac{\epsilon}{Pr(\mathcal{E})}\sum_{\ell\in L}\mathbb{E}\left[\mathrm{OPT}_{2\ell}(\frac{2\ell}{n}\boldsymbol{b})\right] - \frac{1}{Pr(\mathcal{E})}\mathbb{E}\left[\mathrm{OPT}_{\epsilon n}(\epsilon\boldsymbol{b})\right] \\
\ge\ & \mathrm{OPT} - \frac{4}{1-2\epsilon}\sum_{\ell\in L} h_l\frac{l}{n}\mathrm{OPT} - \frac{2\epsilon}{1-2\epsilon}\sum_{l\in L}\frac{l}{n}\mathrm{OPT} - \frac{\epsilon}{1-2\epsilon}\mathrm{OPT} \\
\ge\ & \mathrm{OPT} - \frac{13\epsilon}{1-2\epsilon}\mathrm{OPT}
\end{aligned}$$

where the last inequality follows from the fact that

$$\sum_{\ell\in L}\frac{\ell}{n} = (1-\epsilon), \text{ and } \sum_{\ell\in L} h_\ell\frac{\ell}{n} = \epsilon\sum_{\ell\in L}\sqrt{\frac{\ell}{n}} \le 2.5\epsilon$$

Therefore,

$$\mathbb{E}[\textstyle\sum_{\ell\in L}\sum_{t=\ell+1}^{2\ell}\pi_t x_t(\hat{\boldsymbol{p}}^\ell)] \ge \mathbb{E}[\sum_{\ell\in L}\sum_{t=\ell+1}^{2\ell}\pi_t x_t(\hat{\boldsymbol{p}}^\ell)|\mathcal{E}]\Pr(\mathcal{E}) \ge (1-15\epsilon)\mathrm{OPT}$$

Thus, Theorem 1.1 is proved.

□

4. Necessity of condition on B In this section, we prove Theorem 1.2, i.e., the condition $B \ge \Omega(\log m/\epsilon^2)$ is necessary for any online algorithm to achieve a competetive ratio of $1-O(\epsilon)$. We prove this by constructing an instance of online packing problem with m items and B units of each item such that no online algorithm can achieve a competetive ratio of $1-O(\epsilon)$ unless $B \ge \Omega(\log m/\epsilon^2)$.

In this construction, we refer to the $0-1$ vectors $\boldsymbol{a}_t$'s as demand vectors, and π_t's as profit coefficients. Assume $m = 2^z$ for some integer z. We will construct z pairs of demand vectors such that demand vectors in each pair are complement of each other, and do not share any item. However, every set of z vectors consisting of exactly one vector from each pair will share at least one common item.

To achieve this, consider the 2^z possible boolean strings of length z. The j^{th} boolean string represents j^{th} item for $j = 0, \ldots, 2^z-1$. Let s_{ij} denote the value at i^{th} bit of the j^{th} string. Then, construct a pair of demand vectors $\boldsymbol{v}_i, \boldsymbol{w}_i \in \{0,1\}^m$, by setting $v_{ij} = s_{ij}$, $w_{ij} = 1-s_{ij}$. Following figure illustrates this construction for $m = 8$ ($z = 3$):

Note that pair of vectors $\boldsymbol{v}_i, \boldsymbol{w}_i, i = 1, \ldots, z$ are complement of each other. Consider any set of z demand vectors formed by picking exactly one of the two vectors $\boldsymbol{v}_i$ and $\boldsymbol{w}_i$ for each $i = 1, \ldots, z$. Then,

Demand vectors

Items	v_3	v_2	v_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Demand vectors

Items	w_3	w_2	w_1
0	1	1	1
1	1	1	0
2	1	0	1
3	1	0	0
4	0	1	1
5	0	1	0
6	0	0	1
7	0	0	0

Figure 1: Construction for $m = 8, z = 3$

form a bit string by setting $s_{ij} = 1$ if this set has vector $\boldsymbol{v}_i$ and 0 if it has vector $\boldsymbol{w}_i$. Then, all the vectors in this set share the item corresponding to the boolean string. For example, in above, the demand vectors $\boldsymbol{v}_3, \boldsymbol{w}_2, \boldsymbol{w}_1$ share item $4(=' 100')$, the demand vectors $\boldsymbol{w}_3, \boldsymbol{v}_2, \boldsymbol{v}_1$ share item $3(=' 011')$ and so on.

Now, we construct an instance consisting of

- B/z inputs with profit coefficient 4 and demand vector $\boldsymbol{v}_i$, for each $i = 1, \ldots, z$.
- q_i inputs with profit 3 and demand vector $\boldsymbol{w_i}$, for each i, where q_i is a Binomial$(2B/z, 1/2)$ random variable.
- $\sqrt{B/4z}$ inputs with profit 2 and demand vector $\boldsymbol{w_i}$, for each i.
- $2B/z - q_i$ inputs with profit 1 and demand vector $\boldsymbol{w}_i$, for each i.

Using the properties of demand vectors ensured in the construction, we prove the following claim:

Claim 1 *Let r_i denote number of vectors of type $\boldsymbol{w}_i$ accepted by any $1-\epsilon$ competitive solution for the constructed example. Then, it must hold that*

$$\sum_i |r_i - B/z| \leq 7\epsilon B$$

Proof. Let OPT denote the optimal value of the offline program. And let OPT_i denote the revenue obtained from demands accepted of type i. Let $\text{topw}_i(k)$ denote sum of profits of top k inputs with demand vector $\boldsymbol{w}_i$. Then

$$\text{OPT} = \sum_{i=1}^{z} \text{OPT}_i \geq \sum_{i=1}^{z} (4B/z + \text{topw}_i(B/z)) = 4B + \sum_{i=1}^{z} \text{topw}_i(B/z)$$

Let $\widehat{\text{OPT}}$ be the objective value of a solution which accepts r_i vectors of type $\boldsymbol{w}_i$. Firstly, note that $\sum_i r_i \leq B$. This is because all $\boldsymbol{w}_i$s share one common item, and there are at most B units of this item available. Let Y be the set $\{i : r_i > B/z\}$, and X be the remaining i's, i.e. $X = \{i : r_i \leq B/z\}$. Then, we show that total number of accepted $\boldsymbol{v}_i$s cannot be more than $B - \sum_{i\in Y} r_i + |Y|B/z$. Obviously, the set Y cannot contribute more than $|Y|B/z$ $\boldsymbol{v}_i$s. Let $S \subseteq X$ contribute the remaining v_is. Now consider the item that is common to all $\boldsymbol{w}_i$s in set Y and $\boldsymbol{v}_i$s in the set S (there is at least one such item by construction). Since only B units of this item are available, total number of $\boldsymbol{v}_i$s contributed by S cannot be more than $B - \sum_{i\in Y} r_i$. Therefore number of accepted $\boldsymbol{v}_i$s is less than or equal to $B - \sum_{i\in Y} r_i + |Y|B/z$.

Denote $P = \sum_{i\in Y} r_i - |Y|B/z$, $M = |X|B/z - \sum_{i\in X} r_i$. Then, $P, M \geq 0$. And the objective value

$$\begin{aligned} \widehat{\text{OPT}} &\leq \textstyle\sum_{i=1}^{z} \text{topw}_i(r_i) + 4(B - \sum_{i\in Y} r_i + |Y|B/z) \\ &\leq \textstyle\sum_i \text{topw}_i(B/z) + 3P - M + 4(B - P) \\ &= \text{OPT} - P - M \end{aligned}$$

Since $\text{OPT} \leq 7B$, this means that, $P + M$ must be less than $7\epsilon B$ in order to get an approximation ratio of $1 - \epsilon$ or better. □

Here is a brief description of the remaining proof. By construction, for every i, there are exactly $2B/z$ demand vectors $\boldsymbol{w}_i$ that have profit coefficients 1 and 3, and those have equal probability to take value 1 or 3. Now, from the previous claim, in order to get near-optimal solution, one must select close to B/z demand vectors of type $\boldsymbol{w}_i$. Therefore, if total number of $(3, \boldsymbol{w}_i)$ inputs are more than B/z, then selecting any $(2, \boldsymbol{w}_i)$ will cause a loss of 1 in profit as compared to the optimal profit; and if total number of $(3, \boldsymbol{w}_i)$ inputs are less than $B/z - \sqrt{B/4z}$, then rejecting any $(2, \boldsymbol{w}_i)$ will cause a loss of 1 in profit. Using central limit theorem, at any step, both these events can happen with constant probability. Thus, every decision for $(2, \boldsymbol{w}_i)$ is a mistake with constant probability, which results in a total expected loss of $\Omega(\sqrt{B/z})$ for every i, that is, a total loss of $\Omega(\sqrt{zB})$.

If the number of $\boldsymbol{w}_i$s to be accepted is not exactly B/z, some of these $\sqrt{B/z}$ decisions may not be mistakes, but as in the claim above, such cases cannot be more than $7\epsilon B$. Therefore, the expected value of online solution,

$$\text{ONLINE} \leq \text{OPT} - \Omega(\sqrt{zB} - 7\epsilon B)$$

Since $\text{OPT} \leq 7B$, in order to get $(1-\epsilon)$ approximation factor, we need

$$\Omega(\sqrt{z/B} - 7\epsilon) \leq 7\epsilon \Rightarrow B \geq \Omega(z/\epsilon^2) = \Omega(\log(m)/\epsilon^2)$$

This completes the proof of Theorem 1.2. A detailed exposition of steps used in this proof appears in Appendix C.

5. Extensions

We present a few extensions and implications of our results.

5.1 Alternative bounds in terms of OPT

In Theorem 1.1, we required lower bound conditions on the right-hand side input $B = \min_i b_i$. Interestingly, our conditions can be easily translated into a corresponding condition that depends only on the optimal offline objective value OPT instead of B. In particular, we can show that

PROPOSITION 5.1 *For any $\epsilon > 0$, our online algorithm is $1 - O(\epsilon)$ competitive for the online linear program (2) in random permutation model, for all inputs such that*

$$\frac{OPT}{\pi_{\max}} \geq \Omega\left(\frac{m^2 \log(n/\epsilon)}{\epsilon^2}\right) \tag{26}$$

where $\pi_{\max} = \max_{j=1,\ldots,n} \pi_j$.

We may remark that the above proposition improves upon the existing results that depend on OPT ([13]) by reducing the dependence on ϵ from ϵ^3 to ϵ^2. A key to these improvements is our dynamic pricing approach as opposed to the usual one-time learning approach.

Out of the two forms of conditions (in terms of B as in Theorem 1.1, and in terms of OPT as in Proposition 5.1), the preferable form may depend on the application at hand. Meanwhile, we can show that none of the two conditions subsumes another. In Appendix F, we give examples showing that either of the conditions could be stronger depending on the problem instance.

5.2 Online multi-dimensional linear program

We consider the following more general online linear programs with multi-dimensional decisions $\boldsymbol{x}_t \in \mathbb{R}^k$ at each step, as defined in Section 1:

$$\begin{array}{lll} \text{maximize} & \sum_{t=1}^n \boldsymbol{f}_t^T \boldsymbol{x}_t & \\ \text{subject to} & \sum_{t=1}^n \boldsymbol{g}_{it}^T \boldsymbol{x}_t \leq b_i & i = 1, \ldots, m \\ & \boldsymbol{x}_t^T \boldsymbol{e} \leq 1, \boldsymbol{x}_t \geq 0 & \forall t \\ & \boldsymbol{x}_t \in \mathbb{R}^k & \end{array} \tag{27}$$

Our online algorithm remains essentially the same (as described in Section 3), with $\boldsymbol{x}_t(\boldsymbol{p})$ now defined as follows:

$$\boldsymbol{x}_t(\boldsymbol{p}) = \begin{cases} 0 & \text{if for all } j,\ f_{tj} \leq \sum_i p_i g_{itj} \\ \boldsymbol{e}_r & \text{otherwise, where } r \in \arg\max_j (f_{tj} - \sum_i p_i g_{itj}) \end{cases} \tag{28}$$

Using the complementary conditions of (27), and the lower bound condition on B as assumed in Theorem 1.3, we can prove the following lemmas; the proofs are very similar to the proofs for the one-dimensional case, and will be provided in the appendix.

LEMMA 5.1 *$\boldsymbol{x}_t^*$ and $\boldsymbol{x}_t(\boldsymbol{p}^*)$ differ in at most m values of t.*

LEMMA 5.2 *Let $\boldsymbol{p}$ and $\boldsymbol{q}$ are distinct if $\boldsymbol{x}_t(\boldsymbol{p}) \neq \boldsymbol{x}_t(\boldsymbol{q})$ for some t. Then, there are at most $n^m k^{2m}$ distinct price vectors.*

With the above lemmas, the proof of Theorem 1.3 will follow exactly as the proof for Theorem 1.1.

5.3 General constraint matrices In (1), we assumed that each entry of the constraint matrix is between zero and one and the coefficients in the objective function is positive. Here, we show that these restrictions can be relaxed to

$$\pi_j \text{ either positive or negative, and } \quad \boldsymbol{a}_j = (a_{ij})_{i=1}^m \in [-1,1]^m.$$

(In Remark 1.1, we have already shown that we can deal with large coefficients by first normalizing them using a row scaling.)

First it is easy to note that the requirement $\pi_j \geq 0$ can be relaxed since we never used this condition in our proof.

When negative entries exist in the coefficient matrix, everything in the proof is the same except that there is risk of running out of inventory during the process. The following lemma which is a strengthened statement of Lemma 3.1 shows that this will not happen with high probability in our dynamic price updating algorithm.

LEMMA 5.3 *For any $\epsilon > 0$, with probability $1-\epsilon$:*

$$\sum_{t=\ell+1}^{w} a_{it} x_t(\hat{\boldsymbol{p}}^\ell) \leq \frac{\ell}{n} b_i, \qquad \text{for all } i \in \{1,\ldots,m\},\ \ell \in L,\ \ell+1 \leq w \leq 2\ell$$

given $B = \min_i b_i \geq \frac{10(m+1)\log(n/\epsilon)}{\epsilon^2}$.

PROOF. The idea of the proof is to add an extra n factor to the δ defined in proving Lemma 3.1 to guarantee that for every step, our algorithm will not violate the inventory constraint. This will only affect the condition on B by increasing m to $m+1$ which is no more than a constant factor. The detailed proof is given in Appendix E.1. □

Note that allowing negative a_{ij}'s may have important implications in many management problems, which means that we not only allow to sell products, but also allow buying from others. This will be a fairly general model for many practical management problems.

5.4 Online integer programs From the definition of $x_t(\boldsymbol{p})$ in (12), the algorithm always outputs integer solutions. And, since the competitive ratio analysis will compare the online solution to the optimal solution of the corresponding linear relaxation, the competitive ratio stated in Theorem 1.1 also holds for the online integer programs. The same observation holds for the general online linear programs introduced in Section 5.2 since it also outputs integer solutions. Our result implies a common belief: when relatively sufficient resource quantities are to be allocated to a large number of small demands, linear programming solutions possess a small gap to integer programming solutions.

5.5 Fast solution of large linear programs by column sampling Apart from online problems, our algorithm can also be applied for solving (offline) linear programs that are too large to consider all the variables explicitly. Similar to the one-time learning online solution, one could randomly sample a small subset ϵn of variables, and use the dual solution $\hat{\boldsymbol{p}}$ for this smaller program to set the values of variables x_j as $x_j(\hat{\boldsymbol{p}})$. This approach is very similar to the popular column generation method used for solving large linear programs [12]. Our result provides the first rigorous analysis of the approximation achieved by the approach of reducing the linear program size by randomly selecting a subset of columns.

6. Conclusions In this paper, we provided a $1-o(1)$ competitive algorithm for a general class of online linear programming problems under the assumption of random order of arrival and a mild condition on the right-hand-side input. The condition we use is independent of the optimal objective value, objective function coefficients, or distributions of input data.

Our dynamic learning-based algorithm works by dynamically updating a threshold price vector at geometric time intervals. The algorithm provides the best known bounds for many generalized secretary problems including online routing, adwords allocation problems, etc; also it presents a first non-asymptotic analysis for the online LP, a model frequently used in revenue management and resource allocation. Besides, the geometric learning frequency may also be of interest to management practitioners. It essentially indicates that not only it might be bad to react too slow, but also to react too fast.

There are many remaining questions. Theoretically, is the bound on the size of the right-hand-input B the optimal one can achieve under the current model? We tried to answer this question partially by proving that the bound must be at least $\Omega(\log(m)/\epsilon^2)$. Thus, while the dependence on ϵ is provably optimal, that is not case for the dependence on m achieved by our algorithm; could we find an alternative proving technique to improve it? Furthermore, the comparison between this dynamic price updating mechanism and the other price updating methods in practice will be very interesting.

References

[1] S. Agrawal. Online multi-item multi-unit auctions. Technical report, 2008. URL: `http://www.stanford.edu/~shipra/Publications/quals-report.pdf`.

[2] S. Agrawal, Z. Wang, and Y. Ye. A dynamic near-optimal algorithm for online linear programming. 2009. URL: `http://arxiv.org/abs/0911.2974`.

[3] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *FOCS'93: Proceeding of the 34th Annual Symposium on Foundations of Computer Science*, pages 32–40, 1993.

[4] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exch.*, 7(2):1–11, 2008.

[5] M-F. Balcan, A. Blum, J. D. Hartline, and Y. Mansour. Mechanism design via machine learning. In *FOCS'05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 605–614, 2005.

[6] G. Bitran and R. Caldentey. An overview of pricing models for revenue management. *Manufacturing and Service Operations Management*, 5(3):203–229, 2003.

[7] N. Buchbinder, K. Jain, and J. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Algorithms–ESA 2007*, pages 253–264, 2007.

[8] N. Buchbinder and J. Naor. Improved bounds for online routing and packing via a primal-dual approach. In *FOCS'06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 293–304, 2006.

[9] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009.

[10] S. Chawla, J. D. Hartline, D. Malec, and B. Sivan. Sequential posted pricing and multi-parameter mechanism design. In *STOC'10: Proceedings of the 42nd ACM symposium on Theory of computing*, pages 311–320, 2010.

[11] W. L. Cooper. Asymptotic behavior of an allocation policy for revenue management. *Operations Research*, 50(4):720–727, 2002.

[12] G. Dantzig. *Linear Programming and Extensions.* Princeton University Press, August 1963.

[13] N. R. Devanur and T. P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *EC'09: Proceedings of the tenth ACM conference on Electronic Commerce*, pages 71–78, 2009.

[14] W. Elmaghraby and P. Keskinocak. Dynamic pricing in the presence of inventory considerations: Research overview, current practices and future directions. *Management Science*, 49(10):1287–1389, 2003.

[15] J. Feldman, M. Henzinger, N. Korula, V. Mirrokni, and C. Stein. Online stochastic packing applied to display ad allocation. *Algorithms–ESA 2010*, pages 182–194.

[16] J. Feldman, N. Korula, V. Mirrokni, S. Muthukrishnan, and M. Pal. Online ad assignment with free disposal. In *WINE'09: Proceedings of the fifth Workshop on Internet and Network Economics*, pages 374–385, 2009.

[17] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1 - 1/e. In *FOCS'09: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126, 2009.

[18] G. Gallego and G. van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science*, 40(8):999–1020, 1994.

[19] G. Gallego and G. van Ryzin. A multiproduct dynamic pricing problem and its application to network yield management. *Operations Research*, 45(1):24–41, 1997.

[20] G. Goel and A. Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA'08: Proceedings of the nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 982–991, 2008.

[21] R. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA'05: Proceedings of the sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 630–631, 2005.

[22] C. Maglaras and J. Meissner. Dynamic pricing strategies for multiproduct revenue management problems. *Manufacturing and Service Operations Management*, 8(2):136–148, 2006.

[23] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized on-line matching. In *FOCS'05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273, 2005.

[24] P. Orlik and H. Terao. *Arrangement of Hyperplanes*. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Springer-Verlag, Berlin, 1992.

[25] R. W. Simpson. Using network flow techniques to find shadow prices for market and seat inventory control. *MIT Flight Transportation Laboratory Memorandum M89-1, Cambridge, MA*, 1989.

[26] K. Talluri and G. van Ryzin. An analysis of bid-price controls for network revenue management. *Management Science*, 44(11):1577–1593, 1998.

[27] A. van der Vaart and J. Wellner. *Weak Convergence and Empirical Processes: With Applications to Statistics (Springer Series in Statistics)*. Springer, November 1996.

[28] E. L. Williamson. Airline network seat control. *Ph. D. Thesis, MIT, Cambridge, MA*, 1992.

Appendix A. Supporting lemmas for Section 2

A.1 Hoeffding-Bernstein's Inequality By Theorem 2.14.19 in [27]:

LEMMA A.1 [Hoeffding-Bernstein's Inequality] *Let $u_1, u_2, ...u_r$ be a random sample without replacement from the real numbers $\{c_1, c_2, ..., c_R\}$. Then for every $t > 0$,*

$$P(|\sum_i u_i - r\bar{c}| \geq t) \leq 2\exp(-\frac{t^2}{2r\sigma_R^2 + t\Delta_R}) \tag{29}$$

where $\Delta_R = \max_i c_i - \min_i c_i$, $\bar{c} = \frac{1}{R}\sum_i c_i$, and $\sigma_R^2 = \frac{1}{R}\sum_{i=1}^{R}(c_i - \bar{c})^2$.

A.2 Proof of Lemma 2.1: Let $\boldsymbol{x}^*, \boldsymbol{p}^*$ be optimal primal-dual solution of the offline problem (1). From KKT conditions of (1), $x_t^* = 1$, if $(\hat{\boldsymbol{p}}^*)^T\boldsymbol{a}_t < \pi_t$ and $x_t^* = 0$ if $(\boldsymbol{p}^*)^T\boldsymbol{a}_t > \pi_t$. Therefore, $x_t(\boldsymbol{p}^*) = x_t^*$ if $(\boldsymbol{p}^*)^T\boldsymbol{a}_t \neq \pi_t$. By Assumption 2.1, there are at most m values of t such that $(\boldsymbol{p}^*)^T\boldsymbol{a}_t \neq \pi_t$.

A.3 Proof of inequality (19) We prove that with probability $1 - \epsilon$

$$\hat{b}_i = \sum_{t\in N} a_{it}x_t(\hat{\boldsymbol{p}}) \geq (1 - 3\epsilon)b_i$$

given $\sum_{t\in S} a_{it}x_t(\hat{\boldsymbol{p}}) \geq (1 - 2\epsilon)\epsilon b_i$. The proof is very similar to the proof of Lemma 2.2. Fix a price vector $\boldsymbol{p}$ and i. Define a permutation is "bad" for $\boldsymbol{p}, i$ if both (a) $\sum_{t\in S} a_{it}x_t(\boldsymbol{p}) \geq (1 - 2\epsilon)\epsilon b_i$ and (b) $\sum_{t\in N} a_{it}x_t(\boldsymbol{p}) \leq (1 - 3\epsilon)b_i$ hold.

Define $Y_t = a_{it}x_t(\boldsymbol{p})$. Then, the probability of bad permutations is bounded by:

$$\Pr(|\sum_{t\in S} Y_t - \epsilon\sum_{t\in N} Y_t| \geq \epsilon^2 b_i | \sum_{t\in N} Y_t \leq (1 - 3\epsilon)b_i) \leq 2\exp\left(-\frac{b_i\epsilon^3}{3}\right) \leq \frac{\epsilon}{m\cdot n^m} \tag{30}$$

where the last inequality follows from the inequality that $b_i \geq \frac{6m\log(n/\epsilon)}{\epsilon^3}$. Summing over n^m "distinct" prices and $i = 1, \ldots, m$, we get the desired inequality.

Appendix B. Supporting lemmas for Section 3

B.1 Proof of Lemma 3.1 PROOF. Consider the i^{th} component $\sum_t a_{it}\hat{x}_t$ for a fixed i. For ease of notation, we temporarily omit the subscript i. Define $Y_t = a_t x_t(\boldsymbol{p})$. If $\boldsymbol{p}$ is an optimal dual solution for (21), then by the complementarity conditions, we have:

$$\textstyle\sum_{t=1}^{\ell} Y_t = \sum_{t=1}^{\ell} a_t x_t(\boldsymbol{p}) \le (1-h_\ell) b\frac{\ell}{n} \tag{31}$$

Therefore, the probability of "bad" permutations is bounded by:

$$\begin{array}{rl} & P(\sum_{t=1}^{\ell} Y_t \le (1-h_\ell)\frac{b\ell}{n},\ \sum_{t=\ell+1}^{2\ell} Y_t \ge \frac{b\ell}{n}) \\ \le & P(\sum_{t=1}^{\ell} Y_t \le (1-h_\ell)\frac{b\ell}{n}, \sum_{t=1}^{2\ell} Y_t \ge \frac{2b\ell}{n}) + P(|\sum_{t=1}^{\ell} Y_t - \frac{1}{2}\sum_{t=1}^{2\ell} Y_t| \ge \frac{h_\ell}{2}\frac{b\ell}{n},\ \sum_{t=1}^{2\ell} Y_t \le \frac{2b\ell}{n}) \end{array}$$

Define $\delta = \frac{\epsilon}{m\cdot n^m \cdot E}$. Using Hoeffding-Bernstein's Inequality (Lemma A.1 in appendix, here $R = 2\ell$, $\sigma_R^2 \le b/n$, and $\Delta_R \le 1$), we have:

$$\begin{array}{rcl} P(\sum_{t=1}^{\ell} Y_t \le (1-h_\ell)\frac{b\ell}{n}, \sum_{t=1}^{2\ell} Y_t \ge \frac{2b\ell}{n}) & \le & P(\sum_{t=1}^{\ell} Y_t \le (1-h_\ell)\frac{b\ell}{n} | \sum_{t=1}^{2\ell} Y_t \ge \frac{2b\ell}{n}) \\ & \le & P(\sum_{t=1}^{\ell} Y_t \le (1-h_\ell)\frac{b\ell}{n} | \sum_{t=1}^{2\ell} Y_t = \frac{2b\ell}{n}) \\ & \le & P(|\sum_{t=1}^{\ell} Y_t - \frac{1}{2}\sum_{t=1}^{2\ell} Y_t| \ge h_\ell \frac{b\ell}{n} | \sum_{t=1}^{2\ell} Y_t = \frac{2b\ell}{n}) \\ & \le & 2\exp\left(-\frac{\epsilon^2 b}{2+h_l}\right) \le \frac{\delta}{2} \end{array}$$

and

$$P(|\sum_{t=1}^{\ell} Y_t - \tfrac{1}{2}\sum_{t=1}^{2\ell} Y_t| \ge \tfrac{h_\ell}{2}\tfrac{b\ell}{n},\ \sum_{t=1}^{2\ell} Y_t \le \tfrac{2b\ell}{n}) \le 2\exp(-\tfrac{\epsilon^2 b}{8+2h_\ell}) \le \tfrac{\delta}{2}$$

where the last steps hold because $h_\ell \le 1$, and the condition made on B.

Next, we take a union bound over n^m distinct prices, $i = 1, \ldots, m$, and E values of ℓ, the lemma is proved. □

B.2 Proof of inequality (25) PROOF. The proof is very similar to the proof of Lemma 3.1. Fix a $\boldsymbol{p}$, ℓ and $i \in \{1, \ldots, m\}$. Define "bad" permutations for $\boldsymbol{p}, i, \ell$ as those permutations such that all the following conditions hold: (a) $\boldsymbol{p} = \hat{\boldsymbol{p}}^{\ell}$, that is, $\boldsymbol{p}$ is the price learned as the optimal dual solution for (21), (b) $p_i > 0$, and (c) $\sum_{t=1}^{2\ell} a_{it} x_t(\boldsymbol{p}) \le (1 - 2h_\ell - \epsilon)\frac{2\ell}{n} b_i$. We will show that the probability of these bad permutations is small.

Define $Y_t = a_{it} x_t(\boldsymbol{p})$. If $\boldsymbol{p}$ is an optimal dual solution for (21), and $p_i > 0$, then by the KKT conditions the i^{th} inequality constraint holds with equality. Therefore, by observation made in Lemma 2.1, we have:

$$\textstyle\sum_{t=1}^{\ell} Y_t = \sum_{t=1}^{\ell} a_{it} x_t(\boldsymbol{p}) \ge (1-h_\ell)\frac{\ell}{n} b_i - m \ge (1 - h_\ell - \epsilon)\frac{\ell}{n} b_i \tag{32}$$

where the second last inequality follows from $B = \min_i b_i \ge \frac{m}{\epsilon^2}$, and $\ell \ge n\epsilon$. Therefore, the probability of "bad" permutations for $\boldsymbol{p}, i, \ell$ is bounded by:

$$P(|\sum_{t=1}^{\ell} Y_t - \tfrac{1}{2}\sum_{t=1}^{2\ell} Y_t| \ge h_\ell \tfrac{b_i \ell}{n} | \sum_{t=1}^{2\ell} Y_t \le (1 - 2h_\ell - \epsilon)\tfrac{2\ell}{n} b_i) \quad \le \quad 2\exp\left(-\tfrac{\epsilon^2 b_i}{2}\right) \le \delta$$

where $\delta = \frac{\epsilon}{m\cdot n^m \cdot E}$. The last inequality follows from the condition on B. Next, we take a union bound over the n^m "distinct" $\boldsymbol{p}$'s, $i = 1, \ldots, m$, and E values of ℓ, we conclude that with probability $1 - \epsilon$

$$\sum_{t=1}^{2\ell} a_{it}\hat{x}_t(\hat{\boldsymbol{p}}^{\ell}) \ge (1 - 2h_\ell - \epsilon)\frac{2\ell}{n} b_i$$

for all i such that $\hat{\boldsymbol{p}}_i > 0$ and all ℓ. □

Appendix C. Detailed steps for Theorem 1.2 Let $c_1, \ldots, c_n$ denote the n customers. For each i, the set $R_i \subseteq \{c_1, \ldots, c_n\}$ of customers with bid vector $\boldsymbol{w}_i$ and bid 1 or 3 is fixed. $|R_i| = 2B/z$ for all i. Conditional on set R_i the bid values of customers $\{c_j, j \in R_i\}$ are independent random variables that take value 1 or 3 with equal probability.

Now consider t^{th} bid $(2, \boldsymbol{w_i})$. In at least 1/2 of the random permutations, the number of bids from set R_i before the bid t is less than B/z. Conditional on this event, with a constant probability the bids in R_i before t take values such that the bids after t can make the number of $(3, \boldsymbol{w}_i)$ bids more than B/z with a constant probability and less than $B/z - \sqrt{B/4z}$ with a constant probability. This probability

calculation is similar to the one used by Kleinberg [21] in his proof of necessity of condition $B \geq \Omega(1/\epsilon^2)$. For completeness, we derive it in the Lemma C.1 towards the end of the proof.

Now, in the first kind of instances (where number of $(3, \boldsymbol{w}_i)$ bids are more than B/z) retaining $(2, \boldsymbol{w}_i)$ is a "potential mistake" of 1, and in second kind, skipping $(2, \boldsymbol{w}_i)$ is a potential mistake of 1. We call it a potential mistake because it will cost a revenue loss of 1 if the online algorithm decides to pick B/z of $\boldsymbol{w}_i$ bids. $|r_i - B/z|$ of these mistakes may be recovered in each instance by deciding to pick $r_i \neq B/z$ of $\boldsymbol{w}_i$ bids. The total expected number of potential mistakes is $\Omega(\sqrt{Bz})$ (since there are $\sqrt{B/4z}$ of $(2, \boldsymbol{w}_i)$ bids for every i). By Claim 1, no more than a constant fraction of instances can recover more than $7\epsilon B$ of the potential mistakes. Let ONLINE denote the expected value for the online algorithm over random permutation and random instances of the problem. Therefore,

$$\text{ONLINE} \leq \text{OPT} - \Omega(\sqrt{zB} - 7\epsilon B)$$

Now, observe that $\text{OPT} \leq 7B$. This is because by construction every set of demand vectors (consisting of either $\boldsymbol{v}_i$ or $\boldsymbol{w}_i$ for each i) will have at least 1 item in common, and since there are only B units of this item available, at most $2B$ demand vectors can be accepted giving a profit of at most $7B$. Therefore, $\text{ONLINE} \leq \text{OPT}(1 - \Omega(\sqrt{z/B} - 7\epsilon))$, and in order to get $(1-\epsilon)$ approximation factor we need

$$\Omega(\sqrt{z/B} - 7\epsilon) \leq O(\epsilon) \Rightarrow B \geq \Omega(z/\epsilon^2)$$

This completes the proof of Theorem 1.2.

LEMMA C.1 *Consider $2k$ random variables $Y_j, j = 1, \ldots, 2k$ that take value $0/1$ independently with equal probability. Let $r \leq k$.Then with constant probability $Y_1, \ldots, Y_r$ take value such that $\sum_{j=1}^{2k} Y_j$ can be greater or less than its expected value k by $\sqrt{k}/2$ with equal constant probability.*

$$Pr(|\sum_{j=1}^{2k} Y_j - k| \geq \lceil \sqrt{k}/2 \rceil \mid Y_1, \ldots Y_r) \geq c$$

for some constant $0 < c < 1$.

PROOF.

- Given $r \leq k$, $|\sum_{j \leq r} Y_j - r/2| \leq \sqrt{k}/4$ with constant probability (by central limit theorem).
- Given $r \leq k$, $|\sum_{j > r} Y_j - (2k - r)/2)| \geq 3\sqrt{k}/4$ with constant probability.

Given the above events $|\sum_j Y_j - k| \geq \sqrt{k}/2$, and by symmetry both events have equal probability. □

Appendix D. Online multi-dimensional linear program

D.1 Proof of Lemma 5.1 PROOF. Using Lagrangian duality, observe that given optimal dual solution $\boldsymbol{p}^*$, optimal solution $\boldsymbol{x}^*$ is given by:

$$\begin{array}{ll} \text{maximize} & \boldsymbol{f}_t^T \boldsymbol{x}_t - \sum_i p_i^* \boldsymbol{g}_{it}^T \boldsymbol{x}_t \\ \text{subject to} & \boldsymbol{x}_t^T \boldsymbol{e} \leq 1, \boldsymbol{x}_t \geq 0 \end{array} \tag{33}$$

Therefore, it must be true that if $x_{tr}^* = 1$, then $r \in \arg\max_j f_{tj} - (\boldsymbol{p}^*)^T \boldsymbol{g}_{tj}$ and $f_{tr} - (\boldsymbol{p}^*)^T \boldsymbol{g}_{tr} \geq 0$ This means that for t's such that $\max_j f_{tj} - (\boldsymbol{p}^*)^T \boldsymbol{g}_{tj}$ is strictly positive and $\arg\max_j$ returns a unique solution, $\boldsymbol{x}_t(\boldsymbol{p}^*)$ and $\boldsymbol{x}_t^*$ are identical. By random perturbation argument there can be at most m values of t which do not satisfy this condition (for each such t, $\boldsymbol{p}$ satisfies an equation $f_{tj} - \boldsymbol{p}^T \boldsymbol{g}_{tj} = f_{tl} - \boldsymbol{p}^T \boldsymbol{g}_{tl}$ for some j, l, or $f_{tj} - \boldsymbol{p}^T \boldsymbol{g}_{tj} = 0$ for some j). This means $\boldsymbol{x}^*$ and $\boldsymbol{x}_t(\boldsymbol{p}^*)$ differ in at most m positions. □

D.2 Proof of Lemma 5.2 PROOF. Consider nk^2 expressions

$$\begin{array}{ll} f_{tj} - \boldsymbol{p}^T g_{tj} - (f_{tl} - \boldsymbol{p}^T g_{tl}), & 1 \leq j, l \leq k, j \neq l, 1 \leq t \leq n \\ f_{tj} - \boldsymbol{p}^T g_{tj}, 1 \leq j \leq k, & 1 \leq t \leq n \end{array}$$

$\boldsymbol{x}_t(\boldsymbol{p})$ is completely determined once we determine the subset of expressions out of these nk^2 expressions that are assigned a non-negative value. By theory of computational geometry, there can be at most $(nk^2)^m$ such distinct assignments. □

Appendix E. General constraint matrix case

E.1 Proof of Lemma 5.3 Proof.

The proof is very similar to the proof of Lemma 3.1. To start with, we fix $\boldsymbol{p}$ and ℓ. This time, we say a random order is "bad" for this $\boldsymbol{p}$ if and only if there exists $\ell \in L$, $\ell+1 \le w \le 2\ell$ such that $\boldsymbol{p} = \hat{\boldsymbol{p}}^l$ but $\sum_{t=\ell+1}^{w} a_{it} x_t(\hat{\boldsymbol{p}}^l) > \frac{l}{n} b_i$ for some i . Define $\delta = \frac{\epsilon}{m \cdot n^{m+1} \cdot E}$. Then by Hoeffding-Berstein's Inequality one can show that for each fixed $\boldsymbol{p}$, i ℓ and w, the probability of that bad order is less than δ. Then by taking union bound over all $\boldsymbol{p}$, i, ℓ and w, the lemma is proved. □

Appendix F. Competitive ratio in terms of OPT In this section, we show prove Proposition 5.1 that provides a lower bound condition only in OPT (and not on B). We also show that the conditions on OPT and B don't subsume each other. We will give examples showing that either of the conditions could be stronger depending on the problem instance.

Proof of Proposition 5.1: We only give a brief proof since the main idea is very similar to our main theorem but only with some adjustment of the parameters. Without loss of generality, assume $\pi_{\max} = 1$. Also, we assume $a_{ij} \in \{0,1\}$. We still take $L = \{n\epsilon, 2n\epsilon, ...\}$, and compute the dual price of following program for each $\ell \in L$, which we use on inputs from $\ell+1$ to 2ℓ:

$$\begin{array}{lll} \text{maximize} & \sum_{t=1}^{\ell} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \le (1 - h_{\ell,i}) \frac{\ell}{n} b_i, & i = 1, \dots, m \\ & 0 \le x_t \le 1, & t = 1, \dots, \ell \end{array} \tag{34}$$

Here, instead of using $h_\ell = \epsilon\sqrt{\frac{n}{\ell}}$ as in (21), we are using $h_{\ell,i} = \sqrt{\frac{2nm\log(n/\epsilon)}{\ell b_i}}$. Then by the same argument as we used in the proof of our main theorem, with probability at least $1 - O(\epsilon)$ none of the constraints will be violated and $\sum_{t=1}^{2\ell} a_{it} x_t(\hat{\boldsymbol{p}}^\ell) \ge (1 - 2h_{\ell,i} - \epsilon)\frac{2\ell}{n} b_i$ for all ℓ and i. Therefore, comparing to the optimal value of

$$\begin{array}{lll} \text{maximize} & \sum_{t=1}^{2\ell} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{2\ell} a_{it} x_t \le (1 - 2h_\ell - \epsilon)\frac{2\ell}{n} b_i & i = 1, ..., m \\ & 0 \le x_t \le 1 & t = 1, ..., 2\ell, \end{array} \tag{35}$$

the difference in value of new algorithm is at most $\sum_i (2h_{\ell,i} - 2h_\ell)^+ \frac{2l}{n} b_i$ (we have assumed that $\pi_{max} = 1$). Also, we know by the proof of Theorem 1.1, using (35) gives a value of at least $(1 - O(\epsilon))$OPT, therefore the loss of the new algorithm is at most

$$O(\epsilon OPT) + 2\sum_{\ell,i} (2h_{\ell,i} - 2h_\ell)^+ \frac{\ell}{n} b_i \le O(\epsilon OPT) + O\left(\frac{m^2 \log(n/\epsilon)}{\epsilon}\right) = O(\epsilon OPT)$$

The last inequality is because

$$\begin{aligned} \sum_{\ell,i} (h_{\ell,i} - h_\ell)^+ \frac{\ell}{n} b_i &\le m \sum_{\ell} \sup_b \left(\sqrt{\frac{2m\ell \log(n/\epsilon) b}{n}} - b\epsilon\sqrt{\frac{\ell}{n}} \right) \\ &\le \sum_{\ell} \sqrt{\frac{\ell}{n}} \frac{m^2 \log(n/\epsilon)}{\epsilon} \\ &\le O\left(\frac{m^2 \log(n/\epsilon)}{\epsilon}\right) \end{aligned}$$

When $a_{ij} \notin \{0,1\}$, π_{max} can be taken as the maximum bid price per order, i.e. $\pi_{max} = \max_j \frac{\pi_j}{\sum_i a_{ij}}$. □

Next, we compare Theorem 1.1 and Proposition 5.1 by providing examples where conditions for one may be satisfied and not the other.

Example F.1 **Condition on B fails but condition on $\frac{OPT}{\pi_{max}}$ holds.**
We start from any instance for which both conditions hold. However, we add another item to this instance,

which has very few copies and contributes very little to the optimal solution(much less than OPT/m). Then condition on B will fail but the $\frac{OPT}{\pi_{max}}$ one still holds to guarantee that our algorithm returns a near-optimal solution. Intuitively, condition on $\frac{OPT}{\pi_{max}}$ works better when the coefficients $\pi_i s$ are uniform but the number of items b_i is highly non-uniform.

EXAMPLE F.2 **Condition on $\frac{OPT}{\pi_{max}}$ fails but condition on B holds.**
Again, we start from any instance for which both the conditions hold. Now we replace one coefficient π_i to take a value that equals to the original optimal objective value. Now, $\frac{OPT}{\pi_{max}}$ is less than 2, but the value of B stays unchanged. Intuitively, condition on B condition works better when the number of items are close to uniform but the coefficients $\pi_i s$ are not.

Above examples show that the conditions on B and OPT don't subsume each other. However, since the condition on B is checkable, while the condition on OPT is not, the former might be preferable in practice.