PRODUCTION AND OPERATIONS MANAGEMENT
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 1059-1478 | EISSN 1937-5956 | 00 | 0000 | 0001

POMS
DOI 10.1287/xxxx.0000.0000

Dynamic Learning and Assortment Decisions under a Non-parametric Choice Model

Yifan Li[a], Hang Wei[a], Lei Xie[a], Chaonan Zheng[a], Houmin Yan[b]
[a]College of Business, Shanghai University of Finance and Economics, 200433, China
[b]Department of Management Sciences, College of Business, City University of Hong Kong, Hong Kong, Kowloon, China

Abstract: We consider a dynamic leaning and assortment selection problem. Every period features the arrival of one customer, whose purchase pattern follows a non-parametric choice model. The firm offers a subset (assortment) of n substitutable products, from which the customer decides his/her purchase according to the specific purchase pattern. All potential purchase patterns are assumed to be known to the firm; however, the arrival probabilities of each arriving customer's type are not. The firm can only observe transaction data, without insights into customer types. The firm's objective is to optimize the total expected revenue over T periods while learning the arrival probabilities by incomplete information. We develop a upper confidence bound(UCB)-based policy that creates a linear confidence space for the purchase probability vector and selects the assortment that maximizes potential revenue within this space. This approach achieves a sublinear regret which depends on the total number of index sets, effectively capturing the complexity of the learning problem within our non-parametric choice model. To improve the computational tractability of assortment decisions, we use a subgradient-based method by implementing Linear Programming (LP) relaxation and assortment randomization in the assortment selection process. We reveal that this algorithm reduces the regret performance to be linear under a general instance as the penalty of relaxation and assortment randomization, but there exists a boundary condition that can lead to a sublinear regret.

Key words: Non-parametric choice model; Assortment selection; Dynamic learning; UCB; Binary online gradient descent.

1 Introduction

1.1 Background and Motivation

The dynamic assortment and learning problem has captured growing attention in the field of online learning (Rusmevichientong et al. 2010, Sauré and Zeevi 2013, Agrawal et al. 2017, Agrawal et al. 2019, Chen et al. 2021b, Li et al. 2024), where companies simultaneously learn the patterns of customer choice and maximize expected revenue by offering assortments. A customer's choice pattern typically adheres to a choice model that defines the purchasing behavior or process within a set of substitutable products (Talluri and Van Ryzin 2004). However, most studies pay attention to parametric choice models, such as the Multinomial Logit (MNL) model and Markov Chain choice models. These approaches define the probability of customer

choices as a function of utility, including product attributes and prices. To the best of our knowledge, limited research has been focused on dynamic learning and assortment problems under non-parametric choice models. In a non-parametric choice model, each customer class or type is characterized by an arrival probability and a distinct ranking of a subset of products, known as a preference list. There are no restrictions on potential preference lists. When presented with an assortment of products, a customer will select the highest-ranking product available in his/her preference list. If none of the offered products appear in the customer's preference list, he/she will choose not to make a purchase and leave without buying anything. In the literature, two prominent non-parametric choice models are widely recognized: rank-based choice models and tree-based choice models. In the rank-based choice model, the preference list consists solely of a product list arranged in preference order. A customer will select the highest-ranking product available from this list. In the tree-based choice model, the preference list is represented by the decision path traversed through a decision tree, which is shaped by the products available in the assortment. A customer's purchase decision is guided by the specific decision path that corresponds to his or her individual decision tree. Both rank-based and tree-based choice models allow for a diverse representation of customer preferences, enabling more accurate modeling of consumer behavior in various purchasing contexts. These models are particularly effective in capturing irrational purchase behaviors and addressing the non-independence of irrelevant alternatives (non-IIA) property that traditional parametric choice models possess.

We are motivated to investigate a dynamic learning and assortment problem using a generic non-parametric choice model. In each period, there is one arriving customer whose purchase pattern follows a non-parametric choice model. All potential purchase patterns are assumed to be known to the firm; however, the probabilities of each arriving customer's pattern are not. The firm can only observe transaction data, without insights into customer types. More specifically, we explore how to learn the unknown arrival probability of each customer type within a known collection of customer types. The firm decides to offer a subset (assortment) of n substitutable products, from which the customer decides his/her purchase according to the specific purchase pattern. The firm's objective is to optimize the total expected revenue over the T periods while learning the arrival probabilities by incomplete information.

1.2 Contributions

Our study contributes to the existing literature on dynamic learning and assortment in the following two aspects.

1. Learning purchase probabilities using a UCB-based method. A key objective of this learning problem is to accurately estimate arrival type probabilities. However, within the framework of a non-parametric choice model, a significant challenge arises: the observation of customer purchases does not allow firms to effectively tailor their offerings to specific customer types. This is because customers from different types may respond similarly to a given assortment, leading to identical purchase decisions. As a result, purchase

observations provide incomplete information on customer types. To estimate arrival type probabilities, we consider the aggregation of arrival probabilities for customers who exhibit the same purchase to a given assortment, which we refer to as the customer index set in our analysis. For each customer index set, we can formulate a concentration inequality around the sum of arrival probabilities for all customer types within that set. By considering all customer index sets, we can establish a linear confidence space for the purchase probability vector, which is a crucial component of our subsequent development of learning algorithms. We develop a bandit-like algorithm that achieves an expected regret of order at $O(\sqrt{\kappa KT \log \kappa T})$, where κ denotes the total number of the index set.

To some extent, κ serves as an important indicator of problem complexity and has a significant impact on the optimal performance that can be achieved. An arbitrary collection of customer types could lead to an overwhelming number of customer index sets. It is true that in cases of extreme irrationality among customers, we may encounter situations where no discernible patterns emerge between the customer index sets. As demonstrated in relatively rational scenarios discussed by Wang (2022) and Chen and Mišić (2022), we respectively present illustrative examples in the rank-based choice model and the tree-based choice model and show that the number of potential index sets may not be excessively large.

2. Improving computational tractability using a subgradient-based method. It is well established in the literature that assortment optimization problems under non-parametric choice models are *NP-hard*. This holds for both rank-based choice models (Feldman et al. 2019) and tree-based choice models (Chen and Mišić 2021, Aouad et al. 2018). This requires us to consider not only regret performance and the associated exploration-exploitation trade-off, but also incorporate computational tractability into this balance.

We leverage LP-relaxation along with a subgradient-based method to decrease the computational complexity of the assortment optimization problem, effectively transforming it from an integer programming (IP) problem into an LP problem. A randomization process is then used to generate the assortment. Although this algorithm significantly improves computational efficiency, it does come at the expense of regret performance. Due to loss in the process of randomizing an assortment, this leads to a linear regret $O(n\sqrt{T} + \sqrt{\kappa nT \log \kappa T} + nT)$ for any general instance of the true non-parametric choice model, where the term $O(nT)$ represents the penalty of relaxation and randomization. Despite the linear regret in general cases, we reveal that there does exist some special case, referring to the boundary condition discussed in Section 5.3, that can lead to sublinear regret by eliminating the term $O(nT)$.

1.3 Organization fo the Paper

The rest of this paper is organized as follows. Section 2 reviews the relevant literature. In Section 3, we formulate the problem. Section 4 analyzes the UCB-based algorithm and its theoretical performance. We propose a subgradient-based algorithm and examine its theoretical performance in Section 5. Then, we conduct a numerical analysis in Section 6 to assess the performance and computational efficiency of both the UCB-based and subgradient-based algorithms. Finally, we conclude the paper in Section 7.

2 Literature Review

2.1 Non-parametric Choice Models and Assortment Optimization

Our research is related to the literature on non-parametric choice models. Unlike parametric choice models depending on predefined functional forms, non-parametric choice models can capture complex patterns and relationships in consumer behavior without such constraints. This flexibility makes non-parametric models particularly valuable in understanding dynamic learning and assortment optimization. The rank-based choice model and the tree-based choice model are two prominent non-parametric choice models that are frequently utilized to describe customers' purchasing patterns. Farias et al. (2013) propose a non-parametric data-driven approach viewing choice models generically as distributions over rankings (or preference lists) of products. Specifically, the purchase pattern in the rank-based choice model is defined as the preference list of customer type g meaning that σ_g specifies a rank order over all potential products with $\sigma_g(a)$ denoting the preference rank of product a (Jagabathula and Vulcano 2018). Usually, lower ranks indicate higher preference, so that a customer described by σ_g prefers product a to product b if and only if $\sigma_g(a) < \sigma_g(b)$. Chen and Mišić (2022) propose a tree-based choice model in which each type of customer is associated with a purchase decision tree. Furthermore, Chen and Mišić (2022) demonstrate that any choice function, irrespective of its rationality, can be represented as a decision forest. Our paper studies a dynamic learning and assortment problem under a generic non-parametric choice model that includes tree-based and rank-based selection patterns.

Another stream of literature is online learning for assortment optimization. Learning algorithms for assortment optimization are increasingly becoming a significant area of research. Sauré and Zeevi (2013) utilize the explore-then-commit approach to investigate a range of stylized assortment planning scenarios where a predetermined transition threshold is applied throughout the selling period. Given some additional parameter identification assumptions, Sauré and Zeevi (2013) demonstrate an asymptotic $O(n \log T)$ regret bound. Agrawal et al. (2017) consider a Thompson sampling algorithm for selecting dynamic assortments with a fixed cardinality constraint K that achieves a regret bound of $O(\sqrt{nT} \log TK)$. Agrawal et al. (2019) customize the UCB approach to solve the dynamic assortment optimization problem and achieve a regret bound of $\tilde{O}(\sqrt{nT})$. Chen et al. (2021b) develop a trisection-based policy combined with adaptive confidence bound construction and relax the regret bound on the dependence of n. Chen et al. (2021a) study the problem of dynamic assortment planning under the nested logit model through a novel UCB policy with an aggregated estimation scheme. The aforementioned algorithms are efficient online learning methods tailored for more tractable choice models, such as MNL models, which have been extensively studied in recent literature. Recently, Li et al. (2024) study a dynamic assortment selection problem under a Markov chain choice (MCC) model through a fast linear system-based explore-then-commit learning algorithm that balances the trade-off between exploration and exploitation. In addition, contextual bandit learning algorithms

for assortment optimization have also been examined in several studies (Bernstein et al. 2019, Kallus and Udell 2020, Simchi-Levi and Xu 2022, Chen et al. 2021b).

Our work also falls into the literature of assortment problems under non-parametric choice models. Honhon et al. (2012) study the retailer's product selection problem when products have different price and cost parameters based on a ranking-based consumer choice model. Jagabathula and Rusmevichientong (2017) capture the joint effect of assortment and price based on a non-parametric framework. Paul et al. (2018) consider assortment and pricing problems based on a non-parametric tree customer choice model. Feldman et al. (2019) focus on the assortment optimization problem under a k-product non-parametric choice model. Jagabathula and Vulcano (2018) develop a tractable non-parametric expectation maximization (EM) algorithm to fit choice models by aggregating transaction data. Most of the estimation methods proposed in the literature mainly use offline transaction data to estimate the most fitting choice model (Farias et al. 2013, van Ryzin and Vulcano 2015, 2017). Recently, Ho-Nguyen and Kılınç-Karzan (2021) study estimating a non-parametric choice model from observational data in a dynamic setting based on online convex optimization. Susan et al. (2024) study the problem of actively learning a non-parametric choice model based on consumers' decisions and introduce a directed acyclic graph (DAG) representation encoding all the information about the choice model which can be inferred from the available data. However, to the best of our knowledge, there remains a gap in the literature regarding dynamic learning and assortment problems based on non-parametric choice models.

2.2 Methodology

The algorithms proposed in this paper are relevant to the combinatorial bandit literature. A combinatorial bandit can be understood as a linear bandit with a unique combinatorial action set (Lattimore and Szepesvári 2020). Cesa-Bianchi and Lugosi (2012) provide a comprehensive overview of known applications in the online combinatorial bandit. Bubeck and Cesa-Bianchi (2012) focus on two extreme cases in which the analysis of regret is particularly simple and elegant: i.i.d. payoffs and adversarial payoffs. Cohen and Hazan (2015) investigate a new Follow the Perturbed Leader (FTPL) algorithm for online structured prediction problems. In certain situations, the dynamic assortment problem can be considered as a straightforward combinatorial bandit problem with semi-bandit feedback, where the revenue generated from each decision is solely determined by that particular choice. Semi-bandits have been introduced in the context of shortest-path problems by Gergely (2015). Abernethy et al. (2014) present the core ideas in the prediction with expert advice setting. Combes et al. (2015) investigate stochastic and adversarial combinatorial multi-armed bandit problems. As for the stochastic setting with semi-bandit feedback, they derive a problem-specific regret lower bound, and analyze how it scales with the dimension of the decision space.

Online convex optimization (OCO) is also a relevant research area for the methodology presented in this paper. The online convex optimization model, initially defined by Martin (2003), has subsequently garnered

significant attention within the learning community. Yu and Neely (2020) consider the problems with time-varying objective functions as in the classical online convex optimization. Ding et al. (2021) investigate the online convex optimization with long-term constraints which is widely used in various resource allocations and recommendation systems. Lesage-Landry et al. (2021) explore online optimization with binary decision variables and convex loss functions by designing a novel algorithm called binary online gradient descent (bOGD), which bounds its expected dynamic regret. Shen et al. (2023) form an OCO problem in which the objective function remains fixed while the domain changes over time. Recently, Meng and Liu (2024) focus on dynamic regret for non-stationary online convex optimization with full information. Cardoso et al. (2024) generalizes the online convex optimization framework to study the online saddle point problem. Different from the above bandit learning and online convex optimization literature, we develop two algorithms to solve dynamic assortment optimization problems considering not only the regret performance, but also the computational tractability.

3 Problem Formulation

3.1 Basic Model

We consider learning and assortment optimization with a non-parametric choice model over a finite horizon. The potential product set is denoted as $[n] = \{1, 2, ..., n\}$. All products are substitutes. The selling price of product $i \in [n]$ is p_i. Let $\boldsymbol{p} = (p_1, \cdots, p_n)$ be a price vector. Denote by product 0 the no-purchase option which is always available. At a generic period, the firm selects assortment S from the potential product set $[n]$. The offered assortment is encoded as a vector $S = (x_1, x_2, ..., x_n) \in \{0,1\}^n$, where $x_i = 1$ indicates that product i is offered and $x_i = 0$ indicates that product i is not offered.

All customers belong to a collection of customer types $\mathcal{F}$. An arriving customer is of type $g \in \mathcal{F}$ with probability λ_g. Let $\boldsymbol{\lambda} \in \Lambda = \{\boldsymbol{\lambda} : \sum_{g\in\mathcal{F}} \lambda_g = 1, 0 \le \lambda_g \le 1, \forall g \in \mathcal{F}\}$ be a *type probability distribution* over all purchase patterns in $\mathcal{F}$. We assume that the purchase or choice pattern of each type $g \in \mathcal{F}$ customer can be captured by any non-parametric choice model, where the selected product $c(S,g)$ is deterministic rather than randomly chosen for a given customer type g and assortment S. Denote by $I_{i,S} = \{g \in \mathcal{F} : c(S,g) = i\}$ the index set of the types of customers who buy product i when the set of products S is offered. Note that $\{I_{i,S}\}_{i\in S\cup\{0\}}$ are mutually exclusive sets and $\cup_{i\in S\cup\{0\}} I_{i,S} = \mathcal{F}$. Let $I = \{I_{i,S} : \forall i \in S, S \in \mathcal{S}\}$ be a collection of all index sets. For any index set $I \in I$, denote by $\lambda(I) = \sum_{g\in I} \lambda_g$ its total probability, which is the sum of the probabilities of the types of customers in the set. It is worthwhile to point out that $\lambda(I_{i,S}) = 0$ if $i \in [n] \setminus S$ and $\sum_{i\in S\cup\{0\}} \lambda(I_{i,S}) = 1$. Given any offered assortment $S \in \{0,1\}^n$, define $\mathbf{v}_S(\boldsymbol{\lambda}) = (v_{0,S}(\boldsymbol{\lambda}), v_{1,S}(\boldsymbol{\lambda}), \cdots, v_{n,S}(\boldsymbol{\lambda}))^T = (\lambda(I_{0,S}), \lambda(I_{1,S}), \cdots, \lambda(I_{n,S}))^T \in [0,1]^{n+1}$ as a *purchase probability vector* for all products given assortment S and a known type probability distribution $\boldsymbol{\lambda}$. It is important to note that learning the purchase probability vector is generally more straightforward than estimating the type probability distribution using transaction data.

It is worthwhile to point out that we consider a general non-parametric choice model, including the rank-based choice model, the tree-based choice model, and their mixtures. Throughout this paper, assume that the size of any offered assortment is equal to K ($K \leq n$), which represents either the capacity of an offered assortment or the consideration set of a customer, i.e., $\sum_{i\in[n]} x_i \leq K$. Denote by $\mathcal{S} = \{S \in \{0,1\}^n : \sum_{i\in[n]} x_i \leq K\}$ all applicable assortments. Assume that there is an ample supply of inventory available.

3.2 Optimization Problem and Performance Metric

The expected revenue with the assortment S is given by $R(S,\boldsymbol{\lambda}) = \boldsymbol{p}\mathbf{v}_S(\boldsymbol{\lambda})$. We aim to find an optimal product assortment that maximizes the expected revenue for any arriving customer, without prior information regarding customer types. The assortment optimization problem under the nonparametric choice model then can be formulated as

$$\max_{S\in\mathcal{S}} \quad R(S,\boldsymbol{\lambda}) = \boldsymbol{p}\mathbf{v}_S(\boldsymbol{\lambda}) \tag{1}$$

which has been proven to be *NP-hard* by Feldman et al. (2019).

Practically, the firm does not know the true type probability distribution $\boldsymbol{\lambda}^0 \in \Lambda$. Therefore, we need to learn the purchase probability vector $\mathbf{v}_S(\boldsymbol{\lambda})$ by offering product assortments and maximize the expected revenue over the finite horizon T. Given history information $\mathcal{H}_t = \{S_1, \cdots, S_{t-1}, c_1, \cdots, c_{t-1}\}$, decision mapping $\pi_t : \{0,1\}^{n\times(t-1)} \times [n]^{t-1} \to \{0,1\}^n$ yields an assortment decision at period t. Denoted by $\pi = (\pi_1, ..., \pi_T)$ an admissible policy over the finite horizon T. We measure the performance of an admissible policy π through its regret, defined as

$$\mathrm{Reg}^{\pi}(T,\boldsymbol{\lambda}^0) = \sum_{t=1}^{T} \left\{ R\left(S^*,\boldsymbol{\lambda}^0\right) - \mathbb{E}^{\pi}\left[R\left(S_t^{\pi},\boldsymbol{\lambda}^0\right)\right]\right\}$$

where $\mathbb{E}^{\pi}\left[\sum_{t=1}^{T} R\left(S_t^{\pi},\boldsymbol{\lambda}^0\right)\right]$ is the expected cumulative revenue of the policy and S^* is the optimal assortment obtained from $\max_{S\in\mathcal{S}} R(S,\boldsymbol{\lambda}^0)$. $\mathbb{E}^{\pi}$ is the expectation operator given policy π. Our objective is to propose an admissible policy π that prescribes a sequence of history-dependent assortments $(S_1^{\pi}, S_2^{\pi}, ..., S_T^{\pi})$ aimed at minimizing regret. The worst-case regret is $\mathrm{Reg}^{\pi}(T) = \max_{\boldsymbol{\lambda}\in\Lambda} \mathrm{Reg}^{\pi}(T,\boldsymbol{\lambda})$.

There are two key challenges in solving the above learning and assortment optimization problem. At first, unlike the parametric choice models like MNL where parameters of each product can be estimated separately (Agrawal et al. (2019)), customer types in nonparametric choice models can only be observed collectively through purchase data. This complicates the process of gaining a comprehensive understanding of nonparametric choices, especially when data is limited. Furthermore, assortment optimization problems under any fully rank-based or tree-based choice model are classified as *NP-hard* binary optimization problems. This classification indicates that deriving structural insights about the optimal assortment is unlikely without imposing additional assumptions on the nonparametric choice model. This challenge pertains to the computability of algorithms. Secondly, if we approach the problem with an approximate solution to reduce

computational complexity, there will inevitably be a revenue gap between the optimal solution and the approximate one. This discrepancy results in a linear increase in total regret. This situation can be likened to a seesaw: prioritizing algorithm performance can significantly compromise computability, while enhancing solvability will inevitably introduce a linear term in the regret performance. Therefore, it is challenging for the firm to balance the trade-off among exploration, exploitation, and computational complexity.

In the sequel, we first concentrate on estimating the purchase probability vector using a UCB-based method, temporarily ignoring concerns about computational complexity. Following this, we will address the issue of computational tractability in determining the optimal assortment set by applying a subgradient-based approach, which could reach a sublinear regret for some special condition, but linear regret for generalized conditions.

4 UCB-based Method on Learning Purchase Probabilities

4.1 Constructing Linear Confidence Space

As the first step, we use the transaction data to estimate the purchase probability vector. Based on the history information at a generic period, we can observe the number of any assortment S offered in the data set, defined as $n(S)$. Let $\tilde{\mathbf{v}}_S(\boldsymbol{\lambda})$ be the empirical frequency of product sales, which can be viewed as the realized purchase probability vector given assortment S. For example, consider the assortment $S = (1,1,0,0)$ which has been offered 20 times in the data set. In this case, the no-purchase option was selected five times, product 1 was purchased ten times, and product 2 was purchased five times. Consequently, the empirical frequency of product sales given assortment S is represented as $\tilde{\mathbf{v}}_S = (\frac{5}{20}, \frac{10}{20}, \frac{5}{20}, 0, 0)^\top$.

Taking each assortment as an arm, we can simply have $\|\tilde{\mathbf{v}}_S(\boldsymbol{\lambda}) - \mathbf{v}_S(\boldsymbol{\lambda})\|_1 \leq O(\sqrt{\frac{\log nT}{n(S)\vee 1}})$ with a high probability by Hoeffding Inequality. Alternatively, the upper bound of the confidence interval decreases as an assortment is offered more often. The choice of an assortment is crucial to accelerate the estimation process. However, as the number of applicable assortments increases exponentially with n, this results in inefficiencies in traditional bandit methods concerning both regret performance and computational complexity.

To efficiently utilize the information gathered from the observation sets, we restrict attention to the index sets, as the probability of an index set can be estimated by various offered assortments. In the previous example, the offered assortment is $S = (1,1,0,0)^T = \{1,2\}$ and the corresponding purchase probability vector is represented as $\mathbf{v}_S(\boldsymbol{\lambda}) = (\lambda_1 + \lambda_2, \lambda_3 + \lambda_4, \lambda_5 + \lambda_6, 0, 0)^T$. This allows us to observe the value of $\lambda_3 + \lambda_4$ due to the purchase of product 1. If we offer a different assortment $S' = (0,1,1,0)^T = \{2,3\}$, we obtain its corresponding purchase probability $\mathbf{v}_{S'}(\boldsymbol{\lambda}) = (\lambda_1 + \lambda_2, 0, \lambda_5 + \lambda_6, \lambda_3 + \lambda_4, 0)^T$. The value of $\lambda_3 + \lambda_4$ can be observed again if a product 3 is sold. This dual observation enables us to reconstruct the confidence interval for λ_g values. The above observation suggests that, given a specific assortment, the corresponding index sets can be learned not only through the offer of that assortment but also by any other assortments with a similar index set structure. Therefore, it is unwise to treat each assortment as a separate offer, because

the execution of one assortment not only provides insights into its own performance but also enhances the understanding of other assortments that share similar index structures. This can be viewed as an example of cross-learning among assortments. As a result, it is more efficient to learn the probabilities of the index sets than the probability vectors of the assortments. Let $\kappa = |I|$ be the total number of index sets, which captures the complexity of the learning problem under our non-parametric choice model. Generally speaking, κ can be as large as $K|\mathcal{S}|$.

To gain a better understanding of κ, we present the following examples to illustrate some structures of κ. The customer preference lists in Examples 1 and 2 belong to the rank-based choice model, while the customer choice behavior in Example 3 follows a tree-based structure. In the rank-based choice model, we apply a consider-then-choose decision process proposed by Wang (2022) where consumers first form a consideration set by using some dominance rules and then choose a product within the consideration set. Specifically, if a superior alternative (e.g., with significantly high quality or low price) is present, some inferior options (e.g., with significantly low quality or high price) may never be chosen or even considered. Denote $[n]$ as the set of all products ranked by quality with higher quality corresponding to a higher price and S as the offer set. Similar to Wang (2022), we define Δ_Q to measure the tolerance level of quality, that is, if the quality rank of product i's is lower than both the rank of the highest-quality product and the rank of the next tolerance level of products, then product i is excluded from the consideration set. For instance, $\Delta_Q = 0$ means the consideration set only contains the alternative with the highest quality level, while $\Delta_Q = n-1$ means the consideration set contains all products. Additionally, consumers may have a budget constraint meaning that consumers cannot afford to buy expensive products designated as Δ_B where $\Delta_B = 0$ means that consumers cannot afford to buy any products while $\Delta_B = n$ means there is no budget constraint. Subsequently, we use the tolerance level of quality and budget constraint to illustrate some examples of κ.

EXAMPLE 1. (Rank-based) Consider an example with $[n] = \{1, 2, ..., 10\}$ ranked according to quality, where higher quality corresponds to a higher price. Assume that customers have budget constraints but can tolerate all quality levels of products, that is $\Delta_Q = n-1$. Besides, different customer types have different budget constraints and thus there are ten distinct customer types with a gradually tightening budget constraint as shown in Table 1(a). The rank list is ordered by quality meaning that customers prefer the product with the highest quality. For example, customer type one has no budget constraint and will buy product one if offered while customer type ten has extremely limited budget constraint and can only buy product ten.

Recall that κ represents the size of the index sets in $I = \{I_{i,S} : \forall i \in S, S \in \mathcal{S}\}$. We present all index sets in Table 1(b). For instance, when the firm observes the purchase of product one, it can be inferred that the consumer belongs to type g_1 since her ranking list is $\{1, 2, \cdots, 10\}$ for any potential offer set S containing product one. In other words, if the offer set includes product one but no customer purchases it, the index

Table 1 Example 1: Rank-based Choice Model Illustration

(a) Customer rank lists

Customer type	Rank list	Budget constraint
g_1	$\{1,2,3,...,9,10\}$	$\Delta_B = 10$
g_2	$\{2,3,...,9,10\}$	$\Delta_B = 9$
$\vdots$	$\vdots$	
g_9	$\{9,10\}$	$\Delta_B = 2$
g_{10}	$\{10\}$	$\Delta_B = 1$

(b) Index sets of customer type

Purchased product	Potential offered set S	Index set of customer type
1	$\{1\}$	$\{g_1\}$
2	$\{2\}$	$\{g_1, g_2\}$
	$\{1,2\}$	$\{g_2\}$
3	$\{3\}$	$\{g_1, g_2, g_3\}$
	$\{1,3\}$	$\{g_2, g_3\}$
	$\{1,2,3\}$	$\{g_3\}$
$\vdots$	$\vdots$	$\vdots$
10	$\{10\}$	$\{g_1, g_2, ..., g_9, g_{10}\}$
	$\{1,10\}$	$\{g_2, g_3, ..., g_{10}\}$
	$\vdots$	$\vdots$
	$\{1,2,...,9,10\}$	$\{g_{10}\}$

set certainly does not include the type g_1 customer. From Table 1(b), we list the index set of customer type and can calculate the number of distinct index sets is $1+2+3+...+10=55$. In the general case with n products and n customer types, if the purchased product i is observed and the offer set is $\{i\}$, customers who rank product j before product i will purchase product i since product i is the highest quality in their consideration set. However, if product j ranked before product i is offered in the assortment, the index set of customer type will exclude customer types whose rank lists include product j. Therefore, the number of all index sets is $\kappa = n+(n-1)+...+1 = \frac{(n+1)n}{2}$.

EXAMPLE 2. (Rank-based) Consider an example with $[n] = \{1,2,...,5\}$ ranked according to quality, where higher quality corresponds to a higher price. Assume that customers have budget constraints and can only allow for two products that rank just below the highest quality product, that is $\Delta_Q = 2$. Thus, there are three types of customers shown in Table 2(a). For example, the type g_1 customer prefers the highest quality product and only tolerates two products next to the highest quality product while the type g_2 customer cannot afford to buy the highest quality product and thus the rank list starts from product two. We then present index sets in Table 2(b) and can calculate $\kappa = 6$. Note that product three is included in all customer types' rank list. Therefore, if product three is offered and observed to be purchased, the index set includes all customer types. In the general case with n products and $n-2$ customer types since the quality tolerance level is two products, from product three to product $n-2$, each product observed to be purchases has three index sets of customer type. Therefore, κ includes $3(n-2-2)$ plus the three index sets when product one and n are offered and observed to be purchased, that is, $\kappa = 3(n-4)+3 = 3n-9$.

EXAMPLE 3. (Tree-based) The tree-based choice model has been well documented in marketing, psychology and economics field as pointed out by Chen and Mišić (2021). One of the most well-known irrational behaviors is referred to as the compromise effect. This phenomenon indicates that customers are more likely to purchase a specific product when it is positioned as a compromise option—specifically, when a similar

Table 2 Example 2: Rank-based Choice Model Illustration

(a) Customer rank lists

Customer type	Rank list	Budget constraint
g_1	$\{1,2,3\}$	$\Delta_B = 5$
g_2	$\{2,3,4\}$	$\Delta_B = 4$
g_3	$\{3,4,5\}$	$\Delta_B = 3$

(b) Index sets of customer type

Purchased product	Potential offered set S	Index set of customer type
1	$\{1\}$	$\{g_1\}$
2	$\{2\}$	$\{g_1, g_2\}$
	$\{1,2\}$	$\{g_2\}$
	$\{3\}$	$\{g_1, g_2, g_3\}$
3	$\{1,3\}$	$\{g_2, g_3\}$
	$\{2,3\}$	$\{g_3\}$
4	$\{4\}$	$\{g_2, g_3\}$
	$\{2,4\}$	$\{g_3\}$
5	$\{5\}$	$\{g_3\}$

product is introduced into the offering (Simonson 1989). To illustrate the compromise effect, we consider the following tree structure. Each customer type maintains a preference list based on quality rankings, which

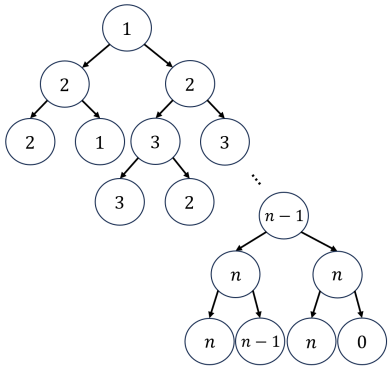


Figure 1 A Tree-based Choice Model with Compromise Effect

can be deduced from the right branch of the tree in Figure 1. However, when evaluating a product i that is included in the offer set, the customer does not make an immediate purchase decision. Instead, they consider whether it would be possible to buy product $i+1$ instead. In this context, product $i+1$ can be seen as a compromise for product i, leading to the existence of the compromise effect between these adjacent products. Consider a type g_i customer with $\Delta_B = n - i + 1$ who begins his/her decision path from product i. We present the index sets in Table 3. The analysis of Example 3 is similar to Example 1 since the customer types shows a nested structure, for example, the type g_i customer behaves exactly the same as the type g_{i+1} when product i is not offered. For a general case with n products and n customer types, if the purchased product is i and product $i+1$ is not offered, all customer types $g_{j'}(j' \leq j)$ would be excluded from the index set whenever some product $j \leq i-2$ is offered. If product $i+1$ is offered, all the index sets would exclude type g_i since the type g_i customer would view product $i+1$ as a compromise and buy it instead of product i. Specifically, when the offer set contains $\{i-1, i\}$ without product $i+1$, the type g_{i-1} customer would view

Table 3 Example 3: Tree-based Choice Model Illustration

(a) Index sets of customer type

Purchased product	Potential offered set S	Index set of customer type
1	$\{1\}$	$\{g_1\}$
2	$\{2\}$	$\{g_1, g_2\}$
	$\{1,2\}$	$\{g_1, g_2\}$
	$\{2,3\}$	$\{g_1\}$
	$\{1,2,3\}$	$\{g_1\}$
3	$\{3\}$	$\{g_1, g_2, g_3\}$
	$\{1,3\}$	$\{g_2, g_3\}$
	$\{1,2,3\}$	$\{g_2, g_3\}$
	$\{3,4\}$	$\{g_1, g_2\}$
	$\{1,3,4\}$	$\{g_2\}$
	$\{1,2,3,4\}$	$\{g_2\}$
$\vdots$	$\vdots$	$\vdots$
i	$\{i\}$	$\{g_1, g_2, ..., g_{i-1}, g_i\}$
	$\{1,i\}$	$\{g_2, ..., g_{i-1}, g_i\}$
	$\vdots$	$\vdots$
	$\{1,2,...,i-2,i\}$	$\{g_{i-1}, g_i\}$
	$\{1,2,...,i-1,i\}$	$\{g_{i-1}, g_i\}$
	$\{i,i+1\}$	$\{g_1, g_2, ..., g_{i-1}\}$
	$\{1,i,i+1\}$	$\{g_2, ..., g_{i-1}\}$
	$\vdots$	$\vdots$
	$\{1,2,...,i-1,i,i+1\}$	$\{g_{i-1}\}$
$\vdots$	$\vdots$	$\vdots$

product i as a compromise and buy it. As a result, for the purchased product i, there would be i new index sets to be counted: $\{g_1, g_2, ..., g_i\}, \{g_2, ..., g_i\}, \{g_{i-1}, g_i\}, \{g_{i-1}\}$, and κ equals to $\frac{n(n+1)}{2} - 1$ for $n \geq 2$.

These examples illustrate that under some rational or irrational conditions, when customers face constraints such as budget and quality tolerance, κ is polynomial with the size of the potential product set n. This significantly reduces the complexity of estimating probabilities. Denote by $\tilde{\lambda}_t(I)$ and $n_t(I)$ the empirical estimation and observation times of the index set $I \in \mathcal{I}$ at the beginning of period t, respectively. The confidence interval of $\tilde{\lambda}_t(I)$ can be alternatively formulated in the following lemma.

LEMMA 1. *The convergence of* $\tilde{\lambda}_t(I)$ *to* $\lambda^0(I)$ *is given by*

$$\mathbb{P}\left\{\exists t \in [T], I \in \mathcal{I} : |\tilde{\lambda}_t(I) - \lambda^0(I)| > \sqrt{\frac{\log 2\kappa T}{n_t(I) \vee 1}}\right\} \leq \frac{1}{T}. \tag{2}$$

Lemma 1 suggests that for any period $t \in [T]$ and index set $I \in \mathcal{I}$, $|\tilde{\lambda}_t(I) - \lambda^0(I)|$ can be bounded within $\sqrt{\frac{\log 2\kappa T}{n_t(I)\vee 1}}$ with a high probability. As a result, we can define linear confidence space $\Lambda_t = \{\boldsymbol{\lambda} \in \Lambda : |\tilde{\lambda}_t(I) - \lambda(I)| \leq \sqrt{\frac{\log 2\kappa T}{n_t(I)\vee 1}}\}$ for some fixed t. Due to the linear bounds, Λ_t is a convex set and contains true type probability distribution $\boldsymbol{\lambda}^0$ with a high probability according to Lemma 1.

4.2 LPUCB Algorithm

Based on the confidence space Λ_t constructed in the previous section, we can calculate the UCB of the expected revenue for any assortment $S \in \mathcal{S}$ through the following linear program, which is defined as **Problem $\mathbf{P}_t(S)$**.

$$\begin{aligned} \max_{\lambda \in \Lambda_t} \quad & \sum_{i \in S} p_i v_{i,S}(\lambda) \\ \text{s.t.} \quad & \lambda(I) \geq \tilde{\lambda}_t(I) - \sqrt{\frac{\log 2\kappa T}{n_t(I) \vee 1}}, \quad \forall I \in \mathcal{I} \\ & \lambda(I) \leq \tilde{\lambda}_t(I) + \sqrt{\frac{\log 2\kappa T}{n_t(I) \vee 1}}, \quad \forall I \in \mathcal{I} \\ & \lambda_g \geq 0, \quad \sum_{g \in \mathcal{F}} \lambda_g = 1. \end{aligned} \tag{3}$$

In **Problem $\mathbf{P}_t(S)$**, the number of linear constraints is κ and the number of variables is $|\mathcal{F}|$. As suggested by Lemma 1, the confidence space Λ_t becomes smaller over time. In Algorithm 1, the firm offers the assortment that yields the largest UCB of the expected revenue at each period. We refer to this algorithm as Linear Programming Upper Confidence Bound (LPUCB).

Algorithm 1 Linear Programming Upper Confidence Bound

Require: Initialize $S_1 \in \{0,1\}^n$ as a randomized set

1: Offer S_1 and observe the actual purchase c_1.
2: Update history information H_2 by tuple (S_1, c_1).
3: **for** $t = 2, \ldots T$ **do**
4: Infer $\tilde{\lambda}_t(I)$ and $n_t(I)$ empirically from H_t.
5: Construct the confidence space Λ_t by $\tilde{\lambda}_t(I)$ and $n_t(I)$ for all $I \in \mathcal{I}$ as in (3).
6: Iterate over $\mathcal{S}$ to solve **Problem $\mathbf{P}_t(S)$** for all $S \in \mathcal{S}$ and select S_t at

$$S_t = \arg\max_{S \in \mathcal{S}} \left(\max_{\lambda \in \Lambda_t} \sum_{i \in S} p_i v_{i,S}(\lambda) \right).$$

7: Offer S_t and observe the actual purchase c_t.
8: Update history information H_{t+1} by tuple (S_t, c_t).
9: **end for**

The LPUCB algorithm is a bandit-like method that treats each assortment $S \in \mathcal{S}$ as an arm or action, updating the upper bound of the expected reward for each arm based on historical observations. Unlike the traditional UCB1 algorithm (Auer et al. 2002), which considers each arm independently, Algorithm 1 calculates the upper bound of the expected reward within the confidence space by **Problem $\mathbf{P}_t(S)$** for all

$S \in \mathcal{S}$. Since observation (S_t, c_t) provides information on both probabilities $\boldsymbol{\lambda}$ and assortment S_t, UCB1 algorithm alone is inefficient in utilizing this historical information. In contrast, Algorithm 1 fully leverages the available data to infer the maximum potential reward of any assortment.

4.3 Theoretical Performance

Next, we derive the theoretical performance of Algorithm 1 in the following result.

THEOREM 1. *For any type probability distribution $\boldsymbol{\lambda}^0 \in \Lambda$ with capacity constraint K, assuming $p_i \in [0,1]$ for any $i \in [n]$ and $\kappa \leq T$, the regret generated from the LPUCB algorithm (Algorithm 1) can be bounded as*

$$\begin{aligned} \mathrm{Reg}^{\pi}(T, \boldsymbol{\lambda}^0) &\leq 4\sqrt{\kappa K T \log 2\kappa T} \\ &= O(\sqrt{\kappa K T \log \kappa T}). \end{aligned} \tag{4}$$

In the proof of Theorem 1, we bound the reward UCB calculated by **Problem** $\mathbf{P}_t(S)$ by taking each index set I as an arm. That is, we break the reward UCB of any assortment S into the UCBs of each index set in the assortment and then aggregate them together to obtain the overall performance of LPUCB algorithm. It is important to note that the $\sqrt{\kappa}$ order in Theorem 1 arises from this process.

We next discuss the results presented in Theorem 1. Interestingly, if the set $\mathcal{F}$ consists of n customer types with each type considering the purchase of exactly one distinct product, then $\kappa = n$ and each customer type can be estimated independently. In this case, the dynamic assortment problem is equivalent to the simplest combinatorial bandit problem with semi-bandit feedback, where the revenue generated by each decision is only dependent on that specific decision itself. Therefore, the LPUCB algorithm calculates the UCB of the expected revenue for each product and selects an assortment that includes the products with the highest UCB values. As mentioned in Exercise 30.5 in Lattimore and Szepesvári (2020), the lower bound of the combinatorial bandit problem with semi-bandit feedback is established as $\Omega(\sqrt{nKT})$, The regret achieved by the LPUCB algorithm, as shown in Theorem 1, approaches this near-optimal performance.

In more complex scenarios, as illustrated in Example 2, we find that $\kappa = O(n)$. In these cases, the regret of the LPUCB algorithm remains near-optimal at $\Omega(\sqrt{nKT})$. However, in Example 1, $\kappa = O(n^2)$ resulting in a regret higher on the order of n. Nonetheless, $\Omega(\sqrt{nKT})$ remains the lower bound of the regret for the combinatorial bandit problem with semi-bandit feedback. Therefore, the complexity of the dynamic assortment problem under a non-parametric choice model can become significantly greater. To some extent, κ can serve as a valuable indicator of problem complexity and significantly influences the best achievable performance. In addition, it is important to note that the LPUCB algorithm is applicable to a wide range of non-parametric choice models, extending beyond just rank-based or tree-based models.

While the algorithm's effectiveness across various non-parametric choice models is impressive, computational complexity remains a critical issue to address. Although solving each **Problem** $\mathbf{P}_t(S)$ is computationally efficient for a given assortment S, there are $|\mathcal{S}| = \sum_{k=1}^{K} n!/k!(n-k)!$ possible assortments at each period.

This number increases dramatically with both n and K, leading to significant computational demands as the size of the problem grows. Next, we will explore subgradient-based methods that can effectively solve the assortment optimization problem while simultaneously learning the purchase probability vector efficiently.

5 Subgradient-based Method on Optimizing Assortments

5.1 bOGD-EG Algorithm

In this section, we propose Algorithm 2, called Binary Online subGradient Descent with Estimated subGradient (bOGD-EG), which is a subgradient-based method. It is important to note that, Algorithm 2 reduces

Algorithm 2 Binary Online subGradient Descent with Estimated Gradient

Require: Initialize $\tilde{S}_1 \in [0,1]^n$ and step size $\eta_t = \frac{1}{\sqrt{t}}$.

1: **for** $t = 1, ...T$ **do**

2: Offer $S_t = \mathcal{R}(\tilde{S}_t)$ and observe the actual purchase c_t.

3: Update history information H_{t+1} by tuple (S_t, c_t).

4: Infer $\tilde{\lambda}_t(I)$ and $n_t(I)$ empirically from H_{t+1}.

5: Construct the confidential space Λ_t by $\tilde{\lambda}_t(I)$ and $n_t(I)$ for all $I \in \mathcal{I}$.

6: Obtain $\hat{\nabla}_t$ from $\mathbf{SG}(\tilde{S}_t; \hat{\boldsymbol{\lambda}}_t)$ for some feasible $\hat{\boldsymbol{\lambda}}_t$ selected from Λ_t.

7: Set $\tilde{S}_{t+1}$ at

$$\tilde{S}_{t+1} = \Pi_{[0,1]^n}(\tilde{S}_t + \eta_t \hat{\nabla}_t).$$

8: **end for**

the computational complexity by identifying the relaxed optimal assortment $\tilde{S}^* \in [0,1]^n$ rather than searching the exact optimal assortment S^*. Alternatively, we view $\tilde{S}^*$ as a probability vector, indicating the likelihood of selecting each product. We further use simple randomization by defining $S_t = \mathcal{R}(\tilde{S}_t)$ where for each product $i \in [n]$, product i is included in the assortment $i \in S_t$ with a probability of $\tilde{x}_{i,t}$. Here $\tilde{S}_t = (\tilde{x}_{1,t}, ..., \tilde{x}_{n,t})$ represents the relaxed assortment. Note that this randomization process does not limit the output assortment S_t to fewer than K products. We first examine the performance of the bOGD-EG algorithm without the capacity constraint and subsequently incorporate the capacity constraint. The randomization process is carried out using a subgradient-based method, where the relaxed assortment $\tilde{S}$ is iteratively updated to approximate $\tilde{S}^*$ by leveraging subgradient information. Note that in the algorithm, $\Pi_{\mathcal{Y}}(x) = \arg\min_{y \in \mathcal{Y}} \|x - y\|_2$ represents the Euclidean projection operator onto the set $[0,1]^n$. Although the exact value of the gradient is unknown due to insufficient information about $\boldsymbol{\lambda}^0$, we select an estimated vector $\hat{\boldsymbol{\lambda}}_t$ that lies within the feasible set Λ_t. This allows us to estimate the gradient and ensures that it converges to its true value over

time. However, even with the relaxed optimal assortment $\tilde{S}^*$, we are still unable to determine the exact optimal assortment S^*. Alternatively, the trade-off for reducing computational complexity is a potential loss in revenue resulting from rounding the relaxed assortment to a feasible assortment.

Next, we introduce the **SubGradient** (**SG**) Problem given the estimated vector $\hat{\boldsymbol{\lambda}}_t$ and the relaxed assortment $\tilde{S}_t$. The primary goal of the SubGradient Problem is to compute the gradient $\hat{\nabla}_t$. It is crucial to understand that the formulation of the SubGradient Problem is significantly influenced by the structure of the choice models. Below, we will outline the SubGradient Problems for both fully rank-based and tree-based choice models.

We first consider a fully rank-based choice model. In a rank-based choice model, the purchase pattern σ_g is defined as the preference list of a type g customer, where σ_g specifies a rank ordering over $[n] \cup \{0\}$ and $\sigma_g(i)$ represents the preference rank of product i (Jagabathula and Vulcano 2018). A lower ranking indicates a larger preference. Alternatively, $\sigma_g(a) < \sigma_g(b)$ indicates that each type g customer has a preference for product a over product b. Conversely, g^j is defined as the product in the jth position on the preference list of a type g customer, indicating the jth most preferred product for that customer type. The notation $|g|$ denotes the rank length of customer type g. We define SubGradient Problem **SG**-*rank*$(\tilde{S}_t, \hat{\boldsymbol{\lambda}}_t)$ as follows,

$$\begin{aligned}
\max_{y} \quad & \sum_{g\in\mathcal{F}} \sum_{j=1}^{|g|} \hat{\lambda}_{g,t} p_{g^j} y_{g,g^j} \\
\text{s.t.} \quad & y_{g,g^j} \le \tilde{x}_{g^j,t} \qquad \forall g \in \mathcal{F}, j \in \{1,...,|g|\}; \\
& y_{g,g^j} \le 1 - \tilde{x}_{g^k,t} \quad \forall g \in \mathcal{F}, j \in \{2,...,|g|\}, k \in \{1,...,j-1\}; \\
& y_{g,g^j} \ge 0 \qquad \forall g \in \mathcal{F}, j \in \{1,...,|g|\}.
\end{aligned} \tag{5}$$

In the above SubGradient Problem **SG**-*rank* which is obviously a LP problem, variable y_{g,g^j} represents the probability that customer type g will purchase product j and the objective function is to maximize the total expected revenue. The first constraint indicates that the purchase probability of product g^j by a type g customer cannot exceed the probability of offering product g^j. The second constraint implies that the purchase probability of product g^j cannot exceed the probability of not offering products $g^k (k \le j-1)$ that are preferred more than product g^j. It is important to note that the problem **SG**-*rank* can be solved efficiently for given $\tilde{S}_t$ and $\hat{\boldsymbol{\lambda}}_t$. This enables us to derive the marginal impact of $\tilde{S}_t$ on the expected revenue, despite an approximate one. However, it is essential to recognize that the optimized objective of problem **SG**-*rank* does not represent the true expected revenue based on the provided $\tilde{S}_t$ and $\hat{\boldsymbol{\lambda}}_t$. Unlike the LP-relaxed version of (IP1) in Feldman et al. (2019) which focuses on finding optimal assortments, our approach aims to utilize the subgradient of $\tilde{S}_t$ within the SubGradient Problem.

Secondly, we move to a fully tree-based choice model. We apply a directed binary tree to formulate the purchase decision tree as used in Chen and Mišić (2021), illustrated in Figure 2. Denote by **leave**(g) and **splits**(g) the set of leaf nodes and the set of split nodes of a tree g. For any split node j, denote by **left**(j) and

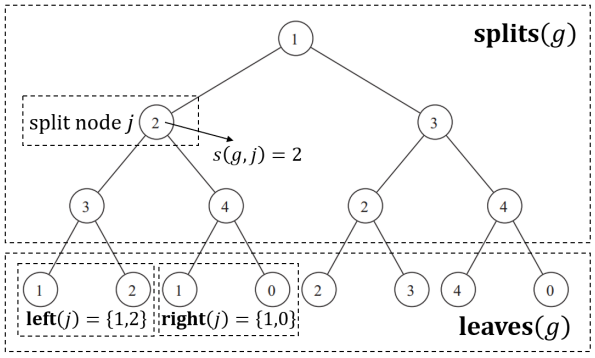


Figure 2 An Illustrated Example of the Tree-based Choice Model

right(j) its sets of left and right leaf nodes, respectively. Each customer begins at the root node of the tree. If the product associated with the root node is included in S, the customer proceeds to the left child node; otherwise, they move to the right child node. This process continues until a leaf node is reached, where the product represented at that node signifies the customer's final decision. To formulate this process, the SubGradient Problem **SG**-*tree*$(\tilde{S}_t, \hat{\boldsymbol{\lambda}}_t)$ is defined as the following

$$
\begin{aligned}
\max_{y} \ & \sum_{g\in\mathcal{F}} \sum_{\ell\in\textbf{leave}(g)} \hat{\lambda}_{g,t} p_\ell y_{g,\ell} \\
\text{s.t.} \ & \sum_{\ell\in\textbf{left}(j)} y_{g,\ell} \le \tilde{x}_{s(g,j),t} && \forall g\in\mathcal{F}, j\in\textbf{splits}(g); \\
& \sum_{\ell\in\textbf{right}(j)} y_{g,\ell} \le 1-\tilde{x}_{s(g,j),t} && \forall g\in\mathcal{F}, j\in\textbf{splits}(g); \\
& \sum_{\ell\in\textbf{leave}(g)} y_{g,\ell} = 1 && \forall g\in\mathcal{F}; \\
& y_{g,\ell} \ge 0 && \forall g\in\mathcal{F}, \ell\in\textbf{leave}(g).
\end{aligned} \tag{6}
$$

Similar to Problem **SG**-*rank*, Problem **SG**-*tree* is a LP problem designed for the tree-based choice model. For any split node j, the aggregation of the purchase probability of the products corresponding to its left proceeding leaf nodes in **left**(j) has to be smaller than the probability that the corresponding product $s(g,j)$ to split node j is offered. Symmetrically for its right proceeding leaf nodes in **right**(j), the purchase probability of corresponding products has to be smaller than the probability where $s(g,j)$ is *not* offered. Problem **SG**-*tree* shares a similar formulation with the LP-relaxed version of **SplitMIO** as described in Chen and Mišić (2021), but it differs in terms of its variables and objectives.

Let $f(\tilde{S}, \hat{\boldsymbol{\lambda}}_t)$ represent the optimized value of the objective function for any of the aforementioned SG problems for given $\hat{\boldsymbol{\lambda}}_t$ and $\tilde{S}\in[0,1]^n$. Here, $\tilde{S}$ represents the relaxed assortment located in a hypercube space rather than an integer set. We can show the concavity property of $f(\tilde{S}, \hat{\boldsymbol{\lambda}}_t)$.

LEMMA 2. *Given fixed value of* $\boldsymbol{\lambda}\in\Lambda$, $f(\tilde{S}, \boldsymbol{\lambda})$ *is a piecewise linear concave function on* $\tilde{S}\in\mathbb{R}^n$.

Lemma 2 ensures that there exists at least one maximizer $\tilde{S}^*\in[0,1]^n$ for any $\boldsymbol{\lambda}$. Note that, due to the piecewise linear structure, $f(\tilde{S}, \boldsymbol{\lambda})$ is not differentiable on those points with inconsistent directional derivative.

This is why the subgradient method is employed instead of the gradient method. Define $\nabla f(\tilde{S},\boldsymbol{\lambda})$, $\partial f(\tilde{S},\boldsymbol{\lambda})$, and ∇ as the gradient, subdifferential, and subgradient, respectively. Particularly, if function $f(\tilde{S},\boldsymbol{\lambda})$ is differentiable at $\tilde{S}$, then $\partial f(\tilde{S},\boldsymbol{\lambda})=\{\nabla\}$, $\nabla f(\tilde{S},\boldsymbol{\lambda})=\nabla$, and

$$f(\tilde{S}',\boldsymbol{\lambda})\le f(\tilde{S},\boldsymbol{\lambda})+\nabla^{\top} f(\tilde{S},\boldsymbol{\lambda})(\tilde{S}'-\tilde{S}).$$

While, if $f(\tilde{S},\boldsymbol{\lambda})$ is not differentiable at $\tilde{S}$, subgradient $\nabla\in\partial f(\tilde{S},\boldsymbol{\lambda})$ can be used as a substitute for the gradient, satisfying the following inequality,

$$f(\tilde{S}',\boldsymbol{\lambda})\le f(\tilde{S},\boldsymbol{\lambda})+\nabla^{\top}(\tilde{S}'-\tilde{S}).$$

In Algorithm 2, we iteratively calculate the relaxed assortment $\tilde{S}\in[0,1]^n$ using the subgradient descent method (or ascent for concave functions) to approximate the optimal relaxed assortment $\tilde{S}^*$. We select $\hat{\boldsymbol{\lambda}}_t$ from the set Λ_t. As the set Λ_t converges to the true value $\boldsymbol{\lambda}$, the estimated subgradient $\hat{\nabla}_t$ will also converge to the true subgradient ∇_t. Since $\tilde{S}$ serves as the resource limit in all linear programming constraints, we only need to aggregate the dual prices of the constraints related to x_i while subtracting the dual prices of the constraints associated with $1-x_i$ in order to obtain the gradient of each x_i with respect to the optimized value.

For Problem **SG**-*rank*$(\tilde{S}_t,\hat{\boldsymbol{\lambda}}_t)$, its dual problem is as follows,

$$\begin{aligned}
\min_{\delta^+,\delta^-}\ & \sum_{g\in\mathcal{F}}\sum_{j=1}^{|g|}\left(x_{g^j}\delta^+_{g,g^j}+\sum_{k=1}^{j-1}(1-x_{g^k})\delta^-_{g,g^j,g^k}\right)\\
\text{s.t. }& \delta^+_{g,g^j}+\sum_{k=1}^{j-1}\delta^-_{g,g^j,g^k}\ge\hat{\lambda}_{g,t}p_{g^j}\quad \forall g\in\mathcal{F},j\in\{1,...,|g|\};\\
& \delta^+_{g,g^j},\delta^-_{g,g^j,g^k}\ge 0\qquad\qquad \forall g\in\mathcal{F},j\in\{1,...,|g|\},k\in\{1,...,j-1\}.
\end{aligned}\tag{7}$$

The above δ^+_{g,g^j} represents the shadow price of resource constraint $y_{g,g^j}\le x_{g^j}$ for any $j\in\{1,...,|g|\}$ and δ^-_{g,g^j,g^k} represents the shadow price of constraint $y_{g,g^j}\le 1-x_{g^k}$ for any $j\in\{1,...,|g|\}$ and $k\in\{1,...,j-1\}$. Similarly, the dual problem of Problem **SG**-*tree*$(\tilde{S}_t,\hat{\boldsymbol{\lambda}}_t)$ is given by,

$$\begin{aligned}
\min_{\delta^+,\delta^-,\delta^1}\ & \sum_{g\in\mathcal{F}}\left(\delta^1_g+\sum_{j\in\mathbf{splits}(g)}\left(x_{s(g,j)}\delta^+_{g,j}+(1-x_{s(g,j)})\delta^-_{g,j}\right)\right)\\
\text{s.t. }& \sum_{j:\ell\in\mathbf{left}(j)}\delta^+_{g,j}+\sum_{j:\ell\in\mathbf{right}(j)}\delta^-_{g,j}+\delta^1_g\ge\hat{\lambda}_{g,t}p_\ell,\quad \forall g\in\mathcal{F},\ell\in\mathbf{leave}(g);\\
& \delta^+_{g,j},\delta^-_{g,j}\ge 0\qquad\qquad \forall g\in\mathcal{F},j\in\mathbf{splits}(g).
\end{aligned}\tag{8}$$

The above $\delta^+_{g,j}$ and $\delta^-_{g,j}$ respectively represent the shadow prices on resources $x_{s(g,j)}$ and $1-x_{s(g,j)}$, and δ^1_g is the shadow price of constraint $\sum_{\ell\in\mathbf{leave}(g)}y_{g,\ell}=1$. By aggregating all the dual prices for each product, the estimated gradient vector $\hat{\nabla}_t=(\hat{\nabla}_{t,1},...,\hat{\nabla}_{t,n})$ can be expressed as follows,

$$\begin{aligned}
\hat{\nabla}_{t,i}&=\sum_{g\in\mathcal{F}}\sum_{j\in\{1,...,|g|\}}\left(\mathbf{1}\{g^j=i\}\hat{\delta}^+_{g,g^j}-\sum_{k\in\{1,...,j-1\}}\mathbf{1}\{g^k=i\}\hat{\delta}^-_{g,g^j,g^k}\right),\forall i\in[n],\ \text{(rank-based)};\\
\hat{\nabla}_{t,i}&=\sum_{g\in\mathcal{F}}\sum_{j\in\mathbf{splits}(g)}\mathbf{1}\{g_j=i\}\left(\hat{\delta}^+_{g,j}-\hat{\delta}^-_{g,j}\right),\qquad\qquad \forall i\in[n],\ \text{(tree-based)}.
\end{aligned}\tag{9}$$

It is important to note that if the dual problem of the SubGradient Problem has multiple solutions, this indicates that $f(\tilde{S},\boldsymbol{\lambda})$ is not differentiable at $\tilde{S}$. We then can select any solution for $\hat{\delta}^{+}$ and $\hat{\delta}^{-}$ to construct the subgradient.

Algorithm 2 now can run efficiently by constructing the confidence space Λ_t and solving the dual problem of the SubGradient Problems to obtain the estimated subgradient. The complexity of the algorithm is independent of the number of potential assortments $\binom{n}{K}$ and is instead polynomial related to n, κ, number of customer types $|\mathcal{F}|$, and the customer type length in the rank-based choice model or the number of split and leaf nodes in the tree-based choice model.

5.2 Theoretical Performance

We next move to present the theoretical performance of the subgradient-based Algorithm 2 without the capacity constraint.

THEOREM 2. *For any problem instance $\boldsymbol{\lambda}^0 \in \Lambda$ without the capacity constraint, assume $p_i \in [0,1]$ for any $i \in [n]$ and $\kappa \leq T$, then the regret generated from Algorithm 2 can be bounded as*

$$\begin{aligned} \mathrm{Reg}^{\pi}(T,\boldsymbol{\lambda}^0) &\leq \frac{3}{2}n\sqrt{T} + 4\sqrt{\kappa nT\log 2\kappa T} + \frac{1}{2}nT \\ &= O(n\sqrt{T} + \sqrt{\kappa nT\log \kappa T} + nT). \end{aligned} \tag{10}$$

Theorem 2 reveals that the regret upper bound of bOGD-EG consists of three parts. The first two components $O(n\sqrt{T}+\sqrt{\kappa nT\log\kappa T})$ account for the process of learning the relaxed optimal assortment $\tilde{S}^*$ through estimation. In particular, the first component $O(n\sqrt{T})$s is generated from the Online subGradient Descent (OGD) part of the algorithm, and the second component $O(\sqrt{\kappa nT\log\kappa T})$ is from the subgradient estimation. When κ is polynomial in n, these two components can be combined together to be $O(\sqrt{\kappa nT\log\kappa T})$, which is the same as LPUCB. Therefore, for the estimation part, bOGD-EG achieves good performance as $\tilde{S}_t$ approaches $\tilde{S}^*$.

The third component $O(nT)$ arises from the process of randomizing the relaxed assortment $\tilde{S}_t$. The regret $O(nT)$ is linear on T. This is because the assortment S_t is randomly selected from the relaxed assortment $\tilde{S}_t$. During this randomization process, some expected revenue is inevitably lost because $f(\tilde{S}_t,\boldsymbol{\lambda}) \geq \mathbb{E}[f(S_t,\boldsymbol{\lambda})]$ when $\tilde{S}_t = \mathbb{E}[S_t]$ according to Jensen's Inequality and the concavity of $f(\cdot,\boldsymbol{\lambda})$ as established in Lemma 2. Due to the loss incurred from randomization, the regret increases linearly, which is an inevitable consequence for algorithms based on LP relaxation.

It is essential to note that the bOGD-EG algorithm has significantly reduced computational complexity through the use of LP relaxation. Recall that the LPUCB algorithm uses $\mathbf{P}_t(S)$ for $|\mathcal{S}|$ times in each period, while bOGD-EG only deploys $\mathbf{SG}(\tilde{S}_t,\hat{\lambda}_t)$ once and selects $\hat{\lambda}_t$ from Λ_t once during each period. In particular, when the length of the rank-based choice model and the total number of nodes of the tree-based choice model are bounded, the dual problems of problems **SG** are polynomial in $|\mathcal{F}|$ and the process of selecting

$\hat{\lambda}_t$ from Λ_t is polynomial in both κ and $|\mathcal{F}|$. This indicates that bOGD-EG has significantly improved computational tractability.

From the discussion above, it is clear that computational complexity plays a crucial role when addressing an *NP-hard* problem. Since reducing computational complexity may increase the total regret, as seen with bOGD-EG, we incorporate this factor into the trade-off that the algorithm must balance. LPUCB and bOGD-EG primarily focus on minimizing total regret and optimizing computational complexity, respectively. Under a general instance of nonparametric choice model, the existence of a computationally tractable algorithm that can achieve sublinear regret remains unclear.

5.3 Boundary Condition

As stated in Theorem 2, the key to mitigating regret lies in minimizing the loss incurred from randomization, which increases linearly with respect to T. Taking a simple scenario, when the optimal relaxed assortment lies on the boundary of $[0,1]^n$, that is, $\tilde{S}^* \in \{0,1\}^n$, the loss of randomization approaches zero as $\tilde{S}_t$ converges to $\tilde{S}^*$ (if this occurs). This indicates that the process is converging effectively. This suggests the potential for the randomization component of regret to increase sublinearly, prompting us to identify the precise conditions and establish the regret order for this component.

THEOREM 3. *If $f(\tilde{S}, \boldsymbol{\lambda}^0)$ has a unique maximizer $\tilde{S}^* \in \{0,1\}^n$ (referred to as the boundary condition) and assuming $\kappa \leq T$, we can identify a positive constant $\underline{\nabla}$ that bounds the regret generated from Algorithm 2 given by*

$$\begin{aligned} \mathrm{Reg}^{\pi}(T, \boldsymbol{\lambda}^0) &\leq \left(1 + \frac{2}{\underline{\nabla}}\right)\left(\frac{3}{2} n\sqrt{T} + 4\sqrt{\kappa n T \log 2\kappa T}\right) \\ &= O(n\sqrt{T} + \sqrt{\kappa n T \log \kappa T}). \end{aligned} \tag{11}$$

Theorem 3 shows that when the assumption of $\tilde{S}^*$ being a unique maximizer is satisfied, it guarantees the existence of some gradient $\nabla^* = (\nabla_1^*, ..., \nabla_n^*) \in \partial f(\tilde{S}^*, \boldsymbol{\lambda}^0)$ such that $|\nabla_i^*| > 0$ for all $i \in [n]$. This condition ensures the convergence of $\tilde{S}_t^{\pi}$ to $\tilde{S}^*$ and subsequently leads to a sublinear increase in loss of randomization. This mitigates the $O(T)$ term in the upper bound of Theorem 2 at the expense of imposing the boundary condition and proves the existence of problem instances where bOGD-EG can achieve sublinear regret at $O(\sqrt{T})$ order.

Therefore, this indicates a strong potential for the bOGD-EG algorithm; under the boundary condition, it achieves an optimal trade-off between algorithm performance and computational complexity. In our numerical tests, we will compare both the regret performance and the efficiency performance of both LPUCB and bOGD-EG algorithms under this boundary condition.

5.4 Capacity Constraint

In this section, we revise the randomization process in the bOGD-EG algorithm by limiting the output assortment to fewer than the capacity K and investigate the theoretical performance under the capacitated

condition. Recall that the capacity K helps limit the number of decisions, enhancing efficiency due to its bandit-like nature in the LPUCB. However, it is more efficient to run bOGD-EG without a capacity constraint. This is because the additional effort required in the randomization process to limit the output assortment fewer than K products in bOGD-EG results in increased complexity.

Recall that set $\mathcal{S} = \{S \in \{0,1\}^n : \sum_{i\in[n]} x_i \leq K\}$ includes all applicable assortments. We introduce a modified randomization process, denoted as $\mathcal{R}^C$ to sample an assortment from $\mathcal{S}$ according to a specific distribution. To ensure convergence, this process guarantees that the probability of each product being offered aligns with the existing assortment $\tilde{S}_t = (\tilde{x}_{1,t}, ..., \tilde{x}_{n,t})$. This results in the construction of a distribution over $\mathcal{S}$ based on $\tilde{S}_t$. Denote by $\text{conv}(\mathcal{S})$ the convex hull of $\mathcal{S}$, to which $\tilde{S}_t$ is projected in each period. We define w_S as a distribution over S and w_S satisfies the following linear equations:

$$\begin{cases} \sum_{S\in\mathcal{S}} w_S S = \tilde{S}; \\ \sum_{S\in\mathcal{S}} w_S = 1; \\ \forall S \in \mathcal{S}, w_S \in [0,1]. \end{cases} \tag{12}$$

According to Carathéodory's Theorem, the solution for the distribution w must exist and the support of w will not exceed $n+1$. However, the complexity of $\mathcal{R}^C$ is polynomial in $|\mathcal{S}|$, which may lead to computational intractability. We apply the randomization process $\mathcal{R}^C$ in bOGD-EG and refer to the modified algorithm as Algorithm 3. The following corollary can be easily derived based on Theorems 2 and 3.

Algorithm 3 Binary Online subGradient Descent with Estimated Gradient (Capacity constraint)

Require: Initialize $\tilde{S}_1 \in [0,1]^n$ and step size $\eta_t = \sqrt{\frac{K}{nt}}$.

1: **for** $t = 1, ...T$ **do**
2: Offer $S_t = \mathcal{R}^C(\tilde{S}_t)$ and observe the actual purchase c_t.
3: Update history information H_{t+1} by tuple (S_t, c_t).
4: Infer $\tilde{\lambda}_t(I)$ and $n_t(I)$ empirically from H_{t+1}.
5: Construct the confidential space Λ_t by $\tilde{\lambda}_t(I)$ and $n_t(I)$ for all $I \in I$.
6: Obtain $\hat{\nabla}_t$ from $\mathbf{SG}(\tilde{S}_t; \hat{\boldsymbol{\lambda}}_t)$ for some feasible $\hat{\boldsymbol{\lambda}}_t$ selected from Λ_t.
7: Set $\tilde{S}_{t+1}$ at

$$\tilde{S}_{t+1} = \Pi_{\text{conv}(S)}(\tilde{S}_t + \eta_t \hat{\nabla}_t).$$

8: **end for**

COROLLARY 1. *For any problem instance $\boldsymbol{\lambda}^0 \in \Lambda$ with capacity K, assume $p_i \in [0,1]$ for any $i \in [n]$ and $\kappa \leq T$, then the regret generated from Algorithm 3 can be bounded as*

$$\begin{aligned} \text{Reg}^{\pi}(T, \boldsymbol{\lambda}^0) &\leq 3\sqrt{nKT} + 4\sqrt{\kappa KT \log 2\kappa T} + \sqrt{nKT} \\ &= O(\sqrt{nKT} + \sqrt{\kappa KT \log \kappa T} + \sqrt{nKT}), \end{aligned} \tag{13}$$

and if $\boldsymbol{\lambda}^0 \in \Lambda$ *and* $\boldsymbol{p} \in [0,1]^n$ *jointly ensure a unique maximizer* $\tilde{S}^* \in \{0,1\}^n$ *for* $f(\tilde{S}, \boldsymbol{\lambda}^0)$ *(boundary condition), we can find some positive constant* $\underline{\nabla}$ *to bound the regret generated alternatively from Algorithm 3 given by*

$$\begin{aligned} \mathrm{Reg}^{\pi}(T, \boldsymbol{\lambda}^0) &\leq \left(1 + \frac{2}{\underline{\nabla}}\right)\left(3\sqrt{nKT} + 4\sqrt{\kappa KT \log 2\kappa T}\right) \\ &= O(\sqrt{nKT} + \sqrt{\kappa KT \log \kappa T}). \end{aligned} \tag{14}$$

6 Numerical Analysis

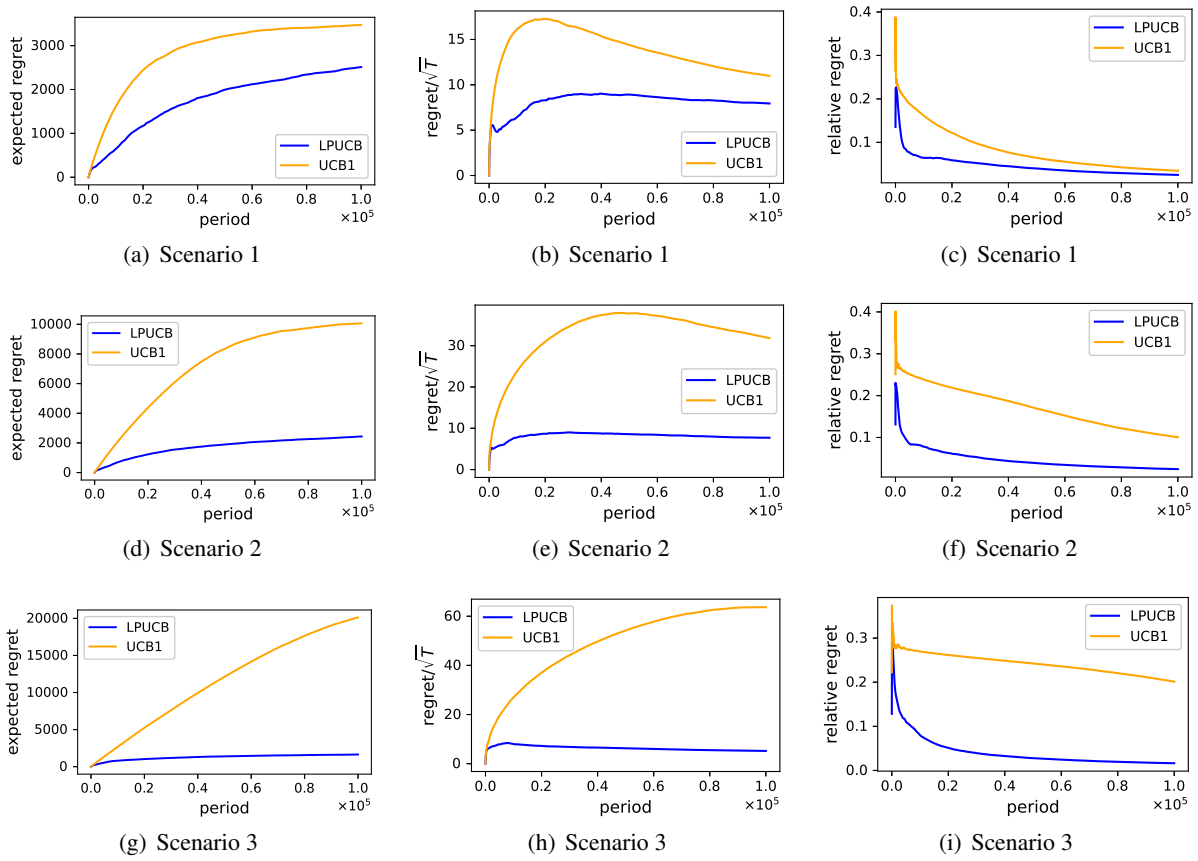
In this section, we conduct a numerical analysis to assess the performance of both the LPUCB and bOGD-EG algorithms. For comparison, we use the widely recognized bandit method UCB1 (Auer et al. (2002)) as our benchmark. We further validate the result in Theorem 1 by investigating the impact of κ on the performance of the algorithms. Finally, we explore the computational efficiency of both LPUCB and bOGD-EG algorithms.

To enhance the computational efficiency of both algorithms, we implement an epoch schedule defined as $0 = \tau_0 < \tau_1 < \tau_2 ... < \tau_m < ...$, which is a method commonly used in the literature. For each $m \in \mathbb{N}^+$, epoch m consists of the periods from $\tau_{m-1} + 1$ to τ_m. During these epochs, S_t^{π} and Λ_t are only updated at the end of each epoch, i.e., $t = \tau_m$. As shown in Theorem 1 of Simchi-Levi and Xu (2022), any epoch schedule adhering to a *doubling-trick*, i.e. $\tau_m \leq 2\tau_{m-1}$ for all m, does not influence the overall order of regret. For the subsequent tests, we will select the epoch schedule as $\tau_m = \lfloor \sqrt{T} \rfloor m$. Specifically, for bOGD-EG, we also update the relaxed assortment at the end of each epoch. Due to the reduced frequency of updates, we allow bOGD-EG to take a larger step with each update, i.e. $\tilde{S}_{t+1} = \Pi_{[0,1]^n}(\tilde{S}_t + (\tau_m - \tau_{m-1})\eta_t \hat{\nabla}_t)$, in which the $\tau_m - \tau_{m-1}$ term is used to compensate for one update in this epoch with length $\tau_m - \tau_{m-1}$.

6.1 Performance of LPUCB

Auer et al. (2002) shows that the expected regret of UCB1 is $\tilde{O}(\sqrt{NT})$, where N denotes the number of total arms (assortments). As the number of the total products n increases, the regret of UCB1 would increase significantly. In contrast, the regret of LPUCB may remain relatively stable, as κ is primarily influenced by the structure of choice models.

In the numerical study, we first consider three scenarios with $K = 3$ and $|\mathcal{F}| = 50$ with a fully rank-based choice model. Scenarios 1, 2, and 3 consist of 10, 15, and 20 products, respectively. For all three scenarios, customer types are randomly generated with a length of $K = 3$. Product prices are randomly selected from the interval $[0,1]$ and $\boldsymbol{\lambda}^0$ are randomly generated, ensuring that $\sum_{g \in \mathcal{F}} \lambda_g^0 = 1$. The results are illustrated in Figure 3. Our observations indicate that as the number of products n increases, the regret of UCB1 rises significantly. In contrast, the regret of our LPUCB exhibits much less fluctuation when the number of assortments $|\mathcal{F}|$ remains fixed. To evaluate the order of regret, we analyze whether the term $\mathrm{Reg}^{\pi}(T)/\sqrt{T}$ can be effectively bounded by a constant. This will allow us to numerically demonstrate that



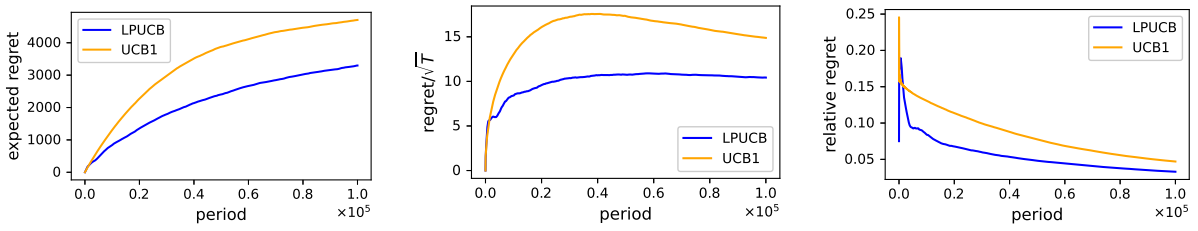
(a) Scenario 1 (b) Scenario 1 (c) Scenario 1

(d) Scenario 2 (e) Scenario 2 (f) Scenario 2

(g) Scenario 3 (h) Scenario 3 (i) Scenario 3

Figure 3 Performance of LPUCB and UCB1 in a Fully Rank-Based Choice Model

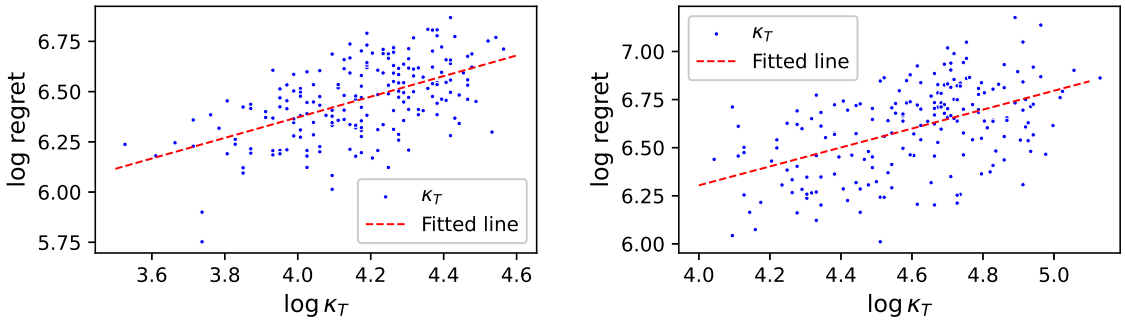
the regret is indeed of order $O(\sqrt{T})$. The results suggest that LPUCB can be easily bounded by constants across all three scenarios. In contrast, UCB1 exhibits an increasing trend when $n = 20$, which is attributed to a larger assortment set. Regarding relative regret, LPUCB converges significantly faster than UCB1. We have similar observations in a fully tree-based choice model. We formulate Scenario 4 consisting 10 products with $K = 3$ and $|\mathcal{F}| = 50$, where $\mathcal{F}$ is replaced with a fully tree-based choice model consisting of purchase decision trees randomly generated with maximum depth at 3 following Assumption 1 in Chen and Mišić (2021). Similar results are shown in Figure 4, and these results show the robust performance of LPUCB under different collections of customer types.



(a) Scenario 4 (b) Scenario 4 (c) Scenario 4

Figure 4 Performance of LPUCB and UCB1 under a Fully Tree-Based Choice Model

We next examine how κ, the total number of index sets, influences the performance of the algorithms. We perform 200 rounds of simulation with fixed parameters: $n = 10$, $K = 3$, and $T = 10000$. For both the fully rank-based and fully tree-based choice models, each round included randomly generated customer types, varying from 50 to 100. Denote by $\kappa_T = \sum_{I \in \mathcal{I}} \mathbb{I}\{n_T(I) > 0\}$ the number of index sets observed during the T horizon. As κ is the upper bound of κ_T, we can more precisely assess the order of regret associated with κ_T compared to that of κ. We therefore plot all 200 points on a plane, with the vertical axis representing $\log(\mathrm{Reg}^{\pi}(T))$ and the horizontal axis representing $\log(\kappa_T)$ in order to examine the relationship between these two factors. By taking the logarithm of both sides of Equation (4), we find that the coefficient of $\log(\kappa_T)$ is theoretically equal to $\frac{1}{2}$. Therefore, we anticipate that performing a simple linear regression on these 200 points will yield a coefficient close to $\frac{1}{2}$. The results can be seen in Figure 5, with the linear regression outcomes presented in Table 4. We observe a significant linear relationship between $\log(\mathrm{Reg}^{\pi}(T))$ and $\log(\kappa_T)$ ($***$ indicates a significant level at 0.01) under both choice models. Under a fully tree-based model, the coefficient for $\log(\kappa_T)$ is measured at 0.4916 which meets the expected value of 0.5, while under a fully rank-based model, the coefficient 0.5126 is slightly higher than the expected value of 0.5. This can be attributed to the fact that for each observed index set, its UCB can be validly formulated after it has been observed for the first time. Consequently, the regret would contain a linear term of κ_T representing the warm-up cost. It is important to note that κ is assumed to be smaller than T. Thus, we have $\kappa_T \leq \sqrt{\kappa T}$ which aligns with the regret established in Theorem 1. This linear term may push the coefficient of $\log(\kappa_T)$ to be slightly larger than 0.5, but it remains within an acceptable range.



(a) Rank-based (b) Tree-based

Figure 5 Relationship between $\log(\mathrm{Reg}^{\pi}(T))$ and $\log(\kappa_T)$ of LPUCB

6.2 Performance of bOGD-EG

We next move to assess the performance of bOGD-EG. As previously discussed, if the optimal assortment $\tilde{S}^*$ maximizing $f(\tilde{S}, \boldsymbol{\lambda}^0)$ lies interior the relaxed region $[0,1]^n$, bOGD-EG can ensure convergence of $\tilde{S}_t^{\pi}$ to $\tilde{S}^*$, but not to the true optimal one S^*, resulting in a linearly increasing regret without the boundary condition.

Table 4 Linear Regression Outcome of LPUCB between $\log(\mathrm{Reg}^{\pi}(T))$ and $\log(\kappa_T)$

(a) Rank-based

Variables	Coefficient	t value
Constant	4.3217	18.818 (***)
$\log(\kappa_T)$	0.5126	9.317 (***)

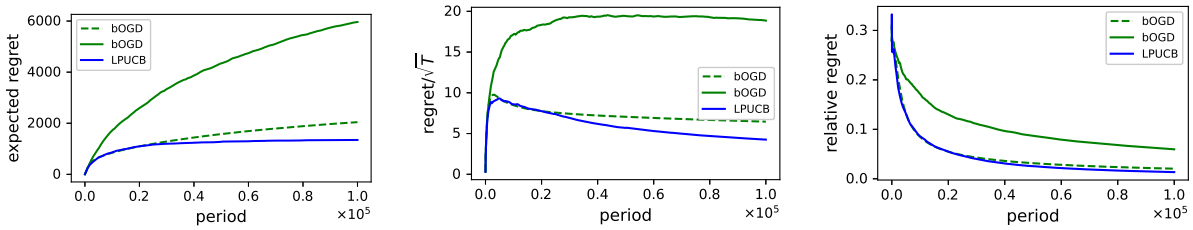
(b) Tree-based

Variables	Coefficient	t value
Constant	4.3378	15.579 (***)
$\log(\kappa_T)$	0.4916	8.138 (***)

Subsequently, we evaluate bOGD-EG using numerical examples that meet the boundary condition, while excluding the capacity constraint in our numerical study. Specifically, there exists a unique $\tilde{S}^*$ maximizing $f(\tilde{S}, \boldsymbol{\lambda}^0)$ on $\{0,1\}^n$, indicating that $\tilde{S}^* = S^*$.

Note that regret $\mathrm{Reg}^{\pi}(T)$ accounts for the performance difference between optimal assortment S^* and S_t^{π}. To quantify the loss of randomization process, we introduce the *pseudo regret* $\sum_{t=1}^{T} f(\tilde{S}^*, \boldsymbol{\lambda}^0) - \mathbb{E}^{\pi}[f(\tilde{S}_t^{\pi}, \boldsymbol{\lambda}^0)]$, representing the loss resulting from using the LP-relaxed SG problem (7) to compute gradients. Note that bOGD-EG simultaneously records S_t^{π} and $\tilde{S}_t^{\pi}$. This allows us to plot both the regret curve and the pseudo-regret curve on the same sample paths. The difference between regret and pseudo regret is $\sum_{t=1}^{T} \mathbb{E}^{\pi}[f(\tilde{S}_t^{\pi}, \boldsymbol{\lambda}^0) - f(S_t^{\pi}, \boldsymbol{\lambda}^0)]$, accounting for the total loss incurred by the randomization process $\mathcal{R}$ in all periods. As mentioned in Section 5.2, the difference is greater than zero due to the concavity of function $f(\cdot)$.

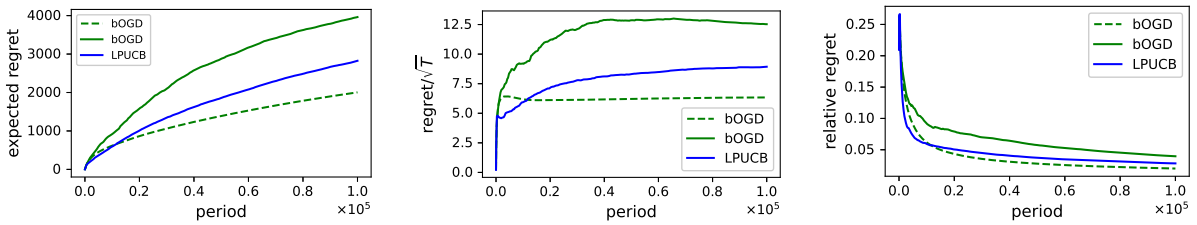
In Scenario 6, we set $n = 10$, $K = 10$ and $|\mathcal{F}| = 50$ with a fully rank-based choice model whose maximum length is 3. We compare the performance of bOGD-EG (both regret and the pseudo one) and LPUCB and illustrate the results in Figure 6. The solid green curve represents the regret of bOGD-EG and the dashed green curve represents its pseudo-regret. We first observe that both the regret and the pseudo regret of



(a) Scenario 6 (b) Scenario 6 (c) Scenario 6

Figure 6 Performance of bOGD-EG under a Fully Rank-Based Choice Model

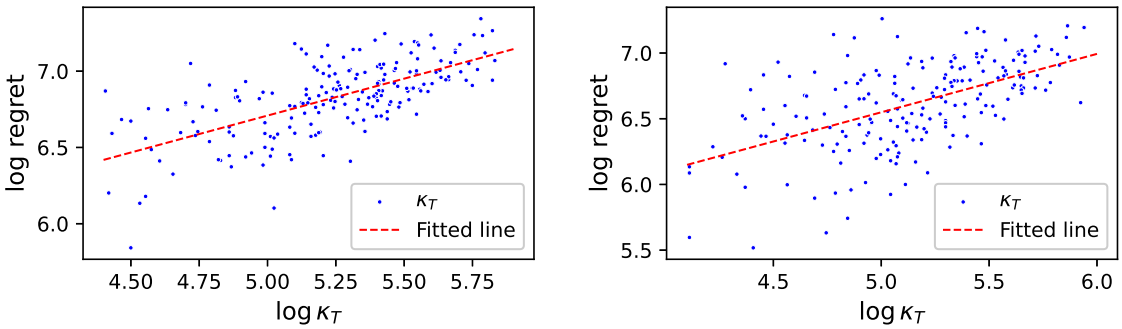
bOGD-EG exhibit a clear $\sqrt{T}$ rate of convergence. However, the regret of bOGD-EG is significantly larger than the pseudo-regret, highlighting the loss incurred due to randomization. In comparison to LPUCB, the regret of bOGD-EG is also considerably larger. This can be interpreted as the cost associated with LP relaxation and the resulting reduction in computational complexity. In Scenario 7, we use a fully tree-based choice model that has a maximum preference depth of 3. Figure 7 illustrates results that are very similar to those observed previously.



(a) Scenario 7 (b) Scenario 7 (c) Scenario 7

Figure 7 Performance of bOGD-EG under a Fully Tree-Based Choice Model

Next, we examine how κ influences the performance of bOGD-EG. Similarly, we perform 200 rounds of simulations with fixed parameters: $n = K = 8$ and $T = 10000$. Each round includes randomly generated customer types ranging from 30 to 100, allowing a reduction in the number of customer types to effectively control the range of κ_T for both the rank-based and tree-based choice models. We plot $\log(\mathrm{Reg}^{\pi}(T))$-$\log(\kappa_T)$ of all 200 points in Figure 8 and illustrate the linear regression result in Table 5. For both rank-based and



(a) Rank-based (b) Tree-based

Figure 8 Relationship between $\log(\mathrm{Reg}^{\pi}(T))$ and $\log(\kappa_T)$ of bOGD-EG

Table 5 Linear Regression Outcome of bOGD-EG between $\log(\mathrm{Reg}^{\pi}(T))$ and $\log(\kappa_T)$

(a) Rank-based

Variables	Coefficient	t value
Constant	4.2881	21.010 (***)
$\log(\kappa_T)$	0.4842	12.448 (***)

(b) Tree-based

Variables	Coefficient	t value
Constant	4.3272	17.555 (***)
$\log(\kappa_T)$	0.4444	9.298 (***)

tree-based choice models, we observe that the coefficient of $\log(\kappa_T)$ is slightly lower than 0.5 compared to LPUCB. This result can be mainly attributed to the additional term $n\sqrt{T}$ in Theorem 3. This is because, the regret upper bound in Theorem 3 implies a linear relationship between $\log(\mathrm{Reg}^{\pi}(T))$ and $\log(\sqrt{\kappa}+\sqrt{n})$ as n and T are fixed throughout this test. As shown in Figure 9, the curve $\log(\mathrm{Reg}^{\pi}(T)) = \log(\sqrt{\kappa}+\sqrt{n}) = \frac{1}{2}\log\kappa + \log(1+\frac{\sqrt{n}}{\sqrt{\kappa}})$ has an increasing derivative converging to $\frac{1}{2}$. This implies that the coefficient of a linear regression of randomly selected points on this curve would be close to $\frac{1}{2}$ but would not exceed this

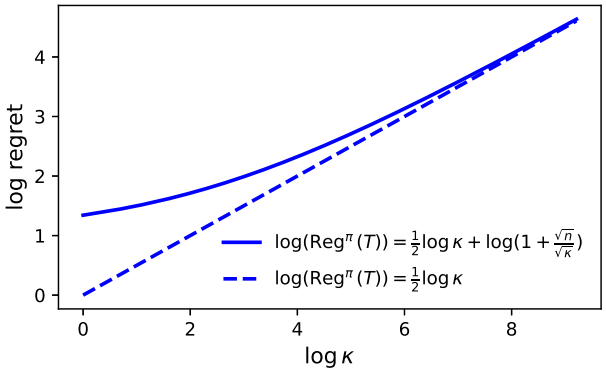


Figure 9 Theoretical Relationship between $\log(\mathrm{Reg}^{\pi}(T))$ and $\log \kappa$ of bOGD-EG

value. This observation provides a clear explanation for the results presented in Table 5. However, the latent influence of the warm-up cost still exists in bOGD-EG, which may potentially push the coefficient to be higher than $\frac{1}{2}$ especially when κ is large and approaches T.

6.3 Computational Efficiency of Both LPUCB and bOGD-EG

Finally, we compare the computational efficiency of both algorithms. We first design scenarios with varying values of n, T and randomly selected sets of $\mathcal{F}$ without the capacity constraint. As noted previously, the LPUCB algorithm exhibits polynomial complexity with respect to $|\mathcal{S}|$, κ and $|\mathcal{F}|$. In contrast, the bOGD-EG algorithm has a polynomial complexity in each period with respect to n, κ and $|\mathcal{F}|$. First, we expect a gradual increase in the running time for both algorithms as the sizes of $|\mathcal{F}|$ and T increase. However, we expect a more pronounced increase in running time with respect to n of LPUCB, while bOGD-EG shows only a gradual increase with n of bOGD-EG. The results are presented in Table 6. The efficiency of

Table 6 Running Time of Both Algorithms without Capacity Constraint (in Seconds)

Scenario \ Horizon		$T = 10000$	$T = 50000$	$T = 100000$
$n = 8$ $\lvert\mathcal{F}\rvert = 50$	LPUCB	7.01	16.49	32.58
	bOGD	1.26	4.46	7.92
$n = 10$ $\lvert\mathcal{F}\rvert = 50$	LPUCB	32.43	97.64	143.12
	bOGD	1.39	5.30	9.94
$n = 12$ $\lvert\mathcal{F}\rvert = 50$	LPUCB	115.98	319.59	523.52
	bOGD	1.59	6.31	11.97
$n = 10$ $\lvert\mathcal{F}\rvert = 100$	LPUCB	52.30	188.22	316.20
	bOGD	2.90	10.86	23.70
$n = 10$ $\lvert\mathcal{F}\rvert = 150$	LPUCB	68.67	256.97	437.41
	bOGD	4.35	19.71	36.15

bOGD-EG consistently outperforms that of LPUCB across all scenarios tested. In addition, we observe that n does not significantly impact the efficiency of bOGD-EG. In contrast, LPUCB exhibits an exponential

increase in running time as n grows, and this is mainly because $|\mathcal{S}| = 2^n$ without the capacity constraint. Regarding $|\mathcal{F}|$ and T, the running time of both algorithms increases only slightly. This indicates that bOGD-EG demonstrates overall better performance in terms of computational complexity compared to LPUCB, though at the expense of regret performance.

With the capacity constraint, we test the previously designed scenarios, but with a capacity constraint of $K = 3$. We expect that LPUCB will run significantly faster under these conditions, as the number of assortments is reduced. Conversely, due to the modified randomization process $\mathcal{R}^C$, the complexity of bOGD-EG is expected to increase dramatically with the growth of n. The results are presented in Table 7. The findings show that the performance of LPUCB has improved significantly due to the reduction in the number of applicable assortments. In contrast, the performance of bOGD-EG has deteriorated when using the $\mathcal{R}^C$ process across all conditions. In particular, we observe that the running time of bOGD-EG increases at an accelerating rate as n increases, indicating the computational intractability associated with $\mathcal{R}^C$. Nevertheless, bOGD-EG continues to demonstrate better performance than LPUCB when n, $|\mathcal{F}|$ and T are relatively large.

Table 7 Running Time of Both Algorithms with Capacity Constraint (in Seconds)

Scenario \ Horizon		$T = 10000$	$T = 50000$	$T = 100000$
$n = 8$ $\lvert\mathcal{F}\rvert = 50$	LPUCB	2.27	8.60	12.22
	bOGD	2.45	5.93	8.96
$n = 10$ $\lvert\mathcal{F}\rvert = 50$	LPUCB	5.04	16.68	23.84
	bOGD	3.73	9.10	13.85
$n = 12$ $\lvert\mathcal{F}\rvert = 50$	LPUCB	6.56	23.59	34.87
	bOGD	6.58	15.30	22.57
$n = 10$ $\lvert\mathcal{F}\rvert = 100$	LPUCB	8.66	36.16	66.06
	bOGD	4.70	12.09	18.49
$n = 10$ $\lvert\mathcal{F}\rvert = 150$	LPUCB	13.52	54.72	104.59
	bOGD	5.74	15.41	23.37

7 Concluding Remarks

This paper considers a dynamic learning and assortment selection problem with a non-parametric choice model. The arrival probability of each known customer type is unknown to the firm while the purchase information is observable. The firm offers an assortment of n substitutable products with capacity constraints at K. Each arriving customer then decides to make a purchase according to his/her specific purchase pattern. The objective of the firm is to optimize the total expected revenue over the T periods while learning the arrival probabilities by incomplete information. To estimate the arrival probabilities, we aggregate the arrival probabilities of customers with the same reaction to a given assortment referred to as the customer index set. Then, we design the LPUCB algorithm that constructs a linear confidential space for the purchase

probability vector, and selects the assortment with the highest potential revenue within the confidential space. This bandit-like algorithm achieves the expected regret of order at $O(\sqrt{\kappa KT \log \kappa T})$, where κ serves as a valuable indicator of the complexity of the problem and significantly influences the best achievable performance.

The LPUCB algorithm is applicable to a wide range of non-parametric choice models with good performance. However,computational complexity remains a critical issue to address. As a result, we design a subgradient-based method called bOGD-EG to reduce computational complexity by identifying the relaxed optimal assortment rather than searching for the exact optimal assortment. We reveal that the theoretical performance of bOGD-EG without the capacity constraint can be bounded with $O(n\sqrt{T} + \sqrt{\kappa nT \log \kappa T} + nT)$ where $O(nT)$ increases linearly due to the loss incurred from randomization. We further find that imposing a boundary condition that requires the optimal relaxed assortment to lie within $\{0,1\}^n$ significantly decreases the $O(\sqrt{nKT})$ term and guarantees sublinear regret.

Finally, we perform a numerical analysis to evaluate the performance of the LPUCB and BOGD-EG algorithms. Our findings indicate that LPUCB converges significantly faster than the traditional UCB1 bandit method in terms of relative regret. In comparison, the regret associated with bOGD-EG is also notably higher than that of LPUCB. This can be attributed to the costs associated with LP relaxation and the resulting reduction in computational complexity. However, when computing efficiency is considered, bOGD-EG shows overall better performance compared to LPUCB, albeit at the cost of increased regret.

Acknowledgment

This study was supported by the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government, and Laboratory for AI-Powered Financial Technologies.

References

Abernethy, Jacob, Chansoo Lee, Abhinav Sinha, Ambuj Tewari. 2014. Online linear optimization via smoothing. *Proceedings of The 27th Conference on Learning Theory*, 35 807–823.

Agrawal, Shipra, Vashist Avadhanula, Vineet Goyal, Assaf Zeevi. 2017. Thompson sampling for the MNL-bandit. *arXiv e-prints*, arXiv:1706.00977doi:10.48550/arXiv.1706.00977.

Agrawal, Shipra, Vashist Avadhanula, Vineet Goyal, Assaf Zeevi. 2019. Mnl-bandit: A dynamic learning approach to assortment selection. *Operations Research*, 67 (5), 1453-1485.

Aouad, Ali, Vivek Farias, Retsef Levi, Danny Segev. 2018. The approximability of assortment optimization under ranking preferences. *Operations Research*, 66 (6), 1661-1669.

Auer, P., Paul Fischer, N. Cesa-Bianchi. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47 (3), 235-256.

Bernstein, Fernando, Sajad Modaresi, Denis Sauré. 2019. A dynamic clustering approach to data-driven assortment personalization. *Management Science*, 65 (5), 2095-2115.

Bubeck, Sébastien, Nicolò Cesa-Bianchi. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, .

Cardoso, Adrian Rivera, He Wang, Huan Xu. 2024. The online saddle point problem and online convex optimization with knapsacks. *Mathematics of Operations Research*, 0 (0), Articles In Advance.

Cesa-Bianchi, Nicolo, Gabor Lugosi. 2012. Combinatorial bandits. *Journal of Computer and System Sciences*, 78 (5), 1404–1422.

Chen, Xi, Chao Shi, Yining Wang, Yuan Zhou. 2021a. Dynamic assortment planning under nested logit models. *Production and Operations Management*, 30 (1), 85-102.

Chen, Xi, Yining Wang, Yuan Zhou. 2021b. Optimal policy for dynamic assortment planning under multinomial logit models. *Mathematics of Operations Research*, 46 (4), 1639-1657.

Chen, Yi-Chun, Velibor V. Mišić. 2021. Assortment optimization under the decision forest model. *arXiv e-prints*, arXiv:2103.14067.

Chen, Yi-Chun, Velibor V. Mišić. 2022. Decision forest: A nonparametric approach to modeling irrational choice. *Management Science*, 68 (10), 7090-7111.

Cohen, Alon, Tamir Hazan. 2015. Following the perturbed leader for online structured learning. *Proceedings of the 32nd International Conference on Machine Learning*, 37 1034–1042.

Combes, Richard, M. Sadegh Talebi, Alexandre Proutiere, Marc Lelarge. 2015. Combinatorial bandits revisited. *Advances in Neural Information Processing Systems*, 28 2116–2124.

Ding, Xiaofeng, Lin Chen, Pan Zhou, Zichuan Xu, Shiping Wen, John C.S. Lui, Hai Jin. 2021. Dynamic online convex optimization with long-term constraints via virtual queue. *Information Sciences*, 577 (2021), 140-161.

Farias, Vivek F, Srikanth Jagabathula, Devavrat Shah. 2013. A nonparametric approach to modeling choice with limited data. *Management Science*, 59 (2), 305-322.

Feldman, Jacob, Alice Paul, Huseyin Topaloglu. 2019. Technical note—assortment optimization with small consideration sets. *Operations Research*, 67 (5), 1283-1299.

Gergely, Neu. 2015. Explore no more: Improved high-probability regret bounds for nonstochastic bandits. *Advances in Neural Information Processing Systems*, 28 3168–3176.

Ho-Nguyen, Nam, Fatma Kılınç-Karzan. 2021. Technical note—dynamic data-driven estimation of nonparametric choice models. *Operations Research*, 69 (4), 1228-1239.

Honhon, Dorothee, Sreelata Jonnalagedda, Pan Xiajun Amy. 2012. Optimal algorithms for assortment selection under ranking-based consumer choice models. *Manufacturing & Service Operations Management*, 14 (2), 279-289.

Jagabathula, Srikanth, Paat Rusmevichientong. 2017. A nonparametric joint assortment and price choice model. *Management Science*, 63 (9), 3128–3145.

Jagabathula, Srikanth, Gustavo Vulcano. 2018. A partial-order-based model to estimate individual preferences using panel data. *Management Science*, 64 (4), 1609-1628.

Kallus, Nathan, Madeleine Udell. 2020. Dynamic assortment personalization in high dimensions. *Operations Research*, 68 (4), 1020-1037.

Lattimore, Tor, Csaba Szepesvári. 2020. *Bandit algorithms*. Cambridge University Press.

Lesage-Landry, Antoine, Joshua A Taylor, Duncan S Callaway. 2021. Online convex optimization with binary constraints. *IEEE Transactions on Automatic Control*, 66 (12), 6164-6170.

Li, Shukai, Qi Luo, Zhiyuan Huang, Cong Shi. 2024. Online learning for constrained assortment optimization under markov chain choice model. *Operations Research*, .

Martin, Zinkevich. 2003. Online convex programming and generalized infinitesimal gradient ascent. *In Proceedings of the 20th International Conference on Machine Learning*, 928–936.

Meng, Qingxin, Jianwei Liu. 2024. Nonstationary online convex optimization with multiple predictions. *Information Sciences*, 654 (2024), 140-161.

Paul, Alice, Jacob Feldman, James Mario Davis. 2018. Assortment optimization and pricing under a nonparametric tree choice model. *Manufacturing & Service Operations Management*, 20 (3), 550–565.

Rusmevichientong, Paat, Zuo-Jun Max Shen, David B. Shmoys. 2010. Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations Research*, 58 (6), 1666-1680.

Sauré, Denis, Assaf Zeevi. 2013. Optimal dynamic assortment planning with demand learning. *Manufacturing & Service Operations Management*, 15 (3), 387-404.

Shen, Lingqing, Nam Ho-Nguyen, Fatma Kılınç-Karzan. 2023. An online convex optimization-based framework for convex bilevel optimization. *Proceedings of The 27th Conference on Learning Theory*, 198 1519–1582.

Simchi-Levi, David, Yunzong Xu. 2022. Bypassing the monster: A faster and simpler optimal algorithm for contextual bandits under realizability. *Mathematics of Operations Research*, 47 (3), 1904-1931.

Simonson, Itamar. 1989. Choice based on reasons: The case of attraction and compromise effects. *Journal of Consumer Research*, 16 (2), 158-174. URL `http://www.jstor.org/stable/2489315`.

Susan, Fransisca, Negin Golrezaei, Ehsan Emamjomeh-Zadeh, David Kempe. 2024. Active learning for non-parametric choice models. *arXiv e-prints*, arXiv:2208.03346v2.

Talluri, Kalyan T., Garrett J. Van Ryzin. 2004. *The theory and practice of revenue management*. Springer US.

van Ryzin, Garrett, Gustavo Vulcano. 2015. A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Science*, 61 (2), 281–300.

van Ryzin, Garrett, Gustavo Vulcano. 2017. An expectation-maximization method to estimate a rank-based choice model. *Operations Research*, 65 (2), 396–407.

Wang, Ruxian. 2022. The threshold effects on consumer choice and pricing decisions. *Manufacturing & Service Operations Management*, 24 (1), 448-466.

Yu, Hao, Michael J Neely. 2020. A low complexity algorithm with $O(\sqrt{T})$ regret and $O(1)$ constraint violations for online convex optimization with long term constraints. *Journal of Machine Learning Research*, 21 (1), 1–24.