https://pubsonline.informs.org/journal/moor

MATHEMATICS OF OPERATIONS RESEARCH
Vol. 49, No. 1, February 2024, pp. 297–325
ISSN 0364-765X (print), ISSN 1526-5471 (online)

Assortment Planning for Recommendations at Checkout Under Inventory Constraints

Xi Chen,[a] Will Ma,[b,*] David Simchi-Levi,[c] Linwei Xin[d]

[a] Stern School of Business, New York University, New York, New York 10012; [b] Graduate School of Business, Columbia University, New York, New York 10027; [c] MIT Institute for Data, Systems, and Society; Department of Civil and Environmental Engineering; and Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139; [d] Booth School of Business, University of Chicago, Chicago, Illinois 60637

*Corresponding author

Contact: xchen3@stern.nyu.edu, https://orcid.org/0000-0002-9049-9452 (XC); wm2428@gsb.columbia.edu, https://orcid.org/0000-0002-2420-4468 (WM); dslevi@mit.edu, https://orcid.org/0000-0002-4650-1519 (DS-L); linwei.xin@chicagobooth.edu, https://orcid.org/0000-0002-8160-6877 (LX)

Received: July 23, 2018
Revised: October 1, 2019; November 3, 2020; August 19, 2022
Accepted: January 15, 2023
Published Online in Articles in Advance: February 23, 2023

MSC2020 Subject Classification: Primary: 68W27

https://doi.org/10.1287/moor.2023.1357

Abstract. In this paper, we consider a personalized assortment planning problem under inventory constraints, where each arriving customer type is defined by a primary item of interest. As long as that item is in stock, the customer adds it to the shopping cart, at which point the retailer can recommend to the customer an assortment of add-ons to go along with the primary item. This problem is motivated by the new "recommendation at checkout" systems that have been deployed at many online retailers, and it also serves as a framework that unifies many existing problems in online algorithms (e.g., personalized assortment planning, single-leg booking, and online matching with stochastic rewards). In our problem, add-on recommendation opportunities are eluded when primary items go out of stock, which poses additional challenges for the development of an online policy. We overcome these challenges by introducing the notion of an inventory protection level in expectation and derive an algorithm with a 1/4-competitive ratio guarantee under adversarial arrivals.

Funding: This work was supported by the Adobe Data Science Research Award and the Alibaba Innovation Research Award. L. Xin was partly supported by the National Science Foundation (NSF) [Award CMMI-1635160], X. Chen was supported by the NSF [CAREER Award IIS-1845444]. W. Ma and D. Simchi-Levi were supported by the Accenture and MIT Alliance in Business Analytics.

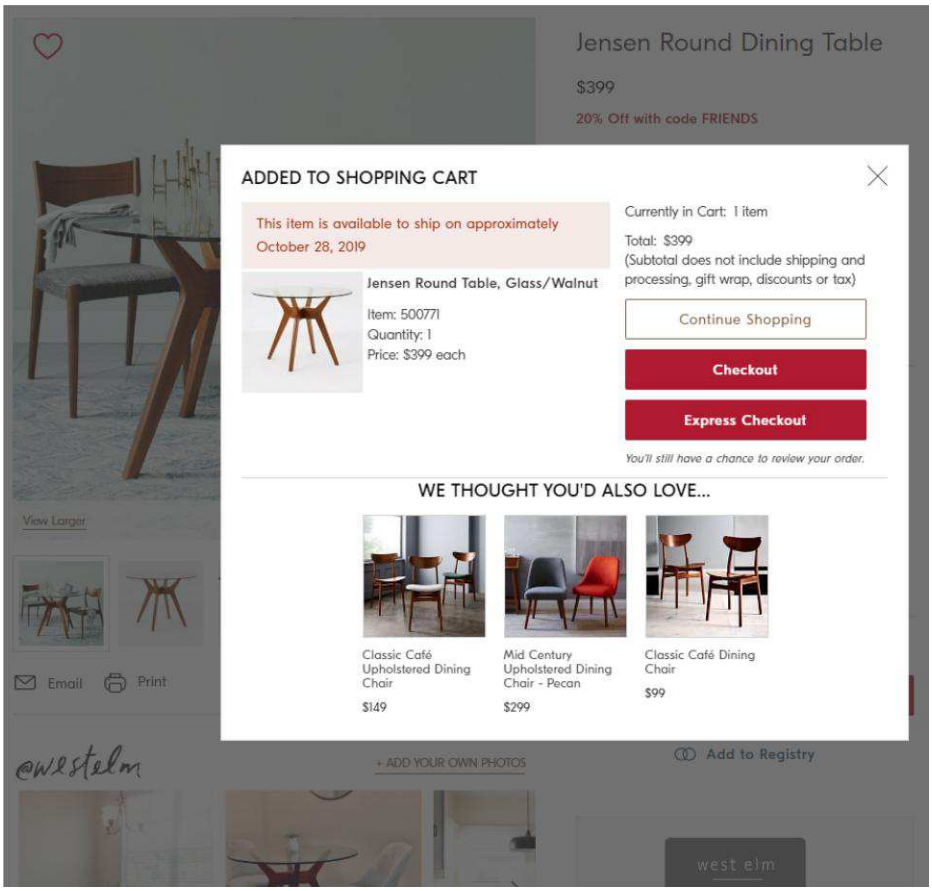
Keywords: revenue management • personalized recommendations • online algorithms • competitive ratio

1. Introduction

The past decade has seen a tremendous increase in the sophistication of online recommendation systems. Because different customers arriving to the website have different preferences, instead of recommending the same products to everyone, *personalized* recommendation has become widely adopted for both enhancing customer experience and boosting revenue. For example, Amazon.com makes recommendations based on a customer's past purchases, the ratings that customer has given to products, and the purchasing behavior of similar customers; this has dramatically increased click-through and conversion rates (Linden et al. [30]). Another example is Netflix, which tailors recommendations of movies to customers' taste. A recent paper by Wood [41] highlights the popularity of personalized recommendation, reviewing several start-ups that are adopting customer-specific operational decisions, including Stitch Fix, Trunk Club, Birchbox, and Club W.

Despite the success of personalized recommendations, it is often difficult to accomplish: many customers are transient and make a one-time purchase without registering an account. Even when historical purchasing information is available, it can be irrelevant or misleading, because a customer can return to the website planning to purchase products in a different category. One way to overcome these obstacles is to make recommendations based on the product a customer has added to her shopping cart. For example, in Figure 1, which is taken from the furniture retailer West Elm, the customer has added a dining table to her shopping cart, and the website has recommended an assortment of chair sets that could accompany the table. As another example, Walmart has begun to implement this idea in the form of a new *recommendation-at-checkout* system for its online grocery. As described in Yuan et al. [44], "When a customer finishes shopping and clicks checkout, she would see a 'stock up' page with at most 8 *additional* items recommended, generated by our system" (p. 62, italics in original). In contrast to traditional recommendation systems, Walmart's recommendation-at-checkout system is based on the products that the customer is

Figure 1. (Color online) Screenshot taken from WestElm.com.



already purchasing. The recommended products are usually complementary to the products already in the cart (e.g., cereal to go with milk) and can even be offered at discounted prices and seen as a bundle deal.

1.1. Model and Main Tension

Motivated by these recommendation-at-checkout systems, we consider the following dynamic recommendation problem. A firm is selling n items with starting inventory amounts $b_1, \ldots, b_n$, and there is no replenishment. Customers arrive sequentially, and each customer t plans on buying her *primary item* i_t (the dining table, in the example in the preceding paragraph). As long as item i_t is in stock, the customer adds it to her shopping cart. At this point, the system recommends to the customer an *assortment of add-on items* (the three chair sets, in the aforementioned example). Add-ons that are out of stock cannot be recommended. In its full generality, our model allows for add-ons to be sold at a fixed discount, and constraints on how many or what types of add-ons can be recommended.

The customer chooses at most one of the add-ons to purchase alongside item i_t, according to known choice probabilities. These choice probabilities are estimated by aggregating the decisions of past customers who added i_t to their shopping carts. In this vein, i_t can be seen as the *type* of customer t, and our model allows for general, different choice functions for customers with different primary items. That is, we allow for n different choice functions, each of which is associated with a different primary item. We make a mild *substitutability* assumption on these choice functions, as introduced in the recent papers by Golrezaei et al. [24] and Gallego et al. [23]; this assumption essentially says that the recommended add-ons are substitutes (in the preceding example, the chair sets are clearly substitutes for each other, because the customer would want at most one to go alongside the dining table).

In this work, we do not consider the learning problem; instead, we focus on the dynamic optimization problem when the choice functions are given. Our goal is to develop a recommendation algorithm, which sequentially decides on the add-on assortment, whenever a primary item is added to a customer's shopping cart. The objective is to maximize the expected profit earned before the selling season is over or the inventories run out. We take the sequence of primary items as exogenously given, and our control lever is which add-ons to recommend.

Online personalized assortment planning is a challenging problem because of the inventory constraints that tie together the assortments that can be recommended to the heterogeneous customers. A few recent papers have pioneered the study of this problem under different settings (Bernstein et al. [4], Gallego et al. [23], Golrezaei et al. [24]). Compared with existing models, the following aspects of our recommendation-at-checkout problem introduce additional challenges for the development of an online policy.

1. In the previously considered choice models, if a specific item is unavailable, it does not decrease the customer's interest in other items. By contrast, in our problem, if a customer's primary item has stocked out, then she never adds it to her shopping cart, and there is no opportunity to recommend other add-on items.

2. Most existing work on assortment optimization assumes that each item has a fixed price. Our results hold even if each item has a discounted price whenever it is recommended as an add-on. In this case, the total profit cannot be analyzed by tallying the inventory depleted (as is done in the previous work), because one would not know how much of that inventory was sold at the full versus discounted price.

All in all, in existing models, an item is withheld from the assortment shown only to reduce the cannibalization of other sales. In our model, with or without discounted add-on prices, the system may want to *withhold an item because it could be someone else's primary item*. As a result, our problem requires new insights on the trade-off between recommending add-ons to maximize immediate profit and preserving inventory over time.

1.1.1. Discussion of the Model and Primary Item Assumption.

We take the perspective of the "recommendation at checkout" team at an e-commerce platform who only recommends add-ons but has no control over the primary items i_t being checked out. Moreover, we assume there is no substitution in the primary items-that is, stocking out of one item (as a result of frequently recommending it as an add-on) does not cause customers to substitute similar ones for their primary items. We justify this assumption in three ways. First, many e-commerce platforms display items in the initial search even if they are out of stock, making customers less likely to substitute with a second choice (contrast this with add-ons, where out-of-stock items can be deliberately omitted). Second, because we are anchoring customers' choice models to their primary item i_t, allowing for a customer t to have multiple i_t's in mind requires an explosion of customer types that is not easy to model (see Xu and Wang [42]; such a model depends on whether the customer is forward looking to the add-on assortment). Finally, the main tension introduced in our recommendation problem is not about substitution, and our main technique of a "protection level in expectation" (discussed subsequently) is needed even in a setting with no substitution. We allow for substitution in the add-ons because the generality is convenient.

1.2. Performance Metric and Main Result

To evaluate online recommendation algorithms, we adopt the worst-case *competitive ratio* as our performance metric. Under this metric, no assumptions are made about the types of future customers; the sequence of customer types can be thought of as chosen *adversarially*. The benefit of this metric theoretically is that it provides a universal approximation guarantee that is irrespective of the arrival process; the competitive ratio is a lower bound to the approximation ratio under any forecasted stochastic arrival model. In practice, algorithms that maximize the competitive ratio are used in settings where the demand is highly uncertain or used in conjunction with forecasting algorithms. For more background on why the competitive ratio is useful, we refer to the book by Borodin and El-Yaniv [7]. Evidence of practical efficacy is also demonstrated through numerical simulations of our competitive-ratio-based algorithm; these are presented in Section 6.

To define the competitive ratio, let $\mathsf{OPT}(i_1,\dots,i_T)$ denote an upper bound based on a linear program (LP) (specified in Section 2.2) to the expected profit of any algorithm, even one that knows both the problem instance $\mathcal{I}$ (including choice functions, prices, and initial inventories) and the sequence of customer types $i_1,\dots,i_T$ in advance. Let $\mathsf{ALG}(i_1,\dots,i_T)$ denote the expected profit of an online algorithm, which knows $\mathcal{I}$ but does not know in advance the sequence $i_1,\dots,i_T$. Note that neither algorithm knows in advance the outcomes of the randomness in the customers' purchase decisions. We then define the competitive ratio of the algorithm to be

$$\inf_{\mathcal{I}} \inf_{i_1,\dots,i_T} \frac{\mathsf{ALG}(i_1,\dots,i_T)}{\mathsf{OPT}(i_1,\dots,i_T)}. \tag{1}$$

Our main result is an algorithm whose competitive ratio is at least $1/4-\varepsilon$, where ε is an error that can be made arbitrarily small at the expense of runtime polynomial in $\frac{1}{\varepsilon}$. The error of ε is introduced by the fact that our overall algorithm needs to sample its own decisions from the past.

1.3. Directly Related Existing Results

We explain our new algorithmic techniques in Section 1.4 but first motivate them by discussing the existing algorithms and bounds that are closely related.

Consider the special case of our problem where the primary item for each customer is a *different* dummy item with price 0 and starting inventory 1. In this case, there is no concern about primary items stocking out, and an assortment of nondummy items (with fixed add-on prices) can be recommended to each of the arriving customers. This is the personalized assortment problem introduced in Golrezaei et al. [24]. They prove that for any starting inventory amounts and *substitutable* choice models, the competitive ratio is at worst $1/2$. This worst case arises when the starting inventories are 1, in which case the algorithm of Golrezaei et al. [24] is equivalent to the greedy algorithm, maximizing the immediate profit from each customer without considering future inventory.

In our general problem, there is the further concern about *protecting* items, because each unit of inventory could sold in one of two ways—the "good option," where it is sold as a primary item, and the "bad option," where it is sold as an add-on—with the good option being better than the bad option by a potentially unbounded factor (because the good option enabled the recommendation of other items). This trade-off has been analyzed in the single-leg booking problem by Ball and Queyranne [3] and Lan et al. [28]. For a single item with two selling options, a competitive ratio of $1/2$ can be achieved by a simple rule: flip a coin of probability $1/2$; if heads, protect *all* units of the item from being sold under the bad option; if tails, allow all units of the item to be sold under either option.

It may appear that our competitive ratio of $1/4$ is easily obtained by combining these two existing results. Indeed, one could decide whether to protect each item with probability $1/2$, losing a factor of $1/2$, and otherwise maximize the immediate profit from each customer, losing another factor of $1/2$ as a result of being greedy. However, this is not the case, because in our problem, the majority of the profit could require a specific *combination* of items. For example, for an arrival sequence, the majority of the profit could occur from selling item 2 as an add-on to item 1, in which case the algorithm only obtains this profit if it both (i) decides to protect item 1 from selling out as an add-on and (ii) decides *not to protect* item 2 from being sold as an add-on.

If we use the strategy of protecting each item with probability $1/2$ independently, then in the preceding example where the profit comes from selling item 2 as an add-on to item 1, we have already lost a factor of $1/2 \times 1/2 = 1/4$ before getting to any additional factor lost as a result of being greedy. Therefore, this does not achieve a competitive ratio of $1/4$. In fact, it appears that any algorithm that fixes the items and levels of inventory to protect beforehand, even if it determines these using some coordinated randomness, has a competitive ratio strictly less than $1/4$; we explain using a detailed example in Section 2.3.

1.4. New Algorithmic Techniques

This brings us to our main algorithmic innovation, which is a *protection level in expectation*. Our algorithm ensures that the expected number of units of each item sold under the bad option does not exceed half of its starting inventory, where *the expectation is over all possible realizations* of the purchase decisions of past customers (and the expectation can be computed because the types of past customers have already been revealed). By contrast, the existing algorithms fix the inventory to protect beforehand and only ensure that on any *specific realization* the protected inventory is not sold under the bad option.

Our algorithm is defined by a simple greedy rule:

> Recommend the profit-maximizing add-on assortment to each customer while ensuring that the expected units of each item sold as an add-on never exceeds half of its starting inventory.

Of course, although this constraint is related to the expected remaining inventory, the algorithm is also physically constrained from recommending any items whose realized remaining inventory is 0. As a result, implementing such a rule poses many challenges. For example, suppose that both items i and j have remaining inventory in reality and that the algorithm is deliberating whether to maximize immediate profit by recommending item j as an add-on to primary item i. Let $d_j^{\texttt{curr}}$ denote the expected number of units of item j that have been sold as an add-on thus far. To evaluate whether recommending item j would push $d_j^{\texttt{curr}}$ above its allowable threshold, the algorithm must know the following:

- If item j is recommended given the current inventory state, how much is added to $d_j^{\texttt{curr}}$? In other words, what is the measure of the present inventory state that has been realized?
- What are the other possible inventory states for the present time step, and what decisions would have been made on those?

We answer these questions by envisioning time as passing fluidly over a unit interval $[0, 1]$ for each time step. We consider the entire distribution for the inventory state $(I_1, \ldots, I_n)$ at the start of the time step and evaluate how the values of $(d_1^{\texttt{curr}}, \ldots, d_n^{\texttt{curr}})$ would increase if we recommend at each state the profit-maximizing assortment from

the items j such that both $I_j > 0$ and $d_j^{\texttt{curr}} < \frac{b_j}{2}$. If this causes any add-on thresholds to be violated by the end of the time step, then we consider the first *breakpoint* s^* in [0, 1] where $d_{j^*}^{\texttt{curr}}$ has reached its threshold for some item j^*. Over the time interval $[0, s^*]$, we recommend the already computed profit-maximizing assortments at each inventory state. In the time interval after s^*, we reoptimize the assortments for each state, where j^* is no longer eligible to be an add-on (and this could completely change the optimal assortments). The algorithm repeats this process, potentially adding multiple breakpoints in a time step, until the end of the continuous interval [0,1] is reached with no threshold violations. Finally, the algorithm returns to reality and makes the corresponding decision given the inventory state at hand, which is in the form of a random assortment if there were breakpoints during the time step.

There is a final computational challenge caused by the fact that there could be exponentially many inventory states for a time step. To compensate, at the start of each time step, we sample a fresh empirical distribution for the inventory state using the known customer types $i_1, \ldots, i_{t-1}$ and simulating the random purchase decisions of these customers (while also simulating the random assortments previously recommended by the algorithm). This simulation idea is based on Ma [32], who presents an algorithm that makes decisions based on expectations over all sample paths instead of the realized sample path. However, our algorithm also incorporates the fact that the customer type information i_t is arriving online, which requires new simulations to be run at each time t. To our knowledge, our work is the first to synthesize this simulation idea (also used in Adamczyk et al. [1] and Brubach et al. [8], among others) with the online information structure of competitive ratio analysis.

1.5. New Analytical Bounds and Comparison with Literature

Bounding the competitive ratio of our online algorithm, which makes decisions based on expectations and estimates them via sampling, also requires new analytical techniques.

We first review the competitive ratio bounds for the personalized assortment (Golrezaei et al. [24]) and single-leg booking (Ball and Queyranne [3], Lan et al. [28]) problems from the literature, as well as the online matching with stochastic rewards problem studied in Mehta and Panigrahi [35] and Mehta et al. [36]. The online matching with stochastic rewards problem is a special case of the personalized assortment problem when the items yield the same reward and the assortment shown is constrained to size 1. Therefore, our problem generalizes all three of these problems, following the explanation in Section 1.3 of how our problem captures the first two problems. Overall, our model incorporates the aspect of inventory protection (studied for one item in the single-leg booking problem) into the online assortment and matching problems (where inventory protection was previously a nonfactor).

Table 1 presents a detailed comparison of our model with existing models, as well as our bounds with existing bounds.

Although the competitive ratio achieved by our analysis is smaller, we do provide an upper bound showing that the existing competitive ratios cannot be achieved for our more general problem. Also, although our lower bound of 1/4 on the competitive ratio does not match our upper bound of 0.43, we are not aware of any tight competitive ratio results under both (i) uncertainty in purchasing and (ii) small starting inventories. To highlight the difficulty

Table 1. Comparison of problems and competitive ratios. For each row, bold font indicates the greatest generality.

Problem	Real-time personalized assortment optimization	Online matching w/stochastic rewards	Two-fare single-leg booking problem	Recommendation at checkout
Reference	Golrezaei et al. [24]	Mehta and Panigrahi [35]	Ball and Queyranne [3]	This paper
No. of items	**Multiple**	**Multiple**	Single	**Multiple**
Prices of items	One price per item	Same reward for all items	**Two prices per item**	**Two prices per item**
Choice function	Arbitrary; assume *substitutability*[a]	Purchase probabilities for single items	Deterministic	**Arbitrary; assume *weak substitutability*[a]**
Lower bound	$\frac{1}{2}$[b]	$\frac{1}{2}$[c]	$\frac{1}{2}$	$\frac{1}{4} - \varepsilon$
Upper bound	0.621		$\frac{1}{2}$	0.43

[a]Golrezaei et al. [24] introduce a necessary, mild *substitutability* assumption on the choice model, where the purchase probability of an item can only decrease as additional items are added into the assortment. We call our assumption *weak substitutability* because while we assume substitutability between the add-ons, we allow the purchase probability of an add-on to decrease (to 0) if the primary item is out of stock.

[b]The competitive ratio improves from $\frac{1}{2}$ to $1 - \frac{1}{e} \approx 0.632$ as the minimum starting inventory increases from 1 to ∞, and in this case, matches the upper bound of $1 - \frac{1}{e}$ (the stronger upper bound of 0.621 from Mehta and Panigrahi [35] does not hold here because it requires unit starting inventories).

[c]The competitive ratio improves to 0.534 if the purchase probabilities approach 0 and 0.567 if they are also equal; the first statement is attributable to Mehta et al. [36].

of the problem-even in the special case of online matching with stochastic rewards-with the further assumption that every purchase probability is equal to a single infinitesimal value ρ, there is a gap between the lower bound of 0.567 and the upper bound of 0.621 (see Mehta and Panigrahi [35] and the later work by Mehta et al. [36]). Only when (i) is relaxed and we have the deterministic online *matching* problem does a "random ranking" online algorithm achieve a tight competitive ratio of $1 - 1/e$ (see Karp et al. [26], its generalization by Aggarwal et al. [2], and the unified analysis by Devanur et al. [18]). Alternatively, when (ii) is relaxed and the starting inventories approach ∞, an "inventory balancing" online algorithm can achieve a tight competitive ratio, also of $1 - 1/e$ (see Golrezaei et al. [24]; this result is closely related to the AdWords stream of research initiated in Buchbinder et al. [10], Kalyanasundaram and Pruhs [25], and Mehta et al. [37]). In Appendix C, we discuss why large starting inventories do not appear to improve our competitive ratio, including an example under this regime that shows that the competitive ratio must still be less than $1/2$.

We see our guarantee of $1/4$ as the natural baseline competitive ratio that combines the loss of $1/2$ from the online matching with stochastic rewards problem with the loss of $1/2$ from items having a good option and a bad option. It is particularly interesting to us that the competitive ratio of $1/4$ is not immediately achieved by flipping a coin for each item to decide whether to sell it under its bad option; instead, it is achieved by constraining the *expected* units of each item sold under its bad option.

Finally, we would like to mention that unlike in the classical online matching problem, where it is immediate that greedy is (1/2)-competitive (Aggarwal et al. [2], Karp et al. [26]), showing that greedy is (1/2)-competitive in the online matching with stochastic rewards problem requires a probabilistic dual-balancing argument (Golrezaei et al. [24], Mehta and Panigrahi [35]). Extending this to a (1/4)-competitive algorithm for our recommendation-at-checkout problem requires new insights into how to define dual variables when the algorithm is based on expected add-on sales instead of realized add-on sales. Furthermore, similar to Ma [32], we need to show that the ε errors accumulated from sampling do not propagate in a way such that the final competitive ratio is worse than $1/4 - \varepsilon$.

1.5.1. Additional Result When Add-on Profit Is Small. When the reward from selling an add-on is always lower than that of the primary item, a simple algorithm that never offers add-ons is $1/2$-competitive. More generally, if the maximum reward from selling any add-on j relative to that of the primary item i is R, then the algorithm that never offers add-ons is $1/(1 + R)$-competitive. In Section 5, we formally analyze this no-add-on algorithm and prove these results.

This no-add-on algorithm has a worse guarantee than our algorithm whenever the value of R is large, which we generally expect to be the case. Indeed, the reward earned from selling an item can be interpreted to be its profit margin, of price minus cost. Many online platforms carry "niche" products with high profit margins that are predominantly sold through recommendations. Additionally, in some industries, it is known that the majority of profit comes from selling items that are usually the add-on. For example, when selling printers, a significant fraction of profit arises from the cartridges that come as add-ons; in console gaming, a significant fraction of profit arises from games or accessories instead of the console itself; and in airlines, a significant fraction of profit arises from the ancillary services.

1.6. Other Related Work

The papers that are most related to ours from a technical standpoint have already been discussed in Sections 1.3–1.5. In this section we emphasize several papers that have been previously unmentioned.

The retailing problem of planning a sequence of assortments over a finite horizon subject to various operational constraints (e.g., assortment cardinality and inventory considerations) has been studied extensively since the seminal papers by van Ryzin and Mahajan [40] and Mahajan and van Ryzin [34]. The model we study here, where the trade-off is between immediate profit and future inventory, was originally studied specifically for the multinomial-logit choice model by Talluri and van Ryzin [39] and Rusmevichientong et al. [38].

In this paper, instead of focusing on a specific choice model, we view the assortment decision as a generic lever for managing the trade-off between different depletion options for the inventories, which have different profit rates (see Chan and Farias [12] and Maglaras and Meissner [33]). Conceptually, our analysis holds as long as the add-ons being recommended satisfy the substitutability condition. Computationally, our algorithm can run in polynomial time as long as the single-period assortment optimization problem corresponding to any of the choice models given can be solved efficiently. The paper by Cheung and Simchi-Levi [13] contains a recent summary on the state of the art in single-customer assortment optimization. We also detail specific single-period assortment optimization oracles that can be used under our framework in Section 2.1.

Our paper provides a worst-case approximation guarantee that makes no assumptions on the arrival sequence. This benchmark is referred to as the competitive ratio under adversarial arrivals; see the book by Borodin and El-Yaniv [7] or see the paper by Buchbinder et al. [11], which discusses some recent unified results. In the context of

the recent assortment planning literature, Golrezaei et al. [24] focus on the worst-case competitive ratio, whereas Bernstein et al. [4] and Gallego et al. [23] consider improved approximation guarantees under additional assumptions about the arrival sequence.

Moreover, although our main focus is on assortment optimization, our problem is related to dynamic pricing with inventory control. Since the seminal work of Gallego and van Ryzin [21], which introduces the runout price to deal with inventory constraints, the dynamic pricing problem with limited inventory has received a lot of attention. A large of amount of effort has been devoted to several important problems in dynamic pricing-for example, multiple product pricing, inventory replenishment, correlated demands over time, and the presence of strategic customers. Please refer to Elmaghraby and Keskinocak [19] and Bitran and Caldentey [5] for comprehensive review. We note that most existing work assumes a stochastic model on the demand function; however, our framework allows an adversarially chosen arrival sequence. Recently, some work in dynamic pricing studied adversarial models (e.g., adversarial demand (Bubeck et al. [9]) or adversarial features (Cohen et al. [14])). However, this work does not take inventory control into consideration.

1.7. Outline of the Paper

The rest of the paper is organized as follows. In Section 2, we introduce the specific models and assumptions (e.g., the problem definition and the general choice model considered in the paper). We also provide instances to motivate our algorithm and establish the upper bound of the competitive ratio in Sections 2.3 and 2.4. This upper bound result illustrates the key features of our problem. In Section 3, we describe our proposed algorithm with several illustrative examples. We also provide the running time of the algorithm. In Section 4, we prove our main result, which lower bounds the competitive ratio of our algorithm. In Section 6, we conduct numerical experiments. Finally, we summarize our main contributions and propose directions for future research in Section 7.

2. Model Specification

Let $\mathbb{R}$ denote the set of real numbers, and let $\mathbb{N}$ denote the set of positive integers. For a general $m \in \mathbb{N}$, let $[m]$ denote the set $\{1, \dots, m\}$.

A firm is selling $n \in \mathbb{N}$ different items. Each item $j \in [n]$ starts with an initial inventory of $b_j \in \mathbb{N}$ units, and there is no replenishment. Each item has two prices: the full price $r_j \in \mathbb{R}$ and a discounted price of $r_j^{\text{disc}} \in \mathbb{R}$ satisfying $r_j \geq r_j^{\text{disc}} \geq 0$ (we allow for the single-price case where $r_j = r_j^{\text{disc}}$).

Following the discussion in the introduction, assuming that leftover inventory does not perish, the firm's objective is to prioritize selling items with a high price compared with cost. Therefore, "prices" r_j and r_j^{disc} should actually be interpreted as "profit margins" (price minus cost). However, as is standard in the revenue management literature, we will hereafter refer to r_j and r_j^{disc} as *prices* and the amounts collected as *revenue*.

There are n customer types, one for each item $i \in [n]$, characterized by the population of customers who arrive planning to purchase item i at its full price. Any items that type i customers purchase other than i are sold at their discounted prices. When a customer of type i arrives, assuming she can add her primary item i to the shopping cart, the firm has an opportunity to recommend her an assortment of add-ons. The probabilities of how she would decide between the add-ons is given in the form of a *choice function*, as we define next.

2.1. Choice Functions for Add-ons

Fix an $i \in [n]$, and let $\mathcal{A}_i$ denote the collection of all add-on assortments that are feasible recommendations to a customer of type i. For $A \in \mathcal{A}_i$ and $j \in [n] \setminus i$, let $q_{ij}(A)$ denote the probability that a type i customer chooses to purchase j as an add-on to item i when shown add-on assortment A. Note that $q_{ij}(A)$ can only be nonzero for $j \in A$ and that it is possible for $\sum_{j \in A} q_{ij}(A)$ to be strictly less than 1, in which case $1 - \sum_{j \in A} q_{ij}(A)$ is the probability that the customer purchases no add-on. We make the following assumptions on $\mathcal{A}_i$ and the values of $q_{ij}(A)$ for all customer types i.

Definition 1. We define the assumptions on feasible assortments and choice probabilities as follows.

(1) Assume $\mathcal{A}_i$ is a *nonempty*, *downward-closed* family of subsets of $[n] \setminus i$. That is, $\emptyset \in \mathcal{A}_i$, and if $A_1 \subseteq A_2 \in \mathcal{A}_i$, then $A_1 \in \mathcal{A}_i$.

(2) The probability of selling an add-on j cannot be increased by recommending it in a larger assortment. That is, if $A_1, A_2 \in \mathcal{A}_i$ and $j \in A_1 \subseteq A_2$, then $q_{ij}(A_1) \geq q_{ij}(A_2)$.

(3) There is an efficient algorithm for solving the single-customer assortment optimization problem over $\mathcal{A}_i$ with choice probabilities $q_{ij}(A)$. That is, for any weights $w_1, \dots, w_n \geq 0$, we are given an oracle that computes

$$\max_{A \in \mathcal{A}_i} \sum_{j \in A} w_j q_{ij}(A). \tag{2}$$

Assumptions (1)–(3) are very mild and also made in the related papers by Golrezaei et al. [24] and Gallego et al. [23].

Assumption (1) on $\mathcal{A}_i$ being downward-closed ensures that even if some add-ons stock out, the remaining ones can still be feasibly recommended. It is satisfied if $\mathcal{A}_i$ is all subsets of $[n] \setminus i$, and it is also satisfied by natural constraints on the recommended assortment, such as the space available for add-ons to be displayed.

Meanwhile, assumption (2) is called the *substitutability* assumption and is naturally satisfied by any choice function that arises from a random-utility model, including the commonly used multinomial logit (MNL) and nested logit models. Equivalently, it is also satisfied by any choice function that arises from a distribution over rankings, such as the Markov chain choice model. It essentially says that the recommended add-ons are substitutes, as is the case in Figure 1 from the introduction, where the customer needs at most one chair set to go along with the dining table she is purchasing.

Finally, assumption (3) allows us to defer the static combinatorial optimization problem in (2) to the literature, which focuses on single-customer assortment optimization. The unconstrained (i.e., $\mathcal{A}_i = 2^{[n]\setminus i}$) single-customer problem has been solved in polynomial-time for the aforementioned MNL (Talluri and van Ryzin [39]), d-level nested logit (Li et al. [29]), and Markov chain (Blanchet et al. [6]) choice models. The constrained problem can be solved for MNL (Rusmevichientong et al. [38]) and nested logit (Gallego and Topaloglu [20]) models and approximately solved for several other variants (Désir et al. [16, 17]).

2.2. Problem Definition and Main Result

Having defined the recommendation and purchasing dynamics with a single customer, we now define the overall problem. There are $T \in \mathbb{N}$ customers arriving sequentially. The type $i_t \in [n]$ of a customer $t \in [T]$ is revealed upon arrival, and we say that she *requests* item i_t. If i_t has stocked out, then she leaves without making a purchase. Otherwise, we say that she is *served*, in which case the firm can recommend to her an assortment of add-ons. The firm's goal is to maximize the total revenue earned in expectation before there are no more customers or when all items sell out. However, the firm must make recommendation decisions in an online fashion without knowing the types of future customers $i_{t+1}, \dots, i_T$. A recommendation algorithm is evaluated by its performance relative to a clairvoyant benchmark that knows the entire sequence $i_1, \dots, i_T$ in advance.

To define this benchmark, we first introduce some notation. For most of this paper, it will be more convenient to refer to assortments with the primary item i included.

Definition 2. We use the following notation.

(1) Let S^t refer to the assortment of *all* items seen by customer t, including item i. In the case where i_t is out of stock, $S^t = \emptyset$. Otherwise, $S^t = \{i\} \cup A$ for some $A \in \mathcal{A}_i$. We say that S^t is the assortment *offered* to customer t.

(2) Let $\mathcal{S}_i = \{\{i\} \cup A : A \in \mathcal{A}_i\}$ denote the collection of all nonempty possibilities for S^t when customer t has type i.

(3) For all $i \in [n]$ and $S \in \mathcal{S}_i$, let $p_{ij}(S)$ denote the marginal probability that $j \in [n]$ is sold when S is offered to a customer of type i. Relating to previous notations in Section 2.1, we have that $p_{ij}(S) = 1$ if $j = i$, and $p_{ij}(S) = q_{ij}(S \setminus i)$ otherwise.

Using the above-mentioned notation, we can define the clairvoyant benchmark to which the algorithm's performance will be compared. For any fixed sequence $(i_1, \dots, i_T)$ of customer types, consider the following LP:

$$\begin{aligned}
\max \quad & \sum_{t=1}^{T} \sum_{S \in \mathcal{S}_{i_t}} \left(r_{i_t} + \sum_{j \neq i_t} r_j^{\text{disc}} p_{i_t j}(S) \right) y^t(S) \\
\text{s.t.} \quad & \sum_{t=1}^{T} \sum_{S \in \mathcal{S}_{i_t}} p_{i_t j}(S) y^t(S) \leq b_j, \qquad j \in [n] \\
& \sum_{S \in \mathcal{S}_{i_t}} y^t(S) \leq 1, \qquad t \in [T] \\
& y^t(S) \geq 0 \qquad t \in [T], S \in \mathcal{S}_{i_t}.
\end{aligned} \tag{3}$$

For all t, variable $y^t(S)$ corresponds to the unconditional probability that assortment S is offered to customer t. The objective function is the expected revenue obtained as a result of these probabilistic offerings. The first set of constraints impose that the expected units of item j sold do not exceed its starting inventory of b_j, whereas the next set of constraints impose that the total probability of customer t being offered a nonempty subset is at most 1.

Let $\mathsf{OPT}(i_1, \dots, i_T)$ denote the optimal objective value of this LP. Although LP (3) has exponentially many decision variables and could be difficult to solve, it is only conceptually used by our analysis, and its solution is not required by our algorithm. This type of LP is commonly referred to in revenue management as the (choice-based) deterministic linear program (Gallego et al. [22]). It can also be found in discrete mathematics problems such as

stochastic knapsack (Dean et al. [15]). In either case, the following is a standard result. We provide a self-contained proof of Lemma 1 in Appendix A.

Lemma 1. *Let* $\mathsf{OPT}(i_1,\dots,i_T)$ *be an upper bound on the expected revenue obtainable by any algorithm, even one that knows* $i_1,\dots,i_T$ *at the start of the time horizon.*

Having established $\mathsf{OPT}(i_1,\dots,i_T)$ as an upper bound, we are now ready to define the competitive ratio.

Definition 3. Fix an online algorithm. For a given instance $\mathcal{I}$ (consisting of choice functions, revenues, and starting inventories), let $\mathsf{ALG}(i_1,\dots,i_T)$ denote the expected revenue earned by the algorithm when the sequence of arrival types is $i_1,\dots,i_T$. Then the *competitive ratio* of the algorithm is

$$\inf_{\mathcal{I}} \inf_{i_1,\dots,i_T} \frac{\mathsf{ALG}(i_1,\dots,i_T)}{\mathsf{OPT}(i_1,\dots,i_T)}.$$

In other words, an adversary who knows the algorithm beforehand (but knows neither the outcome of any randomness in the algorithm nor the customers' purchase decisions) chooses an instance and arrival sequence to minimize the online algorithm's expected revenue as a fraction of $\mathsf{OPT}(i_1,\dots,i_T)$. By Lemma 1, an algorithm's competitive ratio cannot be better than 1, because $\mathsf{OPT}(i_1,\dots,i_T)$ is always an upper bound on expected revenue, even when the online algorithm happens to "guess" the sequence $i_1,\dots,i_T$ correctly. We note that this definition of competitive ratio, based on the LP optimum, is standard in online problems where there is stochasticity in the decisions of the arriving agents (see Golrezaei et al. [24] and Mehta and Panigrahi [35]).

We prove the following bound on the competitive ratio.

Theorem 1. *For any* $\varepsilon > 0$, *our online algorithm* (*whose runtime is polynomial in* $\frac{1}{\varepsilon}$ *and the instance parameters*) *ha sa competitive ratio at least* $\frac{1}{4} - \varepsilon$.

Our algorithm needs to estimate probabilities by sampling virtual outcomes for the decisions of past customers, which explains the error of ε. Before we describe our algorithm, we present an example that provides intuition by illustrating the main "difficulty" in the problem.

2.3. A Motivating Example for Our Algorithm

We present an example demonstrating why the fixed-protection-level algorithm, which would follow naturally from the existing literature discussed in Section 1.3, does not suffice for our problem.

Example 1. Fix $N, M \in \mathbb{N}$, which are large integers with $M \gg N$. There are $n = 3$ items, with starting inventories $b_1 = b_2 = N$ and $b_3 = 1$. Item 1 is a dummy item with $r_1 = 0$. Customers of type 1 can be recommended item 2 as an add-on, earning a small but nonzero revenue of $r_2^{\mathsf{disc}} = r_2 = 1/N$ with probability (w.p.) 1. However, item 2 is better used as a primary item for selling item 3, because customers of type 2 can be recommended item 3 as an add-on to earn a large revenue of $r_3^{\mathsf{disc}} = M$ with probability $1/N$. Only customers of type 1 or 2 can arrive. Item 3 can be interpreted as a niche product that is more profitable but difficult to sell on its own, and it usually serves as an add-on as a result of recommendations instead of as a primary item.

The arrival sequence begins with N customers of type 1, and the firm must decide how many units of item 2 to sell as an add-on at the modest price of $1/N$. If the firm is myopic and sells all of it, then this is suboptimal in the scenario where the arrival sequence continues with N customers of type 2, because the firm would be forgoing N opportunities to earn M/N expected revenue. On the other hand, if the firm sells none of item 2 as an add-on, then it earns zero revenue in the scenario where no customers of type 2 arrive.

Following the discussion of the literature in Section 1.3, the firm can balance between the two extreme scenarios by randomly deciding whether it is going to "protect" items from being sold as an add-on, with *independent* probability 1/2 for each item. However, consider the scenario where type 2 customers do arrive. In order to realize the opportunities of selling item 3 as an add-on, the firm must both (i) have inventory of item 2 remaining to serve the type 2 customers and (ii) not be protecting[1] item 3. By independence, this only occurs with probability 1/4, so the firm's expected revenue would be

$$M \cdot \frac{1}{4}\left(1 - \left(1 - \frac{1}{N}\right)^N\right) + 1 \approx M \cdot \frac{1 - 1/e}{4}. \tag{4}$$

(Note that 1 is added to the left-hand side because of the revenue of $N \cdot 1/N$ from selling item 2 as a primary item, which is negligible.) The result of (4) is less than 1/4 of the optimum defined in Section 2.2 using the LP, which equals $M \cdot (1/N)N = M$.

To our knowledge, there is no immediate solution to the problem demonstrated by (4). Even if the algorithm made the improved decision on this example of protecting half of the units of item 2 deterministically (instead of protecting all of the units with probability 1/2), its expected revenue would only increase to

$$M \cdot \frac{1}{2}\left(1 - \left(1 - \frac{1}{N}\right)^{N/2}\right) + 1 \approx M \cdot \frac{1 - 1/\sqrt{e}}{2}, \tag{5}$$

which is still less than 1/4 of the optimum. In fact, for any fixed fraction α of starting inventory to protect, the expected revenue on this example would be

$$\approx M \cdot (1 - \alpha)(1 - e^{-\alpha}), \tag{6}$$

which is strictly less than 1/4 of the optimum for any $\alpha \in [0, 1]$. There also does not appear[2] to exist a simple way to improve on (6) by *correlating* the randomized rounding across items.

The issue with all of these methods is that the decision on how many units of each item to protect is (randomly) fixed from the start, not adapting to the sequence encountered. In the preceding example, the algorithm should be *always* willing to offer the single unit of item 3 as an add-on for the first time, because its probability of being sold is only $1/N$. Of course, it needs to stop after a certain number of times so that item 3 is not deterministically sold as an add-on. This brings us to our main insight: the algorithm should keep track of the *expected* number of units of item 3 sold and stop offering it once this expectation hits a threshold so that item 3 is not always sold as an add-on. Specifically, our algorithm freely offers an item as an add-on as long as both

- the number of units sold, in reality, has not reached its starting inventory, and
- the number of units sold as an add-on, in expectation over the purchase decisions of the sequence of customers encountered, has not reached 1/2 of its starting inventory.

On this example, our algorithm would behave as follows. It would sell 1/2 of the N units of item 2 as add-ons to item 1. Then, it would offer item 3 as an add-on to item 2 as long as the probability of item 3 being sold does not exceed 1/2. Even after $N/2$ offerings (which is the maximum number of attempts because only $N/2$ units of item 2 remain), the unconditional probability of item 3 being sold as an add-on is

$$1 - \left(1 - \frac{1}{N}\right)^{N/2} \approx 1 - 1/\sqrt{e} = 0.393\ldots,$$

which is less than 1/2. Therefore, our algorithm would offer item 3 as long as it is available, earning expected revenue $M \cdot (1 - 1/\sqrt{e})$, which is twice the amount from (5) and surpasses the target competitive ratio of 1/4.

Our algorithm, in its full generality where assortments of multiple add-ons can be offered, is formally specified in Section 3. We first modify the aforementioned example to establish an upper bound on the competitive ratio.

2.4. Upper Bound on Competitive Ratio

We use the example from Section 2.3 to generate an upper bound on the competitive ratio by choosing the worst-case probability distribution between the two scenarios, of type 2 customers arriving versus not arriving.

Example 2 (Upper Bound Construction). Let C denote the constant $\frac{1}{w}$, where w is the unique real number satisfying $we^w = 1$. Consider the following randomized arrival sequence for the instance defined in Example 1:

- Arrivals are $T = N, i_1 = \cdots = i_N = 1$ w.p. $1 - \frac{C}{M}$.
- Arrivals are $T = 2N, i_1 = \cdots = i_N = 1, i_{N+1} = \cdots = i_{2N} = 2$ w.p. $\frac{C}{M}$.

Our construction is similar in spirit to that of Mehta and Panigrahi [35], who also exploit the fact that the optimum can use a fractional LP solution. This is used in the following theorem.

Theorem 2. *Consider the randomized instance defined in Example 2, and let $M, N \to \infty$. The expected value of* OPT $(i_1, \ldots, i_T)$ *is at least $C + 1$. On the other hand, for any online algorithm, the expected value of* ALG$(i_1, \ldots, i_T)$ *is at most $C - \ln C$. By Yao's minimax principle (see Yao [43] or Krumke [27, theorem 1.12]), the competitive ratio of any online algorithm cannot exceed*

$$\frac{C - \ln C}{C + 1} \approx 0.43.$$

The value of $C = \frac{1}{w} = e^w \approx 1.76$ has been chosen to minimize the competitive ratio. We defer the calculations required to prove Theorem 2 to Appendix A.

3. Algorithm

In this section we show how to formally specify our algorithm, which

1. maximizes the expected revenue of the assortment offered to each customer and
2. is subject to the constraint that the expected units of any item sold as an add-on never exceed half of its starting inventory, which we will call its *add-on threshold*.

We divide the specification of the algorithm into the following procedures.

- Constructing the optimal assortment given the appropriate *instructions* of which add-ons cannot be offered (Section 3.1)
- Constructing the instructions given accurate estimates of expected add-on sales (Section 3.2)
- Constructing the estimates via sampling (Section 3.3)
- Presenting the combined algorithm (Section 3.4)

Overall, our algorithm abstracts each time step into a continuous interval and envisions time as passing "fluidly" through it, as described in the introduction in Section 1.4.

3.1. Offering Add-on Assortments

In this section we describe the instructions that enforce the add-on thresholds and how to follow them when offering assortments. The instructions come as a sequence of *protection lists* $\mathcal{L}^1, \dots, \mathcal{L}^T$, where $\mathcal{L}^t$ contains the items that hit their add-on thresholds at time t. A *forbidden set* then accumulates all of the items that have appeared in a protection list and hence can no longer be offered as add-ons. We formally define these in their full generality in the following and then provide examples.

Definition 4 (Protection List and Forbidden Set). For each time period $t = 1, \dots, T$, let $\mathcal{L}^t$ be a list of tuples of the form "(item, probability)"; $\mathcal{L}^t$ is called a *protection list*. Formally,

$$\mathcal{L}^t = ((j_k^t, \rho_k^t) : k \in [K_t]),$$

where $K_t \geq 0$ is its length, $j_k^t \in [n]$, and $\rho_k^t \in [0,1]$.

For all $t \in [T]$, $\mathcal{L}^1, \dots, \mathcal{L}^t$, and $k = 0, \dots, K_t$, let $\mathcal{F}_k^t$ be a set of items, called a *forbidden set*. We have that $\mathcal{F}_k^t$ is a function of $\mathcal{L}^1, \dots, \mathcal{L}^t$ and is defined as follows:

$$\mathcal{F}_k^t(\mathcal{L}^1, \dots, \mathcal{L}^t) := \left(\bigcup_{s<t} \{j_{k'}^s : k' \in [K_s]\} \right) \cup \{j_{k'}^t : k' \leq k\}.$$

In Definition 4, $j_1^t, \dots, j_{K_t}^t$ is the sequence of items that hit their add-on thresholds at time t. It is necessary to allow for multiple items to hit their thresholds during the same time step t, because after the first item hits its add-on threshold, the revenue-maximizing assortment that would be offered can completely change, causing another item to also hit its add-on threshold. A running example that demonstrates this complexity is relegated to Appendix B.

We now provide a simple example of how protection lists work. Suppose that there is one unit of item 1, which is the only item eligible to be offered as an add-on to the first customer, who would purchase it as an add-on with probability 3/4. To prevent item 1 from violating its add-on threshold, list $\mathcal{L}^1 = ((\rho, 2/3))$, which tells the algorithm to only offer it as an add-on with probability 2/3. With probability 1/3, item 1 joins the forbidden set $\mathcal{F}_1^1(\mathcal{L}^1)$ without having ever been offered as an add-on. This ensures that the expected number of units of item 1 sold as an add-on is $\frac{2}{3} \cdot \frac{3}{4} = \frac{1}{2}$, which will never increase again.

We formalize how our algorithm would read a general protection list $\mathcal{L}^t$ to make a randomized offer to customer t in Procedure 1. Note that it also requires lists $\mathcal{L}^1, \dots, \mathcal{L}^{t-1}$ as input. Furthermore, we let $I_1, \dots, I_n$ denote the respective remaining inventory levels of the n items, which we refer to in Procedure 1 to ensure that stocked-out items are not offered.

Procedure 1

Input: lists $\mathcal{L}^1, \dots, \mathcal{L}^t$;
Output: randomized assortment to offer customer t

if $I_{i_t} > 0$ **then**
  $J \leftarrow \{j \neq i_t : I_j = 0\}$ (*these items have stocked out*)
  $k \leftarrow 0$
  while $k < K_t$ **do**
    flip a coin that is heads with probability ρ_{k+1}^t
    if heads **then break** out of **while** loop

7: $k \leftarrow k+1$
8: **end while**
9: $F = \mathcal{F}_k^t(\mathcal{L}^1, \ldots, \mathcal{L}^t) \setminus \{i_t\}$ (*these items are forbidden, based on the randomly chosen index k*)
10: set $w_j = 0$ if $j \in J \cup F$, and $w_j = r_j^{\text{disc}}$ otherwise

$$\textbf{return } \arg\max_{\tilde{A} \in \mathcal{A}_{i_t}} \sum_{j \in \tilde{A}} w_j q_{i_t j}(\tilde{A}) \tag{7}$$

11: **else**
12: **return** $\emptyset$
13: **end if**

We clarify why the assortment $\tilde{A}$ returned in step 10 is valid. First, the problem is computationally tractable as a result of the oracle assumed in Definition 1. Second, $\tilde{A}$ can be assumed to not contain any add-ons $j \in J$ that are out of stock. This is because if it did, then we could remove all such j's from $\tilde{A}$ and not remove any positive terms from the sum (7) (because $w_j = 0$ for all $j \in J$) and yet increase $q_{i_t j'}(\tilde{A})$ for add-ons j' not removed (by the substitutability assumption from Definition 1). Similarly, $\tilde{A}$ does not need to contain any $j \in F$. We assume a consistent tiebreaking for selecting $\tilde{A}$, where $\tilde{A} \cap (J \cup F) = \emptyset$ is enforced.

3.2. Constructing Protection Lists

In Section 3.1, we specified how to follow the instructions contained in the protection lists. Now we specify how to construct the lists $\mathcal{L}^1, \ldots, \mathcal{L}^T$ in sequence so that an item is no longer offered as an add-on after it hits its add-on threshold.

We describe how our procedure constructs protection list $\mathcal{L}^t$ given lists $\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}$. It starts with $\mathcal{L}^t$ as the empty list () and considers what would happen if Procedure 1, given these lists $\mathcal{L}^1, \ldots, \mathcal{L}^t$, was used to offer randomized assortments to customers $1, \ldots, t$. If no add-on thresholds would be violated after time t, then $\mathcal{L}^t = ()$ is returned. Otherwise, some item must have crossed its add-on threshold *at* time t, assuming lists $\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}$ were properly defined. Our procedure then finds a "most violating" item, ℓ, and appends instruction (ℓ, ρ) to list $\mathcal{L}^t$, where ρ is the constant telling Procedure 1 to only offer the originally planned assortments with probability ρ and add ℓ to the forbidden set otherwise. This ensures that ℓ does not violate its add-on threshold. However, other items could still violate their add-on thresholds at time t, so Procedure 1 needs to be rerun with the given lists $\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}$ and the newly augmented list $\mathcal{L}^t$. This process is repeated until all items that need to be protected have been added to $\mathcal{L}^t$.

We now introduce some notation. We emphasize that for the purpose of constructing protection list $\mathcal{L}^t$, the realized inventory levels at time t are irrelevant; all quantities are based on the expected add-on sales over all sample paths of what could have been purchased by customers $1, \ldots, t-1$.

Definition 5. Fix a time $t \in [T]$ at which point arrival types $i_1, \ldots, i_t$ have been revealed. For a given sequence of protection lists $\mathcal{L}^1, \ldots, \mathcal{L}^t$, suppose that Procedure 1 is used to offer randomized assortments to customers $1, \ldots, t$. Define the following for all items $j \in [n]$:

- Let $d_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t) \in [0, b_j]$ denote the expected units of j sold as an add-on to customers $1, \ldots, t$, *not counting* any units sold to customer t if the assortment offered to her by Procedure 1 was determined with index k equal to K_t.
- Let $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t) \in [0, 1]$ denote the expected units of j sold as an add-on to customer t, *only counting* units sold when the assortment offered to her by Procedure 1 was determined with $k = K_t$.

An add-on threshold is violated if for some item j,

$$d_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t) + h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t) > \frac{b_j}{2},$$

in which case the quantities from Definition 5 will allow us to compute the most violating item ℓ and the probability ρ with which we need to scale back its offering. However, there is one final challenge: the quantities from Definition 5 are not computationally tractable, because there could be exponentially many sample paths. To compensate, our algorithm stores the values of $d_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t)$ and increments them using the values of $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t)$, which are estimated via sampling.

We are finally ready to specify our procedure that constructs list $\mathcal{L}^t$, given $\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}$, in Procedure 2. The algorithm's internal variable for $d_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t)$ based on the current time and lists is denoted by $d_j^{\texttt{curr}}$ for all items $j \in [n]$. Meanwhile, ε is a small additive error assumed on the estimates of $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t)$; later we will see that this is possible and sufficient, but for now, it may aid comprehension to assume that $\varepsilon = 0$.

Procedure 2

Input: protection lists $\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}$, sampling error ε;
Output: protection list $\mathcal{L}^t$

1: $\mathcal{L} \leftarrow (), K = 0$
2: **loop**
3: $G \leftarrow [n] \setminus (\mathcal{F}_K^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}) \cup \{i_t\})$ (*these items are not currently forbidden as add-ons*)
4: $\hat{h}_j \leftarrow$ a sampled estimate of $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L})$ for all $j \in G$
5: **if** $d_j^{\mathtt{curr}} + \hat{h}_j \le \frac{b_j}{2} - \varepsilon$ for all $j \in G$ **then**
6: $d_j^{\mathtt{curr}} \leftarrow d_j^{\mathtt{curr}} + \hat{h}_j$ for all $j \in G$
7: **return** $(\mathcal{L}, d_1^{\mathtt{curr}}, \ldots, d_n^{\mathtt{curr}})$
8: **end if**
9: $\ell \leftarrow \arg\min_{j \in G} \frac{\left(\frac{b_j}{2} - \varepsilon\right) - d_j^{\mathtt{curr}}}{\hat{h}_j}$; $\rho \leftarrow \frac{\left(\frac{b_\ell}{2} - \varepsilon\right) - d_\ell^{\mathtt{curr}}}{\hat{h}_\ell}$
10: append (ℓ, ρ) to list $\mathcal{L}$ and increment K by 1
11: $d_j^{\mathtt{curr}} \leftarrow d_j^{\mathtt{curr}} + \rho \cdot \hat{h}_j$ for all $j \in G$
12: **end loop**

Now we establish some basic properties about Procedure 2; their proofs are relegated to Appendix A. The first proposition states that the procedure "correctly" updates its internal variables $d_j^{\mathtt{curr}}$ and that any errors as a result of sampling accumulate additively. The second proposition states that the protection lists $\mathcal{L}^t$ constructed indeed do not cause any add-on thresholds to be violated. The third proposition states that each item needs to be protected by at most one list.

Proposition 1. *Fix $\varepsilon_1, \varepsilon_2 \ge 0$ and consider an iteration of the loop in Procedure 2. Suppose that at the start of this iteration, the value of internal variable $d_j^{\mathtt{curr}}$ is within an additive error ε_1 of $d_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L})$ for all items j. Furthermore, suppose that all of the estimates $\hat{h}_j$ are within an additive error ε_2 of $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L})$. The following statements then hold:*

- *If step 6 is executed during this iteration, then for all $j \in G$, the updated value of $d_j^{\mathtt{curr}}$ satisfies*

$$|d_j^{\mathtt{curr}} - d_j^{t+1}(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}, ())| \le \varepsilon_1 + \varepsilon_2.$$

- *Otherwise, if step 11 is executed during this iteration, then for all $j \in G$, the updated value of $d_j^{\mathtt{curr}}$ (along with the updated value of $\mathcal{L}$) satisfies*

$$|d_j^{\mathtt{curr}} - d_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L})| \le \varepsilon_1 + \varepsilon_2.$$

Proposition 2. *Consider an iteration of the loop in Procedure 2. Suppose for all $j \in [n]$, $d_j^{\mathtt{curr}}$ does not violate its add-on threshold; that is, $d_j^{\mathtt{curr}} \le \frac{b_j}{2} - \varepsilon$. Then for these $j \in G$, the updated value for $d_j^{\mathtt{curr}}$ is still at most $\frac{b_j}{2} - \varepsilon$.*

Proposition 3. *Any item $j \in [n]$ can get added to the protection lists at most once. That is, in step 9 of Procedure 2, ℓ does not appear in $\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}$. Furthermore, ℓ is never again offered as an add-on, and $d_j^{\mathtt{curr}}$ is not increased.*

3.3. Estimating Add-on Sales via Sampling

In Section 3.2, we assumed that the values of $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t)$ could be accurately estimated. By Definition 5, $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t)$ is concerned with the events of add-ons being sold at time t. Note that in this probability space, the protection lists $\mathcal{L}^1, \ldots, \mathcal{L}^t$ are fixed, and the randomness is in the assortments offered by Procedure 1 and the purchase decisions of all past customers (the purchases of customers before t matter because they determine the inventories remaining at time t). Because we know the types $i_1, \ldots, i_t$ and the choice models associated, we can *simulate* both Procedure 1 and the purchase decisions of customers $1, \ldots, t$ from the start to draw samples of the add-ons sold at time t. Our Procedure 3 simply performs this sampling M independent times, where M is a large integer whose value is to be determined later.

Procedure 3

Input: lists $\mathcal{L}^1, \ldots, \mathcal{L}^t$, the number of samples M;
Output: estimates of $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t)$

1: $G \leftarrow [n] \setminus (\mathcal{F}_{K_t}^t(\mathcal{L}^1, \ldots, \mathcal{L}^t) \cup \{i_t\})$ (*these are the items not currently forbidden as add-ons, for which we need to compute $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^t)$*)
2: $C_j \leftarrow 0$ for all $j \in G$
3: **repeat**
4: Simulate a run to the end of time t

5: **if** the assortment offered to customer t by Procedure 1 was chosen in iteration $k = K_t$ **then**
6: $C_j \leftarrow C_j + 1$ for any $j \in G$ purchased by customer t
7: **end if**
8: **until** M runs have passed
9: **return** $\frac{C_j}{M}$ for all $j \in G$

The following is a standard bound on the additive sampling error of Procedure 3, based on the Chernoff–Hoeffding inequality, for which a proof can be found in Lugosi [31].

Lemma 2. *For any $j \in G$, $M \in \mathbb{N}$, and $\varepsilon_2 > 0$, the estimate $\frac{C_j}{M}$ of $h_j^t(\mathcal{L}^1, \dots, \mathcal{L}^t)$ returned by Procedure 3 satisfies*

$$\Pr\left[\left|\frac{C_j}{M} - h_j^t(\mathcal{L}^1, \dots, \mathcal{L}^t)\right| > \varepsilon_2\right] \le 2e^{-2\varepsilon_2^2 M}.$$

3.4. Combined Algorithm

Having derived the procedures in Sections 3.1–3.3, we are now ready to describe our combined algorithm. When a new customer t arrives and her type i_t is revealed, the algorithm first runs itself from the start a large number of times using the known types $i_1, \dots, i_t$ to estimate the expected add-on sales (Procedure 3). These estimates allow for a protection list $\mathcal{L}^t$ to be constructed for customer t (Procedure 2). Using this list, a randomized add-on assortment can be offered to customer t in a way which does not violate the add-on thresholds (Procedure 1). The purchase decision of customer t is then realized, and the algorithm moves on to customer $t + 1$. These cycles are depicted schematically in Figure 2.

Now we show that the combined algorithm accomplishes the goal outlined at the start of Section 3. The following theorem is proven in Appendix A.

Theorem 3. *Fix some small $\varepsilon > 0$ and consider our combined algorithm where Procedure 2 uses error parameter ε and Procedure 3 uses sample size $M = \frac{1}{2}\left(\frac{T+n}{\varepsilon}\right)^2 \ln\frac{2n(T+n)}{\varepsilon}$. For all items j, let $\overline{d}_j$ denote the expected units of j sold as an add-on at the end of the horizon, and let $\hat{d}_j$ denote the recorded value of d_j^{curr} at the end of the horizon.*

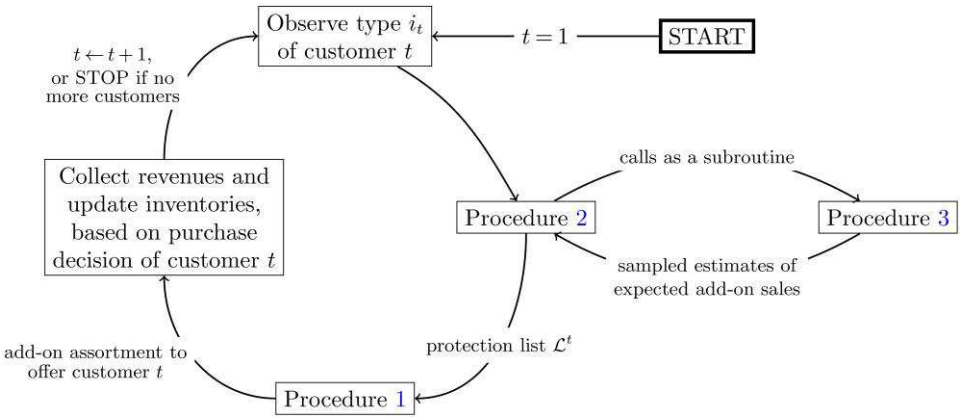
Then with probability at least $1 - \varepsilon$, the inequalities $\hat{d}_j - \varepsilon \le \overline{d}_j \le \frac{b_j}{2}$ hold for all items j. Furthermore, the combined algorithm terminates in runtime polynomial in T, n, and $\frac{1}{\varepsilon}$.

Note that the value of M is chosen so that the sampling error each time Procedure 3 is called is at most $\frac{\varepsilon}{T+n}$.

4. Analysis

In this section we prove Theorem 1, which lower bounds the competitive ratio of our combined algorithm by $1/4 - \varepsilon$ for any $\varepsilon > 0$. We reuse the definitions from the statement of Theorem 3: $\hat{d}_j$ is our algorithm's value of d_j^{curr} at the end of

Figure 2. Schematic of the combined algorithm.



the selling season, whereas $\overline{d}_j$ is the true expected units of item j sold as an add-on. Again, it may aid comprehension to first assume that $\varepsilon = 0$, in which case $\hat{d}_j = \overline{d}_j$ for all $j \in [n]$.

We provide a general road map of the proof:

1. We define each θ_j, the dual variable for the inventory constraint on item j, based on $\overline{d}_j$ and $\hat{d}_j$, providing shadow prices for each unit of starting inventory.
2. To ensure feasibility, we also need to have dual variables λ^t, which provide shadow prices for each customer t (Lemma 3).
3. We bound from above the value of the LP defined in Section 2.2 using weak duality (Lemma 4).
4. To bound the revenue our algorithm earns from selling add-ons to customer t, we need to bound the probability of customer t being served and the probabilities of add-ons not stocking out, which we accomplish given the protection levels in expectation (Lemma 5).
5. This allows us to obtain a lower bound on the expected revenue of our algorithm (Lemma 6).
6. Finally, using a variant of Markov's inequality (Proposition 5), we show that the lower bound from step 5 is at least $\frac{1}{4} - \varepsilon$ of the upper bound from step 3 and that the ε-error in sampling translates to only an ε-loss in revenue (Section 4.3).

4.1. Bounding OPT via Duality

We first introduce the following concepts that will aid our proof.

Definition 6. We define the following notation, for all $j \in [n]$:

- Let a_j denote the number of arriving customers of type j. That is, $a_j = |\{t \in [T] : i_t = j\}|$.
- Set d_j as follows:

$$d_j = \begin{cases} \max\{\overline{d}_j, \hat{d}_j + \varepsilon\} & \text{if } \hat{d}_j \geq \dfrac{1}{2} - \varepsilon; \\ \overline{d}_j & \text{otherwise.} \end{cases}$$

Note that if $\varepsilon = 0$, then $d_j = \overline{d}_j = \hat{d}_j$.

It simplifies the analysis to assume that $a_j \leq b_j$ for all $j \in [n]$. This can be justified by the fact that after b_j customers of a type j have already arrived, there are no units of item j remaining to serve subsequent customers of that type.

Definition 7. For each $i \in [n]$, introduce a fixed assortment S_i^* lying in

$$\arg\max_{S \in \mathcal{S}_i} \sum_{j \neq i} r_j^{\text{disc}} \left(1 - \frac{a_j + 2d_j}{b_j}\right) p_{ij}(S).$$

We can interpret S_i^* as an "ideal" assortment to offer customers of type i, given the a posteriori values of a_j and d_j. We assume that S_i^* does not contain any $j \neq i$ such that $a_j + 2d_j \geq b_j$. This is because if it did, then we could remove all such j's from S_i^* and not remove any positive terms from the above-mentioned sum and yet increase $p_{ij'}(S)$ for add-ons j' that were not removed (by the substitutability assumption from Definition 1).

With this assumption, $d_j < \frac{b_j}{2}$ for all $j \in S_i^* \setminus \{i\}$. By definition, either $d_j \geq \hat{d}_j + \varepsilon$, which would imply $\hat{d}_j < \frac{b_j}{2} - \varepsilon$, or $\hat{d}_j < \frac{1}{2} - \varepsilon$. In both cases, the conclusion is that $\hat{d}_j < \frac{b_j}{2} - \varepsilon$, because b_j is always at least 1. This means that at any point in our algorithm, the value of d_j^{curr} never reached the threshold of $\frac{b_j}{2} - \varepsilon$. As a result, items in $S_i^* \setminus \{i\}$ do not appear in any protection list and are never in the forbidden set for all $i \in [n]$.

Now we are ready to specify the bound on $\mathsf{OPT}(i_1, \ldots, i_T)$, which is the optimal objective value of the LP defined in Section 2.2. The dual of this LP can be written as follows, where $\{\theta_j : j \in [n]\}$ are the variables corresponding to the first n primal constraints and $\{\lambda^t : t \in [T]\}$ are the variables corresponding to the next T primal constraints:

$$\begin{aligned} \min & \sum_{j=1}^{n} b_j \theta_j + \sum_{t=1}^{T} \lambda^t \\ \text{s.t.} & \sum_{j=1}^{n} p_{i_t j}(S)\theta_j + \lambda^t \geq r_{i_t} + \sum_{j \neq i_t} r_j^{\text{disc}} p_{i_t j}(S) && t \in [T], S \in \mathcal{S}_{i_t}, \\ & \theta_j \geq 0 && j \in [n], \\ & \lambda^t \geq 0 && t \in [T]. \end{aligned}$$

We propose the following values for the dual variables:

$$\theta_j = \frac{r_j a_j + 2r_j^{\text{disc}} d_j}{b_j}, \qquad j \in [n],$$

$$\lambda^t = r_{i_t}\left(1 - \frac{a_{i_t}}{b_{i_t}}\right) + \sum_{j \neq i_t} r_j^{\text{disc}}\left(1 - \frac{a_j + 2d_j}{b_j}\right) p_{i_t j}(S^*_{i_t}), \quad t \in [T]. \tag{8}$$

Golrezaei et al. [24] were able to define dual variables that are sample-path dependent and prove feasibility on every sample path. We need to define our dual variables based on the d_j's, which can be seen as aggregate statistics over all the sample paths, in order to prove feasibility. Note that the value of λ^t is fully determined by the type of the customer, i_t.

Lemma 3 (Dual Feasibility). *The solution of the dual LP defined in* (8) *is feasible.*

Proof. We know that $a_{i_t} \le b_{i_t}$ for all $t \in [T]$, by assumption. Also, for $j \neq i_t$, $p_{i_t j}(S^*_{i_t})$ can only be nonzero for $j \in S^*_{i_t}$, which implies $a_j + 2d_j \le b_j$. Therefore, $\lambda^t \ge 0$. It is also easy to see that $\theta_j \ge 0$ for all $j \in [n]$.

The remaining constraints can be rearranged as

$$\lambda^t \ge r_{i_t} - \theta_{i_t} + \sum_{j \neq i_t} p_{i_t j}(S)(r_j^{\text{disc}} - \theta_j), \tag{9}$$

where we have used the fact that $p_{ii}(S) = 1$ for all i and $S \in \mathcal{S}_i$.

Take an arbitrary $t \in [T]$ and $S \in \mathcal{S}_{i_t}$. By our definition of the dual variables, $\theta_j \ge r_j^{\text{disc}} \cdot \frac{a_j + 2d_j}{b_j}$ for all $j \neq i_t$, because $r_j \ge r_j^{\text{disc}}$. Also, it is immediate that $\theta_{i_t} \ge r_{i_t} \cdot \frac{a_{i_t}}{b_{i_t}}$. Therefore, the right-hand side of (9) is at most

$$r_{i_t}\left(1 - \frac{a_{i_t}}{b_{i_t}}\right) + \sum_{j \neq i_t} r_j^{\text{disc}}\left(1 - \frac{a_j + 2d_j}{b_j}\right) p_{i_t j}(S).$$

Now, $S^*_{i_t}$ was specifically chosen to maximize the second part of the preceding expression among all $S \in \mathcal{S}_{i_t}$. Therefore, the preceding expression is no greater than our definition of λ^t, completing the proof of dual feasibility. □

Lemma 4 (Dual Objective Value). *The objective value of the dual solution in* (8) *is at most*

$$\sum_{i=1}^{n} 2r_i a_i + \sum_{j=1}^{n} 2r_j^{\text{disc}} d_j + \sum_{i=1}^{n} a_i \sum_{j \neq i} r_j^{\text{disc}}\left(1 - \frac{a_j + 2d_j}{b_j}\right) p_{ij}(S^*_i).$$

Proof. Substituting our values of θ_j and λ^t into the objective function of the dual LP, we get

$$\begin{aligned}\sum_{j=1}^{n} b_j \theta_j + \sum_{t=1}^{T} \lambda^t &= \sum_{j=1}^{n} (r_j a_j + 2r_j^{\text{disc}} d_j) + \sum_{t=1}^{T}\left(r_{i_t}\left(1 - \frac{a_{i_t}}{b_{i_t}}\right) + \sum_{j \neq i_t} r_j^{\text{disc}}\left(1 - \frac{a_j + 2d_j}{b_j}\right) p_{i_t j}(S^*_{i_t})\right) \\ &= \sum_{j=1}^{n} r_j a_j + \sum_{j=1}^{n} 2r_j^{\text{disc}} d_j + \sum_{i=1}^{n} \sum_{t: i_t = i}\left(r_i\left(1 - \frac{a_i}{b_i}\right) + \sum_{j \neq i} r_j^{\text{disc}}\left(1 - \frac{a_j + 2d_j}{b_j}\right) p_{ij}(S^*_i)\right).\end{aligned}$$

Noting the fact a_i is defined to be the cardinality of the set $\{t \in [T] : i_t = i\}$ and rearranging the terms, we have

$$\sum_{j=1}^{n} b_j \theta_j + \sum_{t=1}^{T} \lambda^t \le \sum_{i=1}^{n} 2r_i a_i + \sum_{j=1}^{n} 2r_j^{\text{disc}} d_j + \sum_{i=1}^{n} a_i \sum_{j \neq i} r_j^{\text{disc}}\left(1 - \frac{a_j + 2d_j}{b_j}\right) p_{ij}(S^*_i),$$

which completes the proof. □

Combining Lemmas 3 and 4 and applying weak duality, we have established in this subsection that the expression from Lemma 4 is an upper bound on $\mathsf{OPT}(i_1, \ldots, i_T)$.

4.2. Bounding the Algorithm's Revenue

We first introduce some notation and an auxiliary proposition.

Definition 8. For a run of our algorithm, define the following random variables:

- For all $t \in [T]$ and $j \in [n]$, let I_j^t denote the inventory of item j remaining at the *end* of time period t. It is understood that $I_j^0 = b_j$, its starting inventory, for all items $j \in [n]$.
- For all $t \in [T]$, let $S^t \in \mathcal{S}_{i_t} \cup \{\emptyset\}$ denote the assortment offered at time t.
- For all $t \in [T]$ and $j \in [n]$, let P_j^t be the indicator random variable for whether customer t bought item j; P_j^t can only potentially be 1 for $j \in S^t$.
- For all $j \in [n]$, let D_j denote the units of item j sold as an add-on by the end of the selling season. Note that $D_j = \sum_{t:i_t \neq j} P_j^t$, and $\mathbb{E}[D_j] = \overline{d}_j$.
- For all $t \in [T]$, let V^t denote the items in $S_{i_t}^*$ with inventory available at time t. That is, $V^t = \{j \in S_{i_t}^* : I_j^{t-1} > 0\}$.

The following proposition is immediate.

Proposition 4. *For all $j \in [n]$, the units of item j sold at full price by the end of the selling season is* $\min\{b_j - D_j, a_j\}$.

Proof. For any $j \in [n]$, consider a single sample path with a fixed value for D_j. The units of item j remaining to sell at full price are $b_j - D_j$. Because our algorithm offers item j to every customer requesting it as long as inventory is available, the number of customers served is $\min\{b_j - D_j, a_j\}$. □

Now we analyze the revenue earned by our algorithm in expectation and compare it with Lemma 4. The following lemma is crucial in bounding the revenue earned from selling add-ons.

Lemma 5 (Bound on Add-on Revenue). *For all $t \in [T]$, the revenue earned by our algorithm from selling add-ons during time t, $\sum_{j \neq i_t} r_j^{\text{disc}} P_j^t$, is in expectation at least*

$$\sum_{j \in S_{i_t}^*, j \neq i_t} r_j^{\text{disc}} p_{i_t j}(S_{i_t}^*) \left(\Pr[I_{i_t}^{t-1} > 0] - \frac{\overline{d}_j}{b_j - a_j} \right).$$

(Recall that for all $j \in S_{i_t}^* \setminus \{i_t\}$, we have $a_j + 2d_j < b_j$, which implies $b_j - a_j \neq 0$.)

Proof. Fix an arbitrary $t \in [T]$. We have

$$\begin{aligned} \mathbb{E}\left[\sum_{j \neq i_t} r_j^{\text{disc}} P_j^t\right] &= \mathbb{E}\left[\sum_{j \neq i_t} r_j^{\text{disc}} p_{i_t j}(S^t)\right] \\ &= \mathbb{E}\left[\sum_{j \neq i_t} r_j^{\text{disc}} p_{i_t j}(S^t) \,\middle|\, I_{i_t}^{t-1} > 0\right] \Pr[I_{i_t}^{t-1} > 0]. \end{aligned} \tag{10}$$

Both equalities use the law of total expectation. The first equality holds because $\mathbb{E}[\sum_{j \neq i_t} r_j^{\text{disc}} P_j^t] = \mathbb{E}_{S^t}[\mathbb{E}[\sum_{j \neq i_t} r_j^{\text{disc}} P_j^t | S^t]]$, and the inner expectation is equal to $\sum_{j \neq i_t} r_j^{\text{disc}} p_{i_t j}(S^t)$, because for a fixed S^t, P_j^t is an independent binary random variable that is equal to 1 with probability $p_{i_t j}(S^t)$. The second equality holds because if $I_{i_t}^{t-1} = 0$, then $S^t = \emptyset$, and $p_{i_t j}(\emptyset) = 0$ for all j.

Now, we claim that on every sample path where $I_{i_t}^{t-1} > 0$, the following holds:

$$\begin{aligned} \sum_{j \neq i_t} r_j^{\text{disc}} p_{i_t j}(S^t) &\geq \sum_{j \neq i_t} r_j^{\text{disc}} p_{i_t j}(V^t) \\ &= \sum_{j \in V^t, j \neq i_t} r_j^{\text{disc}} p_{i_t j}(V^t) \\ &\geq \sum_{j \in V^t, j \neq i_t} r_j^{\text{disc}} p_{i_t j}(S_{i_t}^*). \end{aligned}$$

The first inequality is true because, conditioned on $I_{i_t}^{t-1} > 0$, V^t was a feasible choice of assortment during the maximization step (step 10) of Procedure 1 (add-ons in V^t are never in the forbidden set and have inventory

available). The second inequality is true because, conditioned on $I_{i_t}^{t-1} > 0$, V^t is always a subset of $S_{i_t}^*$ containing i_t, and by the substitutability assumption, $p_{i_t j}(V^t) \geq p_{i_t j}(S_{i_t}^*)$ for all $j \in V^t$. Thus (10) is at least

$$\begin{aligned}
&\mathbb{E}\left[\sum_{j\in V^t, j\neq i_t} r_j^{\text{disc}} p_{i_t j}(S_{i_t}^*) \,\middle|\, I_{i_t}^{t-1} > 0\right] \Pr[I_{i_t}^{t-1} > 0] \\
&= \sum_{j\in S_{i_t}^*, j\neq i_t} r_j^{\text{disc}} p_{i_t j}(S_{i_t}^*) \Pr[I_j^{t-1} > 0 \,|\, I_{i_t}^{t-1} > 0] \Pr[I_{i_t}^{t-1} > 0] \\
&= \sum_{j\in S_{i_t}^*, j\neq i_t} r_j^{\text{disc}} p_{i_t j}(S_{i_t}^*) \Pr[(I_j^{t-1} > 0) \cap (I_{i_t}^{t-1} > 0)] \\
&\geq \sum_{j\in S_{i_t}^*, j\neq i_t} r_j^{\text{disc}} p_{i_t j}(S_{i_t}^*) (\Pr[I_{i_t}^{t-1} > 0] - \Pr[I_j^{t-1} = 0]),
\end{aligned}$$

where in the first equality we have used the definition that V^t is the subset of $S_{i_t}^*$ with inventory remaining.

Finally, $I_j^{t-1} = 0$ implies $I_j^T = 0$. By Proposition 4, this event is equivalent to $D_j + \min\{b_j - D_j, a_j\} = b_j$, which, in turn, is equivalent to $\min\{b_j - a_j, D_j\} = b_j - a_j$, or $D_j \geq b_j - a_j$. By Markov's inequality, this is at most $\frac{\mathbb{E}[D_j]}{b_j - a_j} = \frac{\overline{d}_j}{b_j - a_j}$, completing the proof of Lemma 5. □

Lemma 6 (Algorithm's Expected Revenue). *The expected revenue of our algorithm is at least*

$$\sum_{i=1}^n r_i \cdot \mathbb{E}[\min\{b_i - D_i, a_i\}] + \frac{1}{2}\sum_{j=1}^n r_j^{\text{disc}} \overline{d}_j + \frac{1}{2}\sum_{i=1}^n \sum_{j\neq i} r_j^{\text{disc}} \left(\mathbb{E}[\min\{b_i - D_i, a_i\}] - \frac{a_i}{2} \cdot \frac{a_j + 2d_j}{b_j}\right) p_{ij}(S_i^*).$$

Proof. Recall that for every customer $t \in [T]$, our algorithm serves her as long as item i_t is available. Therefore, the revenue earned on a run of our algorithm is

$$\sum_{t=1}^T \left(r_{i_t} \cdot \mathbf{1}(I_{i_t}^{t-1} > 0) + \sum_{j\neq i_t} r_j^{\text{disc}} P_j^t\right).$$

Its expected value is equal to

$$\mathbb{E}\left[\sum_{t=1}^T r_{i_t} \cdot \mathbf{1}(I_{i_t}^{t-1} > 0)\right] + \mathbb{E}\left[\sum_{t=1}^T \sum_{j\neq i_t} r_j^{\text{disc}} P_j^t\right].$$

The first term is the expected revenue earned from selling items i to customers requesting them (at the full price of r_i), whereas the second term is the expected revenue earned from selling items $j \neq i_t$ as add-ons (at the discounted price of r_j^{disc}) summed over all time periods t.

Applying Proposition 4, the first term is equal to

$$\sum_{i=1}^n r_i \cdot \mathbb{E}\left[\sum_{t:i_t=i} \mathbf{1}(I_i^{t-1} > 0)\right] = \sum_{i=1}^n r_i \cdot \mathbb{E}[\min\{b_i - D_i, a_i\}]. \tag{11}$$

Applying the definitions that $D_j = \sum_{t:i_t\neq j} P_j^t$ and $\mathbb{E}[D_j] = \overline{d}_j$, the second term is equal to

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^T \sum_{j\neq i_t} r_j^{\text{disc}} P_j^t\right] &= \sum_{j=1}^n r_j^{\text{disc}} \cdot \mathbb{E}\left[\sum_{t:i_t\neq j} P_j^t\right] \\
&= \sum_{j=1}^n r_j^{\text{disc}} \overline{d}_j.
\end{aligned} \tag{12}$$

We also analyze the second term in a different way. It can be rearranged as follows:

$$\mathbb{E}\left[\sum_{t=1}^T \sum_{j\neq i_t} r_j^{\text{disc}} P_j^t\right] = \sum_{i=1}^n \sum_{t:i_t=i} \mathbb{E}\left[\sum_{j\neq i} r_j^{\text{disc}} P_j^t\right]. \tag{13}$$

Fix an arbitrary i and apply Lemma 5 for all t such that $i_t = i$. We obtain

$$\begin{aligned}\sum_{t:i_t=i}\mathbb{E}\left[\sum_{j\neq i} r_j^{\text{disc}} P_j^t\right] &\geq \sum_{t:i_t=i}\sum_{j\in S_i^*, j\neq i} r_j^{\text{disc}} p_{ij}(S_i^*)\left(\Pr[I_i^{t-1}>0] - \frac{\overline{d}_j}{b_j - a_j}\right)\\ &= \sum_{j\in S_i^*, j\neq i} r_j^{\text{disc}} p_{ij}(S_i^*)\left(\sum_{t:i_t=i}\Pr[I_i^{t-1}>0] - a_i\cdot\frac{\overline{d}_j}{b_j-a_j}\right)\\ &= \sum_{j\in S_i^*, j\neq i} r_j^{\text{disc}} p_{ij}(S_i^*)\left(\mathbb{E}[\min\{b_i - D_i, a_i\}] - \frac{a_i}{2}\cdot\frac{2\overline{d}_j}{b_j-a_j}\right),\end{aligned}$$

where the final equality follows after the same derivation as (11). Now, by definition, $\overline{d}_j \leq d_j$. Therefore, we can replace $\overline{d}_j$ with d_j and the expression would be no greater. Furthermore, because $2d_j < b_j - a_j$ for all $j \in S_i^* \setminus \{i\}$, $\frac{2d_j}{b_j-a_j} \leq \frac{2d_j+a_j}{b_j}$. Substituting back into (13), we conclude that

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{j\neq i_t} r_j^{\text{disc}} P_j^t\right] \geq \sum_{i=1}^{n}\sum_{j\in S_i^*, j\neq i} r_j^{\text{disc}}\left(\mathbb{E}[\min\{b_i - D_i, a_i\}] - \frac{a_i}{2}\cdot\frac{2d_j + a_j}{b_j}\right) p_{ij}(S_i^*). \tag{14}$$

Taking (11) $+\frac{1}{2}\cdot$ (12) $+\frac{1}{2}\cdot$ (14) completes the proof of Lemma 6. □

4.3. Proof of Theorem 1

In this subsection we aim to prove our main result Theorem 1 by using Theorem 3 to bound the expression from Lemma 4 relative to the expression from Lemma 6. Both expressions can be seen as a sum of three terms, and we compare the respective terms separately. Namely, we compare

$$\sum_{i=1}^{n} r_i\cdot\mathbb{E}[\min\{b_i - D_i, a_i\}] \text{ to } \sum_{i=1}^{n} 2r_i a_i,$$

$$\frac{1}{2}\sum_{j=1}^{n} r_j^{\text{disc}}\overline{d}_j \text{ to } \sum_{j=1}^{n} 2r_j^{\text{disc}} d_j,$$

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i} r_j^{\text{disc}}\left(\mathbb{E}[\min\{b_i - D_i, a_i\}] - \frac{a_i}{2}\cdot\frac{a_j + 2d_j}{b_j}\right) p_{ij}(S_i^*) \text{ to } \sum_{i=1}^{n} a_i\sum_{j\neq i} r_j^{\text{disc}}\left(1 - \frac{a_j+2d_j}{b_j}\right) p_{ij}(S_i^*).$$

Fix a small ε. We will condition on our sampling algorithm not failing (i.e., $\hat{d}_j - \varepsilon \leq \overline{d}_j \leq \frac{b_j}{2}$ for all $j \in [n]$), which occurs with probability at least $1-\varepsilon$. We will scale our total revenue by $(1-\varepsilon)$ at the end and treat a failed run as a run with 0 revenue.

To bound the value $\min\{b_j - D_j, a_j\}$, we use the following modification of Markov's inequality.

Proposition 5. *For some $b \geq 0$, let X be a random variable distributed over $[0, b]$ with $\mathbb{E}[X] \geq \frac{b}{2}$. Then for any a such that $0 \leq a \leq b$, $\mathbb{E}[\min\{X, a\}] \geq \frac{a}{2}$.*

Proof. Consider the random variable $X - \min\{X, a\}$. When $X \geq a$, its value is at most $b - a$, because $X \leq b$. When $X < a$, its value is 0. Therefore, $\mathbb{E}[X - \min\{X, a\}] \leq (b-a)\Pr[X \geq a]$, which in conjunction with $\mathbb{E}[X] \geq \frac{b}{2}$ implies

$$\mathbb{E}[\min\{X, a\}] \geq \frac{b}{2} - (b-a)\Pr[X \geq a]. \tag{15}$$

If $\Pr[X \geq a] \leq \frac{1}{2}$, then (15) implies $\mathbb{E}[\min\{X, a\}] \geq \frac{b}{2} - \frac{b-a}{2} = \frac{a}{2}$, as desired.

Otherwise, if $\Pr[X \geq a] > \frac{1}{2}$, we can immediately conclude from the nonnegativity of X and a that $\mathbb{E}[\min\{X, a\}] \geq a\cdot\Pr[X \geq a]$, which combined with $\Pr[X \geq a] > \frac{1}{2}$ yields the desired result. □

Now, to compare the first terms, consider any $i \in [n]$. We have $\mathbb{E}[D_i] = \overline{d}_i \leq \frac{b_i}{2}$, which implies $\mathbb{E}[b_i - D_i] \geq \frac{b_i}{2}$. Therefore, we can apply Proposition 5 with $X = b_i - D_i$ to obtain $\mathbb{E}[\min\{b_i - D_i, a_i\}] \geq \frac{a_i}{2}$ for all $i \in [n]$. This establishes that

$$\sum_{i=1}^{n} r_i\cdot\mathbb{E}[\min\{b_i - D_i, a_i\}] \geq \frac{1}{4}\sum_{i=1}^{n} 2r_i a_i. \tag{16}$$

To compare the second terms, consider any $j \in [n]$. If $d_j \neq \overline{d}_j$, then $d_j = \hat{d}_j + \varepsilon$ and $\hat{d}_j \geq \frac{1}{2} - \varepsilon$. Also, $\overline{d}_j \geq \hat{d}_j - \varepsilon$. Therefore, for these j, $\frac{\overline{d}_j}{2d_j} \geq \frac{\hat{d}_j - \varepsilon}{4\hat{d}_j + 4\varepsilon} = \frac{1}{4} - \frac{2\varepsilon}{4\hat{d}_j + 4\varepsilon} \geq \frac{1}{4} - \frac{2\varepsilon}{4(1/2-\varepsilon)+4\varepsilon} = \frac{1}{4} - \varepsilon$. This establishes that

$$\frac{1}{2}\sum_{j=1}^{n} r_j^{\text{disc}} \overline{d}_j \geq \left(\frac{1}{4} - \varepsilon\right) \sum_{j=1}^{n} 2 r_j^{\text{disc}} d_j. \tag{17}$$

Finally, to compare the third terms, we can again apply Proposition 5 for all $i \in [n]$ to obtain

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i} r_j^{\text{disc}} \left(\mathbb{E}[\min\{b_i - D_i, a_i\}] - \frac{a_i}{2}\cdot\frac{a_j + 2d_j}{b_j}\right) p_{ij}(S_i^*) \geq \frac{1}{4}\sum_{i=1}^{n} a_i \sum_{j\neq i} r_j^{\text{disc}} \left(1 - \frac{a_j + 2d_j}{b_j}\right) p_{ij}(S_i^*). \tag{18}$$

Adding (16), (17), and (18), we conclude that with probability at least $1 - \varepsilon$, our algorithm earns revenue at least $\frac{1}{4} - \varepsilon$ of the dual objective. Therefore, using our combined algorithm, for any instance and sequence of arrivals, $\mathsf{ALG}(i_1, \ldots, i_T) \geq (1 - \varepsilon)\left(\frac{1}{4} - \varepsilon\right) \cdot \mathsf{OPT}(i_1, \ldots, i_T)$, completing the proof of Theorem 1.

Remark 1. Throughout our algorithm and analysis in Sections 3 and 4, we assumed the existence of an oracle that can solve the single-customer assortment optimization problem (2) *exactly*. However, given an oracle that can only solve (2) *approximately*-that is, return an assortment A' with

$$\sum_{j\in A'} w_j q_{ij}(A') \geq \gamma \cdot \max_{A\in\mathcal{A}_i} \sum_{j\in A} w_j q_{ij}(A)$$

for some $\gamma \leq 1$-our analysis still leads to a competitive ratio guarantee of $\gamma/4$. To see this, note that the only statement that used the optimality of the oracle is Lemma 5. With an approximate oracle instead, the revenue in Lemma 5 is scaled down by a factor of γ, which ultimately leads to the algorithm's total revenue in Lemma 6 being scaled down by the same factor γ. Therefore, the competitive ratio guarantee degrades by a factor of γ.

5. Analysis of the No-add-on Algorithm When Potential Add-on Revenue Is Small

Our main algorithm from Section 3 demonstrates the need to protect items from being sold as an add-on but only *partially*-namely, a 1/2-fraction in expectation should be protected. In this section we analyze the "no-add-on" algorithm, which *fully* protects items in that it does not recommend any add-ons and reserves all inventory to be sold as primary items. We prove that in the regime where the potential add-on revenue is low compared with the profit margins (prices) r_j of primary items-namely, $r_j^{\text{disc}}/r_i \leq R$ if there is any positive probability that item j is sold as an add-on to item i-this algorithm performs well.

Theorem 4. *Let R be a constant such that the expected revenue $\sum_{j\in A} r_j^{\text{disc}} q_{ij}(A)$ of any feasible add-on assortment $A \in \mathcal{A}_i$ to item i is upper bounded by $R \cdot r_i$ for all primary items $i \in [n]$. Specifically,*

$$\max_{i\in[n]} \max_{j: q_{ij}(A)>0 \text{ for some } A\in\mathcal{A}_i} \frac{r_j^{\text{disc}}}{r_i}$$

is a valid value for R, which denotes the maximum ratio between the revenues of an add-on item and a primary item for which it is sold as an add-on with positive probability. Then the online algorithm that never offers any add-ons has a competitive ratio at least $1/(1 + R)$.

It is easy to see that the guarantee of $1/(1 + R)$ for the no-add-on algorithm from Theorem 4 is *tight*, because without inventory constraints, the no-add-on algorithm would lose a factor of $1 + R$ at every time step. Therefore, our main algorithm from Section 3 would have a better worst-case performance as long as $R > 3$ and enjoys a big benefit in that its competitive ratio guarantee of 1/4 *does not depend on R*. We reiterate that in most circumstances, we would expect the value of R to be large, because the online platform can selectively recommend niche items j that have a much larger profit margin than the primary item i. Our numerical experiments in Section 6 also corroborate this, where the no-add-on algorithm does not perform well across the board.

Proof of Theorem 4. On any sequence of arrivals $(i_1, \ldots, i_T)$, let $a_j = |\{t \in [T] : i_t = j\}|$ denote the number of arriving customers of type j for all $j \in [n]$. We upper bound the value of $\mathsf{OPT}(i_1, \ldots, i_T)$ defined by LP (3) as follows. First, note that $i \in S$ for all $S \in \mathcal{S}_{i_t}$, with $p_{i_t i}(S) = 1$ (see Definition 2). Therefore, from the first set of LP constraints,

we can deduce that

$$b_j \geq \sum_{t=1}^{T} \sum_{S \in \mathcal{S}_{i_t}} p_{i_t j}(S) y^t(S) \geq \sum_{t: i_t = j} \sum_{S \in \mathcal{S}_{i_t}} y^t(S) \tag{19}$$

for all j. Consequently, the objective value of the LP can be upper bounded as

$$\begin{aligned} \sum_{t=1}^{T} \sum_{S \in \mathcal{S}_{i_t}} \left(r_{i_t} + \sum_{j \neq i_t} r_j^{\text{disc}} p_{i_t j}(S) \right) y^t(S) &= \sum_{i=1}^{n} \sum_{t: i_t = j} \sum_{S \in \mathcal{S}_{i_t}} y^t(S) \left(r_i + \sum_{j \neq i} r_j^{\text{disc}} q_{ij}(S \setminus i) \right) \\ &\leq \sum_{i=1}^{n} r_i (1+R) \sum_{t: i_t = j} \sum_{S \in \mathcal{S}_{i_t}} y^t(S) \\ &\leq \sum_{i=1}^{n} r_i (1+R) \min\{b_j, a_j\}, \end{aligned}$$

where the first inequality uses the definition of R, and the second inequality uses (19) along with the definition that $a_j = |\{t : i_t = j\}|$ and the LP constraint that $\sum_{S \in \mathcal{S}_{i_t}} y^t(S) \leq 1$ for all t such that $i_t = j$.

Meanwhile, the revenue earned by the no-add-on algorithm on arrival sequence $(i_1, \ldots, i_T)$ is $\sum_{j=1}^{n} r_j \min\{a_j, b_j\}$, because it only collects revenue for primary items, up to a maximum of b_j times for each j. Therefore, the competitive ratio of this online algorithm is at least $1/(1+R)$, completing the proof of Theorem 4. □

6. Numerical Experiments

In this section, we conduct numerical experiments to evaluate the performance of our algorithm. We focus on a particular choice model: at most one add-on can be recommended. We compare our algorithm with four other algorithms: the inventory-balancing policy from Golrezaei et al. [24] (denoted by "GNR") based on a penalty function; the protection-level policy from Ball and Queyranne [3] (denoted by "BQ"), which never offers as add-ons items with less than half of starting inventory remaining; the greedy algorithm (denoted by "Greedy"), which offers the revenue-maximizing add-on assortment each period without considering saving inventory; and the algorithm that never offers add-ons (denoted by "No add-on"). For GNR, we further consider two versions of the algorithm depending on different penalty functions: the linear (denoted by "GNR(L)") and exponential (denoted by "GNR(E)") penalty functions (see Golrezaei et al. [24]). Note that GNR(L), GNR(E), and Greedy are equivalent when the inventory level $b_j = 1$ for all j. In addition, when $b_j = 1$ for all j, we use the probabilistic version of BQ: for each item, we randomly choose a protection level of either 0 or 1 with equal probability.

When the number of products n is small, we are able to evaluate the conditional probabilities exactly (i.e., simulation is unnecessary). More specifically, in each time period, we can track the probability of each inventory state and the expected number of units of each item sold as an add-on. However, the number of states grows exponentially in n. When n is large, we have to use simulation to evaluate each algorithm (over the different realizations of customer decisions). Because we can exactly evaluate algorithms for relatively small problem instances, we compare the simulation results with exact results in those problem instances and find that the simulation results are quite accurate with 500 simulation replications. Therefore, we will use 500 simulation replications in each experiment. We also note that the difference among algorithms seems to be larger in small-sized problem instances than in large-sized problem instances. Hence many of our numerical experiments focus on small-sized problem instances. We choose a revenue vector $r = \{1, 2, \ldots, n\}$ and assume that $r_j^{\text{disc}} = \alpha r_j$ for each j. We also assume that the probability of buying each add-on equals a constant p and all items have the same initial inventory; that is, $b_j = b$ for each j. Then we can set up our experiments by adjusting the following parameters: the number of items n, number of customers T, initial inventory level for each item b, probability of buying each add-on p, and discount factor α. For each fixed (n, T, b, p, α) (called a group of experiments), we randomly generate an arrival sequence $n = 1{,}000$ times. When $b = 1$, we assume that each type of customer appears at most once; when $b > 1$, we generate each customer uniformly from $\{1, \ldots, n\}$. We test on 40 groups of experiments (i.e., 40,000 problem instances) in total. We evaluate each algorithm based on the ratio of the expected revenue generated from that algorithm to the optimal value of the LP.

In summary, our algorithm performs well. In particular, the worst-case ratio among all the 40,000 problem instances under our algorithm equals 0.60, which is much higher than the theoretic worst-case bound 0.25. In addition, it is higher than the worst-case ratio under all other algorithms (see Table 2). Moreover, we calculate the average ratio among the last β-quantile for each algorithm. For example, $\beta = 1$ corresponds to the average ratio among

Table 2. Worst-case and average ratios of algorithms.

Ratio type	Our algorithm	GNR(L)	GNR(E)	Greedy	BQ	No add-on
Worst-case ratio	0.60	0.34	0.34	0.34	0.31	0.15
Average ratio	0.89	0.91	0.91	0.89	0.89	0.72

all the 40,000 problem instances, and $\beta = 0.5$ corresponds to the average ratio among those 20,000 problem instances with the least ratios. We summarize the results in Figure 3. In particular, our algorithm outperforms others when $\beta \leq 0.5$; GNR(L) and GNR(E) eventually perform the best as β approaches 1.

In conclusion, although our algorithm is robust in the sense that it is designed to hedge against the worst-case instance and arrival sequence, our numerical experiments suggest that it is competitive with existing algorithms over average instances and arrival sequences while dominating in the worst cases.

7. Conclusions and Future Directions

Motivated by recommendation-at-checkout systems, we study a new online assortment planning problem. We develop a robust algorithm based on the new notion of a protection level in expectation and bound its competitive ratio under the worst-case arrival sequence. Our numerical experiments demonstrate that it is also highly competitive with existing benchmarks over average instances. One managerial insight from this work is that it can be important to withhold inventory in order to meet the primary demands of future customers, even if an add-on item is not being sold at a discount.

This work is a first attempt to address a setting where (i) the choice function for add-ons is based on the primary item at checkout, and the availability of the primary item determines whether an add-on assortment can be offered, and (ii) each item has a full price when sold as a primary item and a discounted price when sold as an add-on. Thus, our model is simplified in several aspects. Here are some future directions:

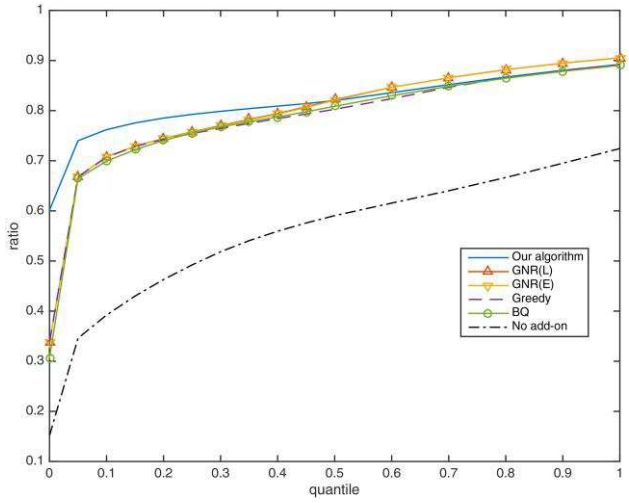
1. It would be interesting to incorporate the problem of learning into our work. Moreover, our model does not consider the case where the choice model could depend on both the primary item and the customer's profile (when such information is available).

2. It would be interesting to allow multiple primary items at checkout. Then, the choice function would depend on a subset of items. Analyzing the relationship between the competitive ratio and the number of primary items is a challenging problem.

3. We assume that when an item is offered as an add-on, it has a single, fixed discount price. However, the level of discount could depend on the primary item with which it is bundled. Such a model is an interesting direction for future work.

4. Along the same vein, trying to design an instance-dependent algorithm and get a bound based on the ratios between the discounts of the items could be interesting.

Figure 3. (Color online) Ratios of algorithms under different quantiles.



Acknowledgments

The authors thank Qihang Lin for early discussions. The authors thank the anonymous reviewers of *Mathematics of Operations Research* for many useful suggestions that have improved the paper.

Appendix A. Omitted Proofs

Proof of Lemma 1. Consider any algorithm, which could know $i_1, \dots, i_T$ at the start of the time horizon, and consider a sample path with that algorithm. For all $t \in [T]$ and $S \in \mathcal{S}_{i_t}$, let $Y^t(S)$ be the indicator random variable for assortment S being offered during time t. For all $t \in [T]$ and $j \in [n]$, let P_j^t be the indicator random variable for customer t buying item j.

Clearly, at most one assortment from $\mathcal{S}_{i_t}$ can be offered during each $t \in [T]$, so we have $\sum_{S \in \mathcal{S}_{i_t}} Y^t(S) \le 1$. Taking the expectation (with respect to both the randomness in the algorithm and the randomness in the customers' decisions) on both sides and letting $y^t(S) := \mathbb{E}[Y^t(S)]$, we have

$$\sum_{S \in \mathcal{S}_{i_t}} y^t(S) \le 1,$$

which shows that the second constraint in LP (3) is satisfied.

During time t, inventory of item $j \in [n]$ gets depleted if and only if $P_j^t = 1$, which is only possible if some $S \in \mathcal{S}_{i_t}$ is offered. Therefore, the depletion of item j during time t is equal to $\sum_{S \in \mathcal{S}_{i_t}} P_j^t \cdot Y^t(S)$. Conditioned on $Y^t(S) = 1$, P_j^t is an independent binary random variable that is 1 with probability $p_{i_t j}(S)$. Because the total depletion of item j over all the time periods cannot exceed b_j, we have

$$\sum_{t=1}^{T} \sum_{S \in \mathcal{S}_{i_t}} P_j^t \cdot Y^t(S) \le b_j$$

$$\Rightarrow \sum_{t=1}^{T} \sum_{S \in \mathcal{S}_{i_t}} \mathbb{E}[P_j^t \mid Y^t(S) = 1] \cdot \Pr[Y^t(S) = 1] \le b_j$$

$$\Rightarrow \sum_{t=1}^{T} \sum_{S \in \mathcal{S}_{i_t}} p_{i_t j}(S) y^t(S) \le b_j,$$

which shows that the first constraint in LP (3) is satisfied.

Finally, using the same arguments, the revenue on a run of the algorithm is

$$\sum_{t=1}^{T} \sum_{S \in \mathcal{S}_{i_t}} \left(r_{i_t} P_{i_t}^t + \sum_{j \ne i_t} r_j^{\text{disc}} P_j^t \right) Y^t(S),$$

whose expectation is

$$\sum_{t=1}^{T} \sum_{S \in \mathcal{S}_{i_t}} \left(r_{i_t} + \sum_{j \ne i_t} r_j^{\text{disc}} p_{i_t j}(S) \right) y^t(S),$$

because $p_{i_t i_t}(S) = 1$ for all $S \in \mathcal{S}_{i_t}$.

We have shown that the expected revenue for every algorithm is equal to the objective value of the LP on a feasible solution of the LP. Therefore, the expected revenue of the algorithm is no greater than $\mathsf{OPT}(i_1, \dots, i_T)$, completing the proof of Lemma 1. □

Proof of Theorem 2. First we show that the expected value of $\mathsf{OPT}(i_1, \dots, i_T)$ is at least $C + 1$. In the case where customers $N+1, \dots, 2N$ do not arrive (which occurs with probability $1 - \frac{C}{M}$), a revenue of $N\left(\frac{1}{N}\right) = 1$ can be obtained by offering assortment {1, 2} during each $t = 1, \dots, N$. In the case where customers $N+1, \dots, 2N$ arrive (which occurs with probability $\frac{C}{M}$), it can be checked that $y^1(\{1\}) = \cdots = y^N(\{1\}) = 1$, $y^{N+1}(\{2,3\}) = \cdots = y^{2N}(\{2,3\}) = 1$ is a feasible solution to the LP. Therefore, in this case, $\mathsf{OPT}(i_1, \dots, i_T) \ge N\left(\frac{1}{N} + \frac{M}{N}\right) = 1 + M$. The expected value of $\mathsf{OPT}(i_1, \dots, i_T)$ is at least

$$\left(1 - \frac{C}{M}\right)(1) + \frac{C}{M}(1 + M) = 1 + C,$$

as desired.

Now consider any online algorithm. It must determine what fraction of the first N customers to offer item 2 to (as an add-on) without knowing whether the subsequent N customers requesting item 2 will arrive. Let $\alpha \in [0, 1]$ denote the fraction of the first N customers to whom the online algorithm offers item 2. After deciding α, should customers $N+1, \dots, 2N$ arrive, the

best the algorithm can do is use every remaining unit of item 2 to try to sell item 3. That is, the algorithm should serve $N - \alpha N$ of the customers of type 2 and offer item 3 as an add-on while it is available.

The expected revenue of such an algorithm is

$$\alpha N\left(\frac{1}{N}\right) + \frac{C}{M}\left((N-\alpha N)\left(\frac{1}{N}\right) + \left(1 - \left(1-\frac{1}{N}\right)^{N-\alpha N}\right)M\right), \tag{A.1}$$

which can be explained as follows:

- αN of the first N customers are offered item 2 as an add-on, guaranteeing a revenue of $\frac{1}{N}$ per customer, and
- only if the next N customers arrive (which occurs w.p. $\frac{C}{M}$) is there additional revenue:
  - — the remaining $N - \alpha N$ units of item 2 will be sold at the price of $\frac{1}{N}$ each, and
  - — the probability that item 3 is *not* sold is equal to the probability that all $N - \alpha N$ customers are offered {2, 3} and reject item 3, which occurs with probability $\left(1-\frac{1}{N}\right)^{N-\alpha N}$.

Note that (A.1) can be rearranged as $\alpha + C\left(1 - \left(1-\frac{1}{N}\right)^{N-\alpha N}\right) + \frac{C(1-\alpha)}{M}$. Taking the limit as $M, N \to \infty$, the expression is equal to

$$\alpha + C(1 - e^{\alpha - 1}).$$

We would like to argue that this expression, which is a continuous and differentiable function of α over the domain $[0,1]$, is maximized at $\alpha = 1 - \ln C$. Indeed, its derivative is $1 - Ce^{\alpha-1}$, which is positive for $\alpha \in [0, 1 - \ln C)$, 0 when $\alpha = 1 - \ln C$, and negative for $\alpha \in (1 - \ln C, 1]$.

Therefore, the expected revenue of any online algorithm is no greater than $1 - \ln C + C\left(1 - \frac{1}{C}\right) = C - \ln C$. Thus the competitive ratio of any online algorithm cannot exceed

$$\frac{C - \ln C}{C + 1}.$$

Substituting in the optimized value of $C = \frac{1}{w}$ (where $we^w = 1$) completes the proof of Theorem 2. □

Proof of Proposition 1. If step 6 of Procedure 2 is executed, then by the linearity of expectation,

$$d_j^{t+1}(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}, ()) = d_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}) + h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}),$$

and the desired result follows from the triangle inequality. Otherwise, if step 11 is executed, then again, by the linearity of expectation,

$$d_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}) = d_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L} - (\ell, \rho)) + \rho \cdot h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L} - (\ell, \rho)),$$

where $\mathcal{L} - (\ell, \rho)$ refers to the list before (ℓ, ρ) was appended. The desired result follows from the triangle inequality and the fact that $\rho \le 1$. □

Proof of Proposition 2. If the d_j^{curr}'s are updated in step 6, then $d_j^{\text{curr}} \le \frac{b_j}{2} - \varepsilon$ by the guarantee in step 5, whereas if they are updated in step 11, then $\rho \cdot \hat{h}_j \le \left(\frac{b_j}{2} - \varepsilon\right) - d_j^{\text{curr}}$ (because $\hat{h}_j \ge 0$) by the choice of ρ. □

Proof of Proposition 3. The first statement follows from the definition of $\mathcal{F}_K^t$, which is the set of items that appear in $\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}$. The second statement follows from the fact that after iteration K with customer t, item j will always appear in F in Procedure 1 and never appear in G in Procedure 2. □

Proof of Theorem 3. Consider any $t \in [T]$. In step 5 of Procedure 1, an extra iteration is possible only for each item that was added to $\mathcal{L}^t$ by step 9 of Procedure 2. However, items can be added to protection lists at most once, by Proposition 3. Therefore, the total number of iterations over $t = 1, \ldots, T$ (i.e., the value of $K_1 + \cdots + K_T$) is at most $T + n$.

As a result, Procedure 3 is called in at most $T + n$ times, and the total number of probabilities $h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L})$ estimated in Procedure 2 is at most $n(T + n)$. Applying Lemma 2 with $\varepsilon_2 = \frac{\varepsilon}{T+n}$, each of the estimates $\hat{h}_j$ is outside of $[h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}) - \varepsilon_2, h_j^t(\mathcal{L}^1, \ldots, \mathcal{L}^{t-1}, \mathcal{L}) + \varepsilon_2]$ with probability at most $2e^{-2\varepsilon_2^2 M} = \frac{\varepsilon}{n(T+n)}$. By the union bound, the probability that *any* estimate is not within ε_2 of its true value is at most ε.

Therefore, with probability at least $1 - \varepsilon$, there are no failures in the sampling, and the requirements on $\hat{h}_j$ in Proposition 1 are always met. For all $j \in [n]$, there is no error in the algorithm's value for d_j^{curr} at the start (i.e., the initial value of ε_1 is 0). Inductively applying Proposition 1, the total error accumulated over at most $T + n$ subiterations is at most $(T + n)\varepsilon_2 = \varepsilon$. That is, $|\hat{d}_j - \overline{d}_j| \le \varepsilon$. We can also inductively apply Proposition 2 to see that $\hat{d}_j \le \frac{b_j}{2} - \varepsilon$.

Combining the two preceding inequalities establishes that with probability at least $1 - \varepsilon$, we have $\hat{d}_j - \varepsilon \le \overline{d}_j \le \frac{b_j}{2}$ for all $j \in [n]$, as desired.

Now we analyze the runtime of our combined algorithm. The time taken by Procedure 1 to offer assortments until the end of time t is $O((t+n)n)$ (in step 9, $\mathcal{F}_k^s$ can easily be computed in $O(n)$ time by having an index from each item to its position in the protection lists). Therefore, the total runtime of Procedure 3 over the up to $T+n$ times it is called is

$$O((T+n)M(T+n)n)) = O\left(\frac{(T+n)^4 n}{\varepsilon^2} \cdot \ln\frac{(T+n)n}{\varepsilon}\right), \tag{A.2}$$

which is polynomial in T, n, and $\frac{1}{\varepsilon}$. It is easy to see that this is the bottleneck operation in the combined algorithm, completing the proof of Theorem 3. □

Appendix B. Running Example for Our Algorithm

In this section we provide a full running example of our algorithm (from Section 3), which demonstrates its full generality as well as the necessity of its complexity.

Our algorithm and analysis both extend to the case where multiple add-ons can be chosen, in which case the choice probabilities $p_{ij}(S)$, indicating the marginal probability of j being purchased when S is offered to type i, need to be generalized to expressions of the form $p_i(S', S)$, indicating the probability of S' being the exact subset chosen when S is offered to type i. Then, $p_{ij}(S)$ is derived through

$$p_{ij}(S) = \sum_{S' \subseteq S : j \in S'} \phi_i(S', S),$$

and the substitutability assumption translates to $p_{ij}(S_1) \geq p_{ij}(S_2)$ if $\{i,j\} \subseteq S_1 \subseteq S_2$. To simplify our notation in describing the choice models, we use two conventions, for brevity:

1. We only list the maximal subsets in $\mathcal{S}_i$ (this is enough because $\mathcal{S}_i$ is downward closed).
2. We do not list choice probabilities that can be inferred from other choice probabilities.

Example B.1. The setup is as follows:

- $b_1 = \infty$, $b_2 = \cdots = b_6 = 1$;
- $r_2 \gg r_3 \gg r_4 \gg r_5 \gg r_6 \gg r_1 = 0$, $r_j = r_j^{\text{disc}}$ for all j;
- $\mathcal{S}_1 = \{\{1,2\},\{1,3\},\{1,4,6\},\{1,5\}\}$:
  - $\phi_1(\{1,2\},\{1,2\}) = \frac{3}{4}$, $\phi_1(\{1,3\},\{1,3\}) = 1$;
  - $\phi_1(\{1,4,6\},\{1,4,6\}) = 1$, $\phi_1(\{1,5\},\{1,5\}) = 1$;
- $\mathcal{S}_3 = \{\{3,4\}\}$, $\phi_3(\{3,4\},\{3,4\}) = \frac{3}{5}$; and
- $(i_1, i_2, i_3) = (1,3,1)$.

To explain choice function ϕ_1 in words, customers of type 1 can be offered either add-on 2, which they purchase with probability 3/4; add-on 3, which they purchase with probability 1; add-ons 4 and 6, which they purchase with probability 1 (for *both*); or add-on 5, which they purchase with probability 1. It is also inferred from substitutability that customers of type 1 can be offered add-on 4 alone, which they purchase with probability 1; or add-on 6 alone, which they purchase with probability 1.

We now provide a full example of how our algorithm offers randomized assortments using Procedure 1, assuming that the proper protection lists have been given.

B.1. Description of Procedure 1.

Assume that the protection lists have been given as $\mathcal{L}^1 = \left(\left(2, \frac{2}{3}\right)\right)$, $\mathcal{L}^2 = ()$, $\mathcal{L}^3 = \left(\left(4, \frac{1}{6}\right), \left(5, \frac{13}{25}\right)\right)$.

Consider time $s = 1$. All inventories are full, so $I_{i_1} = I_1 > 0$ and $J = \emptyset$. With probability $\frac{2}{3}$, $k = 0$ and $F = \emptyset$, so $\tilde{A} = \{1,2\}$. With probability $\frac{1}{3}$, $k = 1$ and $F = \{2\}$, which prevents item 2 from being offered as an add-on, so $\tilde{A} = \{1,3\}$. After time period 1, there are three possibilities for the state of the inventory:

$$(I_1, \ldots, I_6) = \begin{cases} (\infty, 0, 1, 1, 1, 1) \text{ w.p. } \dfrac{2}{3} \cdot \dfrac{3}{4} = \dfrac{1}{2}; \\ (\infty, 1, 1, 1, 1, 1) \text{ w.p. } \dfrac{2}{3} \cdot \dfrac{1}{4} = \dfrac{1}{6}; \\ (\infty, 1, 0, 1, 1, 1) \text{ w.p. } \dfrac{1}{3} \cdot 1 = \dfrac{1}{3}. \end{cases}$$

Now consider $s = 2$. If $I_{i_2} = I_3 = 0$, then we have no choice but to offer $\tilde{A} = \emptyset$. Otherwise, $k = 0$, $F = \{2\}$, and J is either $\{2\}$ or $\emptyset$, so $\tilde{A} = \{3,4\}$. Although the customer buys item 4 as an add-on with probability $\frac{3}{5}$ (greater than $\frac{1}{2}$), we only get to this state

with probability $\frac{1}{2}+\frac{1}{6}=\frac{2}{3}$, so the add-on threshold for item 4 is not violated. After time 2, these are the possibilities for the state:

$$(I_1,\dots,I_6)=\begin{cases}(\infty,?,0,0,1,1) \text{ w.p. } \frac{2}{3}\cdot\frac{3}{5}=\frac{2}{5};\\ (\infty,?,0,1,1,1) \text{ w.p. } \frac{1}{3}+\frac{2}{3}\cdot\frac{2}{5}=\frac{3}{5}.\end{cases}$$

Note that we have merged some states because it will be irrelevant for the remainder of the execution whether $I_2=0$ or $I_2=1$ (item 2 will always be in F and forbidden from being sold as an add-on).

Now consider $s=3$. Note that J always contains 3 and contains 4 with probability $\frac{2}{5}$. Independent of this, $k=0$ w.p. $\frac{1}{6}$, $k=1$ w.p. $\left(1-\frac{1}{6}\right)\frac{13}{25}=\frac{13}{30}$, and $k=2$ w.p. $\left(1-\frac{1}{6}\right)\left(1-\frac{13}{25}\right)=\frac{2}{5}$. The respective forbidden sets are {2}, {2, 4}, and {2, 4, 5}. When $4\notin J$ and $F=\{2\}$, the revenue-maximizing assortment is {1, 4, 6} (guaranteeing the sales of items 4 and 6). Alternatively, if $\{4,5,6\}\cap(J\cup F)=\{4\}$, then $\tilde{A}=\{1,5\}$. Finally, if $\{4,5,6\}\cap(J\cup F)=\{4,5\}$, then $\tilde{A}=\{1,6\}$. Note that because $\tilde{A}$ only depends on $J\cup F$, if $4\in J$, then it does not matter whether $k=0$ ($F=\{2\}$) or $k=1$ ($F=\{2,4\}$). The final possibilities for the state at the end of time $t=3$ can be computed to be

$$(I_1,\dots,I_6)=\begin{cases}(\infty,?,0,0,0,1) \text{ w.p. } \frac{2}{5}\cdot\left(\frac{1}{6}+\frac{13}{30}\right)=\frac{6}{25};\\ (\infty,?,0,0,1,0) \text{ w.p. } \frac{2}{5}\cdot\frac{2}{5}+\frac{3}{5}\cdot\frac{1}{6}=\frac{13}{50};\\ (\infty,?,0,1,0,1) \text{ w.p. } \frac{3}{5}\cdot\frac{13}{30}=\frac{13}{50};\\ (\infty,?,0,1,1,0) \text{ w.p. } \frac{3}{5}\cdot\frac{2}{5}=\frac{6}{25}.\end{cases}$$

It can be seen that at the end of the above-mentioned execution that $\Pr[I_4=0]=\Pr[I_5=0]=\Pr[I_6=0]=\frac{13}{50}+\frac{6}{25}=\frac{1}{2}$; that is, the add-on thresholds for items 4, 5, and 6 are respected. Now we redescribe the execution while interleaving descriptions of how Procedure 2 constructs the protection lists. We assume for simplicity that the sampling error is $\varepsilon=0$.

B.2. Description of Procedure 2.

Consider time $t=1$. At the very start, d_j^{curr} has been initialized to $d_j^1(())=0$ for all $j\in[6]$. During the first run through the loop, $\mathcal{L}$ is (), and G, the set of items that have not reached their add-on thresholds, is $\{2,\dots,6\}$. With an empty protection list, Procedure 1 deterministically offers assortment {1, 2} during time 1, resulting in item 2 being sold as an add-on with probability $h_2^1(\mathcal{L})=\frac{3}{4}$. Therefore, $\hat{h}_2>\frac{1}{2}$, and the add-on threshold in step 5 of Procedure 2 is violated for $j=2$. In step 9, $\ell=2$ and $\rho=\frac{\frac{1}{2}-0}{\frac{3}{4}}=\frac{2}{3}$. We compute ρ so that had the algorithm offered {1, 2} with probability ρ (instead of probability 1), then the expected units of item 2 sold as an add-on would be exactly $\frac{b_2}{2}$, as indicated by the updates in step 11.

The algorithm tries again with $\mathcal{L}=\left(\left(2,\frac{2}{3}\right)\right)$. Now, $\hat{h}_3=\frac{1}{3}$, because the probability of getting to the final iteration with $k=K_t$ is $1-\frac{2}{3}=\frac{1}{3}$, after which item 3 is deterministically sold as an add-on. The add-on thresholds in step 5 are upheld, and the algorithm updates $d_3^{\text{curr}}\leftarrow\frac{1}{3}$, $\mathcal{L}^1\leftarrow\mathcal{L}$. Note that $2\notin G$, and d_2^{curr} will never again be updated, remaining at $\frac{1}{2}$ until the end of the selling season.

Now consider time $t=2$. It is straightforward to see that $\hat{h}_4$ is the only nonzero $\hat{h}_j$, with $\hat{h}_4=\frac{2}{3}\cdot\frac{3}{5}=\frac{2}{5}$. The algorithm updates $d_4^{\text{curr}}\leftarrow\frac{2}{5}$ and proceeds to time 3 with $\mathcal{L}^2=()$.

From the analysis in the previous subsection, the state of the inventory $(I_1,\dots,I_6)$ at the start of time 3 is $(\infty,?,0,0,1,1)$ with probability $\frac{2}{5}$ and $(\infty,?,0,1,1,1)$ with probability $\frac{3}{5}$. The expectations $(d_1^{\text{curr}},\dots,d_6^{\text{curr}})$ are at $\left(0,\frac{1}{2},\frac{1}{3},\frac{2}{5},0,0\right)$. Now consider the execution of Procedure 1 with lists $(\mathcal{L}^1,\mathcal{L}^2,\mathcal{L})$, where $\mathcal{L}=()$. With probability $\frac{2}{5}$ (when $I_4=0$), the subroutine will offer {1, 5} (because $4\in J$), resulting in the deterministic sale of item 5 as an add-on. With probability $\frac{3}{5}$, the subroutine will offer {1, 4, 6} and sell items 4 and 6 as add-ons. Therefore, $\hat{h}_4=\hat{h}_6=\frac{3}{5}$, whereas $\hat{h}_5=\frac{2}{5}$. The add-on threshold in step 5 is violated for both $j=4$ and $j=6$. In step 9, the expression $\frac{\frac{b_j}{2}-d_j^{\text{curr}}}{\hat{h}_j}$ is equal to $\frac{\frac{1}{2}-\frac{2}{5}}{\frac{3}{5}}=\frac{1}{6}$ when $j=4$, and $\frac{\frac{1}{2}-0}{\frac{3}{5}}=\frac{5}{6}$ when $j=6$. Therefore, we choose $\ell=4$ and $\rho=\frac{1}{6}$. In step 11, $d_4^{\text{curr}}\leftarrow\frac{2}{5}+\frac{1}{6}\cdot\frac{3}{5}=\frac{1}{2}$, $d_5^{\text{curr}}\leftarrow\frac{1}{6}\cdot\frac{2}{5}=\frac{1}{15}$, and $d_6^{\text{curr}}\leftarrow\frac{1}{6}\cdot\frac{3}{5}=\frac{1}{10}$. Note that choosing an ℓ that *minimizes* ρ is important for respecting the add-on thresholds; had we chosen $\ell=6$ and $\rho=\frac{5}{6}$, then d_4^{curr} would be updated to $\frac{2}{5}+\frac{5}{6}\cdot\frac{3}{5}>\frac{1}{2}$.

The algorithm tries again with updated list $\mathcal{L}=\left(\left(4,\frac{1}{6}\right)\right)$ and updated expectations $(d_1^{\text{curr}},\dots,d_6^{\text{curr}})=\left(0,\frac{1}{2},\frac{1}{3},\frac{1}{2},\frac{1}{15},\frac{1}{10}\right)$. The inventory possibilities are still the same, and the probability that Procedure 1 gets to the final iteration with $k=K_t$ is $1-\frac{1}{6}=\frac{5}{6}$. Regardless of whether $I_4=1$, it will offer (and successfully sell) item 5 as an add-on, so $\hat{h}_5=\frac{5}{6}$, which exceeds $\frac{1}{2}$. Therefore, $\ell=5$ and $\rho=\frac{\frac{1}{2}-\frac{1}{15}}{\frac{5}{6}}=\frac{13}{25}$. Note that item 6 violated its add-on threshold with an earlier list but no longer violates its add-on threshold

even though $d_6^{\texttt{curr}}$ increased with the earlier iteration. This is why adding items to the list one at a time, even when multiple items violate their add-on thresholds, is important.

The algorithm tries again, having appended $\left(5, \frac{13}{25}\right)$ to $\mathcal{L}$ and updated $d_5^{\texttt{curr}}$ to $\frac{1}{2}$. Now Procedure 1 gets to the final iteration with probability $\left(1-\frac{1}{6}\right)\left(1-\frac{13}{25}\right)=\frac{2}{5}$, in which case it will deterministically sell item 6 as an add-on. Therefore, $\hat{h}_6=\frac{2}{5}$, which when added to $d_6^{\texttt{curr}}=\frac{1}{10}$ is exactly $\frac{1}{2}$. The algorithm proceeds to the next time period with $\mathcal{L}^3 \leftarrow \mathcal{L}$ even though $d_6^{\texttt{curr}}$ will be $\frac{1}{2}$; item 6 will immediately get added to $\mathcal{L}$ if Procedure 1 attempts to offer it as an add-on again.

Appendix C. Exponential Inventory Balancing on Our Problem

First we describe an example with large starting inventories where the competitive ratio is still no better than 1/2. We also discuss why using an exponential penalty function, as in Golrezaei et al. [24], does not seem to improve the competitive ratio for our problem.

C.1. Example.

Consider a "scaled-up" version of Example 2 with initial inventories and a selling season multiplied by k, a large integer. Now the randomized instance is defined as follows:

- $n=3$,
- $r_1=r_1^{\text{disc}}=0$, $b_1=kN$,
- $r_2=r_2^{\text{disc}}=\frac{1}{N}$, $b_2=kN$,
- $r_3=r_3^{\text{disc}}=M$, $b_3=k$,
- $\mathcal{S}_1=\{\{1,2\}\}$, $p_{12}(\{1,2\})=1$,
- $\mathcal{S}_2=\{\{2,3\}\}$, and $p_{23}(\{2,3\})=\frac{1}{N}$.
- The arrivals are as follows:
  - $T=kN, i_1=\cdots=i_{kN}=1$, w.p. $1-\frac{C}{M}$, and
  - $T=2kN, i_1=\cdots=i_{kN}=1, i_{kN+1}=\cdots=i_{2kN}=2$, w.p. $\frac{C}{M}$.

The expected value of $\mathsf{OPT}(i_1,\ldots,i_T)$ has been multiplied by k, so it is now $k(1+C)$. Intuitively, the offline fractional solution will always extract the full revenue of $\frac{1}{N}\cdot kN=k$ from item 2 and extract the revenue of $M\cdot k$ from item 3 w.p. $\frac{C}{M}$. Details on the feasible LP solutions can be found in the proof of Theorem 2.

Now, consider any online algorithm. It faces the same conundrum as before: it must determine what fraction of the first kN customers to offer item 2 without knowing whether the subsequent kN customers will come. Let $\alpha\in[0,1]$ be the fraction of the first kN customers served. If the subsequent kN customers come, $(1-\alpha)kN$ units of item 2 would be remaining for the purpose of selling item 3. The number of units of item 3 sold is then $\min\{X,k\}$, where X is a Binomial random variable with parameters $\left((1-\alpha)kN, \frac{1}{N}\right)$. Clearly,

$$\mathbb{E}[\min\{X,k\}]<\mathbb{E}[X]=(1-\alpha)k. \tag{C.1}$$

In the proof of Theorem 2 we explicitly compute the difference between $\mathbb{E}[\min\{X,k\}]$ and $\mathbb{E}[X]$, although for any fixed $\alpha<1$, the law of large numbers suggests that $\mathbb{E}[\min\{X,k\}]\to\mathbb{E}[X]$ as $k\to\infty$.

Nonetheless, for any k, we can conclude from (C.1) that the expected revenue of an online algorithm serving α of the first kN customers is less than $\alpha k+C(1-\alpha)k$, as $M,N\to\infty$ (for details, see the proof of Theorem 2). Choosing $C=1$, $\alpha k+C(1-\alpha)k=k$ regardless of what α the algorithm chooses, and $\mathbb{E}[\mathsf{OPT}(i_1,\ldots,i_T)]=2k$. Therefore, the competitive ratio must be less than $\frac{1}{2}$ even as $\min\{b_1,\ldots,b_n\}\to\infty$.

C.2. Comparison of Exponential Inventory Balancing and Protection Level in Expectation

For their problem, Golrezaei et al. [24] recommend offering a revenue-maximizing assortment during every time period, where the revenue of each item is scaled by a function of its fraction of starting inventory remaining. This function is called a *penalty function*. The intuition is that if a large fraction of an item has already been sold, then lower priority should be given to selling the remaining units. As the starting inventories become large, their *exponential* penalty function accomplishes the optimal trade-off between selling a high-revenue item and selling a low-revenue item that may not be wanted later.

In our problem, there is a further trade-off between selling a high-revenue item and saving it for *even higher* revenue later. As a result, selling too much of an item as an add-on has the added downside that the inventory could be being disposed for an arbitrarily small fraction of its potential revenue, and every unit depleted as an add-on incurs the *same* risk in revenue loss. The example in the previous subsection illustrates this—every unit of item 2 sold to the first kN customers loses the same fraction ($\frac{1}{N}$) of item 3's large revenue (M), should the final kN customers arrive.

Therefore, placing higher value on the final remaining units of each item in the form of an exponential penalty function appears to yield no benefit. In fact, if we directly use the penalty functions of Golrezaei et al. [24] on our problem, then the competitive ratio guarantee is 0. To see this, note that the penalty function–based algorithm would sell all k copies of item 2 as an add-on, because its "penalized" revenue would still be positive as long as there is positive remaining inventory. Thus in the scenario where the final kN customers arrive, the algorithm would stock out of item 2 and sell no copies of item 3, whereas the optimum would have protected all of its copies of item 2 to earn the large revenue of kM from item 3.

Our algorithm remedies this issue by withholding half of the inventory of each item (in expectation) from being sold as an add-on. In that sense, our algorithm also scales the revenue of each item by a penalty factor, but our penalty factor is a function of the item's *expected* fraction of starting inventory remaining, and that function over $[0,1]$ is the function that is 0 on $\left[0,\frac{1}{2}\right]$ and 1 on $\left(\frac{1}{2},1\right]$.

Endnotes

[1] In general, the firm would need to flip another coin to decide whether to sell item 3 as an add-on, because the full price of item 3 could be M^2 (which is $\gg M$), or item 3 could be a primary item that enables the sales of another add-on with price M^2.

[2] This example shows why the positive correlation between protecting items does not cure the problem. Although negative correlation does cure the problem on this example, it is unclear how to define negative correlation in general when there are a large number of items, all of which could be sold as add-ons to one another.

References

[1] Adamczyk M, Grandoni F, Mukherjee J (2015) Improved approximation algorithms for stochastic matching. Bansal N, Finocchi I, eds. *Algorithms—ESA 2015* (Springer, Berlin, Heidelberg), 1–12.
[2] Aggarwal G, Goel G, Karande C, Mehta A (2011) Online vertex-weighted bipartite matching and single-bid budgeted allocations. *Proc. 22nd Annual ACM-SIAM Sympos. Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia), 1253–1264.
[3] Ball MO, Queyranne M (2009) Toward robust revenue management: Competitive analysis of online booking. *Oper. Res.* 57(4):950–963.
[4] Bernstein F, Kök AG, Xie L (2015) Dynamic assortment customization with limited inventories. *Manufacturing Service Oper. Management* 17(4):538–553.
[5] Bitran G, Caldentey R (2003) An overview of pricing models for revenue management. *Manufacturing Service Oper. Management* 5(3):203–229.
[6] Blanchet J, Gallego G, Goyal V (2016) A Markov chain approximation to choice modeling. *Oper. Res.* 64(4):886–905.
[7] Borodin A, El-Yaniv R (2005) *Online Computation and Competitive Analysis* (Cambridge University Press, Cambridge, UK).
[8] Brubach B, Sankararaman KA, Srinivasan A, Xu P (2017) Attenuate locally, win globally: An attenuation-based framework for online stochastic matching with timeouts. Das S, Durfee E, Larson K, Winikoff M, eds. *Proc. 16th Conf. Autonomous Agents Multiagent Systems* (International Foundation for Autonomous Agents and Multiagent Systems), 1223–1231.
[9] Bubeck S, Devanur NR, Huang Z, Niazadeh R (2019) Multi-scale online learning and its applications to online auctions. *J. Machine Learn. Res.* 20(62):1–37.
[10] Buchbinder N, Jain K, Naor JS (2007) Online primal-dual algorithms for maximizing ad-auctions revenue. Arge L, Hoffmann M, Welzl E, eds. *Algorithms—ESA 2007* (Springer, Berlin), 253–264.
[11] Buchbinder N, Chen S, Naor J, Shamir O (2016) Unified algorithms for online learning and competitive analysis. *Math. Oper. Res.* 41(2):612–625.
[12] Chan CW, Farias VF (2009) Stochastic depletion problems: Effective myopic policies for a class of dynamic optimization problems. *Math. Oper. Res.* 34(2):333–350.
[13] Cheung WC, Simchi-Levi D (2016) Efficiency and performance guarantees for choice-based network revenue management problems with flexible products. Preprint, submitted August 15, http://dx.doi.org/10.2139/ssrn.2823339.
[14] Cohen MC, Lobel I, Paes Leme R (2020) Feature-based dynamic pricing. *Management Sci.* 66(11):4921–4943.
[15] Dean BC, Goemans MX, Vondrák J (2008) Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.* 33(4):945–964.
[16] Désir A, Goyal V, Zhang J (2022) Capacitated assortment optimization: Hardness and approximation. *Oper Res.* 70(2):893–904.
[17] Désir A, Goyal V, Segev D, Ye C (2020) Constrained assortment optimization under the Markov chain–based choice model. *Management Sci.* 66(2):698–721.
[18] Devanur NR, Jain K, Kleinberg RD (2013) Randomized primal-dual analysis of RANKING for online bipartite matching. Khanna S, ed. *Proc. 24th Annual ACM-SIAM Sympos. Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia), 101–107.
[19] Elmaghraby W, Keskinocak P (2003) Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions. *Management Sci.* 49(10):1287–1309.
[20] Gallego G, Topaloglu H (2014) Constrained assortment optimization for the nested logit model. *Management Sci.* 60(10):2583–2601.
[21] Gallego G, van Ryzin G (1994) Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Sci.* 40(8):999–1020.
[22] Gallego G, Iyengar G, Phillips R, Dubey A (2004) Managing flexible products on a network. Preprint, submitted July 1, http://dx.doi.org/10.2139/ssrn.3567371.
[23] Gallego G, Li A, Truong VA, Wang X (2016) Online personalized resource allocation with customer choice. Technical report, Columbia University, New York.
[24] Golrezaei N, Nazerzadeh H, Rusmevichientong P (2014) Real-time optimization of personalized assortments. *Management Sci.* 60(6):1532–1551.
[25] Kalyanasundaram B, Pruhs KR (2000) An optimal deterministic algorithm for online b-matching. *Theoret. Comput. Sci.* 233(1):319–325.
[26] Karp RM, Vazirani UV, Vazirani VV (1990) An optimal algorithm for on-line bipartite matching. *Proc. 22nd Annual ACM Sympos. Theory Comput.* (ACM, New York), 352–358.
[27] Krumke SO (2002) Online optimization: Competitive analysis and beyond. Postdoctoral thesis, Technische Universität Berlin, Berlin.
[28] Lan Y, Gao H, Ball MO, Karaesmen I (2008) Revenue management with limited demand information. *Management Sci.* 54(9):1594–1609.
[29] Li G, Rusmevichientong P, Topaloglu H (2015) The *d*-level nested logit model: Assortment and price optimization problems. *Oper. Res.* 63(2):325–342.
[30] Linden G, Smith B, York J (2003) Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Comput.* 7(1):76–80.
[31] Lugosi G (2009) Concentration-of-measure inequalities. Working paper, Pompeu Fabra University, Barcelona, Spain. http://www.econ.upf.edu/lugosi/anu.pdf.

[32] Ma W (2018) Improvements and generalizations of stochastic knapsack and Markovian bandits approximation algorithms. *Math. Oper. Res.* 43(3):789–812.
[33] Maglaras C, Meissner J (2006) Dynamic pricing strategies for multiproduct revenue management problems. *Manufacturing Service Oper. Management* 8(2):136–148.
[34] Mahajan S, van Ryzin G (2001) Stocking retail assortments under dynamic consumer substitution. *Oper. Res.* 49(3):334–351.
[35] Mehta A, Panigrahi D (2012) Online matching with stochastic rewards. IEEE 53rd Annual Sympos. Foundations Comput. Sci. (IEEE, Piscataway, NJ), 728–737.
[36] Mehta A, Waggoner B, Zadimoghaddam M (2015) Online stochastic matching with unequal probabilities. *Proc. 26th Annual ACM-SIAM Sympos. Discrete Algorithms* (SIAM, Philadelphia), 1388–1404.
[37] Mehta A, Saberi A, Vazirani U, Vazirani V (2007) AdWords and generalized online matching. *J. ACM* 54(5):Article 22.
[38] Rusmevichientong P, Shen ZJ, Shmoys D (2010) Dynamic assortment optimization with a multinomial logic choice model and capacity constraint. *Oper. Res.* 58(6):1666–1680.
[39] Talluri K, van Ryzin G (2004) Revenue management under a general discrete choice model of consumer behavior. *Management Sci.* 50(1):15–33.
[40] van Ryzin G, Mahajan S (1999) On the relationship between inventory costs and variety benefits in retail assortments. *Management Sci.* 45(11):1496–1509.
[41] Wood M (2014) A new kind of e-commerce adds a personal touch. *New York Times* (August 13), https://www.nytimes.com/2014/08/14/technology/personaltech/data-driven-shopping-with-the-personal-touch.html.
[42] Xu Y, Wang Z (2018) Assortment optimization for a multi-stage choice model. Preprint, submitted September 4, http://dx.doi.org/10.2139/ssrn.3243742.
[43] Yao ACC (1977) Probabilistic computations: Toward a unified measure of complexity. 18th Annual Sympos. Foundations Comput. Sci. (IEEE, Piscataway, NJ), 222–227.
[44] Yuan M, Pavlidis Y, Jain M, Caster K (2016) Walmart online grocery personalization: Behavioral insights and basket recommendations. Link S, Trujillo JC, eds. Adv. Conceptual Modeling: ER 2016 Workshops (Springer, Cham, Switzerland), 49–64.