Omnichannel Assortment Optimization under the Multinomial Logit Model with a Features Tree

Venus Lo, Huseyin Topaloglu
School of Operations Research and Information Engineering, Cornell University, Ithaca, New York 14853,
vhl8@cornell.edu, topaloglu@orie.cornell.edu

Problem Definition: We consider the assortment optimization problem of a retailer that operates a physical store and an online store. The products that can be offered are described by their features. Customers purchase amongst the products offered in their store of choice. However, customers of the online store can observe and try out products offered in the physical store, before buying online. These customers revise their mean utilities for online products based on features that these products share with the product they tried out. The full assortment is offered online, and the goal is to select an assortment for the physical store to maximize the retailer's total expected revenue. **Academic/Practical Relevance:** The physical store's assortment affects preferences for online products. Hence, unlike the traditional assortment optimization problem, the physical store's assortment affects revenue from both stores. **Methodology:** We introduce a features tree to organize products by features. The non-leaf vertices on the tree correspond to features, and the leaf vertices correspond to products. The ancestors of a leaf vertex correspond to features of the product. Customers choose amongst the products within their chosen assortment according to the multinomial logit model. We consider two settings: All customers purchase online, or we have a mix of customers purchasing from each store. **Results:** When all customers purchase online, we give an efficient algorithm to find the optimal assortment to display in the physical store. With a mix of customers, the problem becomes NP-hard and we give a fully polynomial-time approximation scheme. We numerically demonstrate that we can approximate the case where the products can have arbitrary combinations of features without a tree structure, and that our FPTAS performs remarkably well. **Managerial Implications:** In assortment planning, one must carefully consider the interactions between the assortments offered in online and physical stores.

Key words: assortment optimization; omnichannel retailing; features-based preferences

1. Introduction

Historically, retailers have operated on a single channel, either as pure offline brick-and-mortar stores or pure online stores. As online shopping has become ubiquitous, customers no longer use a single channel to both research and purchase products (Bachrach et al. (2016)). A customer can test out products at a local retailer before deciding to purchase online, a practice known as showrooming. In response, retailers have shifted away from operating a single channel to selling on multiple channels. Traditional brick-and-mortar stores like Best Buy and Wal-Mart operate websites and offer products that may not be available in-store. Diamonds retailer Blue Nile and eyeglasses retailer Warby Parker started as online retailers, but have opened showrooms to display their products. Recent literature refer to this phenomenon as an omnichannel retail environment because retailers must operate multiple channels as one cohesive unit. In this environment, products may share features so that if a retailer displays a product in-store, the customer can try out the product and modify her preferences of other online products based on similar features. This leads to the study of assortment optimization from an omnichannel viewpoint.

In our paper, we study an assortment optimization model for an omnichannel retailer operating a physical store and an online store. The retailer has n products at his disposal, and offers the full assortment of n products in his online store. He selects a subset of the full assortment for his physical store. Each product consists of several features, and a feature can be shared amongst several products. There are two types of customers: offline and online. An offline customer visits the the physical store to purchase from the assortment she sees in-store. An online customer visits the physical store to test out the products before purchasing from the full assortment online. By trying out the products in-store, she learns whether the products' features are over- or under-valued, and can update her preferences of online-only products based on the features that are shared with the displayed products. She does not have to observe an identical product in-store before updating her preferences online. For example, a customer purchasing a phone would update her opinion of a silver iPhone 8 if she sees a gold iPhone 8, because the phones share many features and experiencing the common features on one phone changes her opinion on the other phone.

The physical store serves as a display front for online customers to test out the products and update product preferences on a feature-by-feature basis, and as the only point of sales to offline customers. The assortment optimization problem is to select a subset of the products from the online assortment to display in the physical store to maximize the expected revenue.

1.1. Our Contributions

We consider a retailer operating two channels: an online store and a physical store. The retailer offers the full assortment online and a subset in his physical store. Each product is associated with a revenue, and the probability that a customer purchases product i depends on her preference weight of product i and her purchasing channel. Products have features and we describe similarities amongst products by their shared features. We model product features with a tree (see Figure 1). Each vertex of the tree is a feature and each leaf corresponds to a product, such that the path from a leaf to the root gives all the features that uniquely defines a product. Two products share a feature if that feature is a common ancestor on the tree.

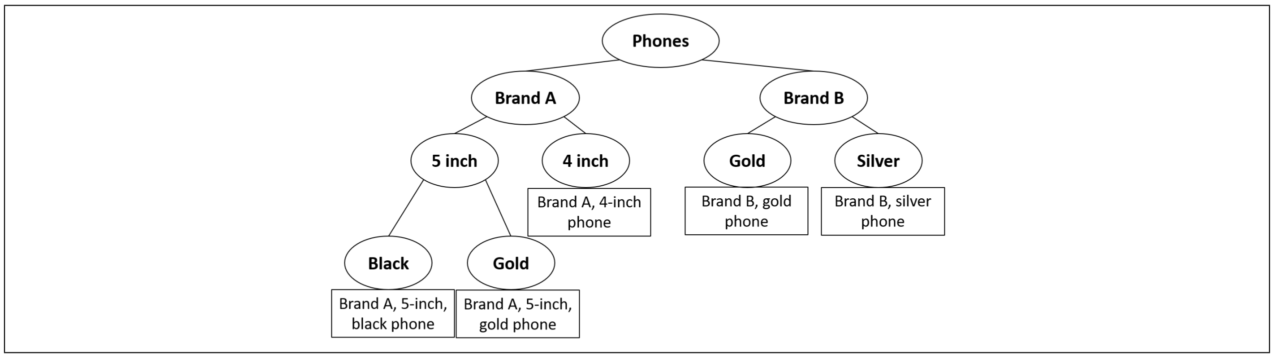

Figure 1: A features tree that describes five phones. Ovals describe features and boxes are the products that result from a set of features. There is one 4-inch phone under Brand A, so we do not differentiate by colours. There is no size feature for a Brand B phone.

Offline and online customers purchase according to the multinomial logit model (MNL). An offline customer decides amongst the products in the in-store assortment; her preference weights for products are given as input parameters and fixed. An online customer visits the physical store to test out features before purchasing from the full assortment online. She updates her preference weights in the online store using the features tree.

The goal is to choose an assortment to display in the physical store which maximizes the retailer's expected revenue across offline and online customers. A mathematical representation will be presented in the next section. We call this problem the OmniChannel Assortment optimization problem (OCA) under a features tree model.

We show that the OCA problem under a features tree model with offline and online customers is NP-hard via a reduction from the partition problem. We refer to the problem with both types of customer as the "general" setting. This is common amongst physical retailers that started their own online store (e.g. Best Buy). The retailer uses the physical store as a display front and to target traditional or impatient customers who consider only the products available at the store.

Since the general setting is NP-hard, we begin by studying the special case of the "showroom" setting, where all customers are online customers and the physical store is a display front. This is common amongst historically online retailers which later opened their own showrooms (e.g. Blue Nile). A customer sees a subset of products in-store and can test out different sizes and settings, but she can only order her ring online with all the required modifications. We present an algorithm that finds the optimal display assortment with runtime polynomial in the number of products.

We present a fully polynomial time approximation scheme (FPTAS) for the general setting, so that its runtime is polynomial in the number of products, the input parameters, and the desired accuracy. Our FPTAS involves creating a geometric grid over the numerator and denominator of the expected revenue function from offline customers. For each point in our grid, we construct a fractional combinatorial optimization problem which requires us to optimize the expected revenue of the online customers subject to constraints defined by the grid point. We parametrize our problem by the online expected revenue, and present a dynamic program to solve our parametrized problem. We compute an assortment for every point in the grid and choose the assortment with the largest total expected revenue amongst all points in our grid.

1.2. Literature Review

To the best of our knowledge, Dzyabura and Jagabathula (2017) were the first to study the assortment optimization problem of an omnichannel retailer. They considered feature classes with several feature values per class. A product consists of one feature value per feature class. In Figure 1, the feature classes are the brand, size, and colour, and the feature values in colour are gold, silver and black. The novelty is that they optimized over the sets of features to display and recovered the optimal assortment from the features set. In the showroom setting, their model is solvable in polynomial-time only when products have unit-revenue, and they presented a FPTAS for arbitrary revenues. In the general setting, it is difficult to recover an optimal assortment; various assortments can display the same set of features but earn different offline expected revenue. They restricted the space of feasible assortments to only the maximum assortments described by sets of features, and gave a FPTAS when the size of the feature classes is fixed. Finally, they demonstrated empirical evidence that customers update preferences by features via a field experiment; participants rank various messenger bags before and after observing a subset of similar bags.

Dzyabura and Jagabathula (2017) allowed products to share feature values over any feature classes, but we restrict features and products to a tree. In Figure 1, an online customer in their model would update her preference for colour on both Brand A and B gold phones when either are displayed. In our model, her preference for a Brand B gold phone with regards to colour is unaffected when she sees a Brand A gold phone. Colour is not a feature shared across brands, and seeing the shade of gold from the former brand does not give her information about the latter. Their model is able to describe product relationships more generally. On the other hand, they required that a product exists for every combination of feature values, so that the online store must carry phones in two sizes for both brands, whereas we recognize that some brands do not offer size options. Our model is more flexible with respect to the products that must be carried by the retailer in the online store. Hence neither model can be considered a generalization of the other and each has its strength. Finally, the features tree model is not NP-hard in the showroom setting, and allows us to optimize over all assortments in the general setting without restricting the size of feature classes. Trees have been used to describe product taxonomy in market segmentation and recommender system construction (Albadvi and Shahbazi (2009), Cho et al. (2002), Gangurde and Akarte (2015), Ziegler et al. (2004)).

Other works have studied challenges in the omnichannel environment. Balakrishnan et al. (2014) modeled the competition between a physical retailer and an online retailer, where customers may visit the physical retailer to test products before purchasing from the online retailer. Gao and Su (2016*b*) considered an omnichannel retailer who sells one product and shares information with customers to increase profit. To provide information about the online store, a display model is available in-store so that customers can try out the product and purchase online, even if it is sold out. To provide information on the physical store, in-store availability is posted online so that customers can evaluate the utility of visiting the physical store. They classified regimes where each option improves profit. In a similar paper, Gao and Su (2016*a*) considered an omnichannel retailer with a Buy-Online-Pickup-Instore (BOP) option, where customers can order before pick up to mitigate stock-out risk. This model incorporated the store's ability to cross-sell when customers visit, and they gave conditions under which BOP increased profitability. These papers considered a single product with independent demand, but we integrate a choice model so that demand depends on the assortment. They assumed that each customer has a private valuation which is revealed after seeing the product, whereas we assume a shift in market preference when features are seen.

Empirical evidence supports the importance of studying an omnichannel retail environment. Bell et al. (2017) investigated glasses retailer Warby Parker, which used to sell online exclusively and had a sampling program for customers to try glasses for five days. They found that online sales increased and returns decreased in cities where Warby Parker opened a showroom to let customers visit and try out frames before ordering online. Fornari et al. (2016) and Avery et al. (2012) looked at adding a physical store to the online store for an electronics retailer and a high-end apparel and furnishing retailer respectively. Both studies found that online sales increased in the long-run when customers have an additional channel to research products. In the reverse direction, customers like to purchase in-store, especially when products are "high-touch" (e.g. clothing, furniture) (Bell et al. (2014)). Retailers benefit from the opportunity to cross-sell products and Bell et al. (2014) found that BOP increases store visits and revenue.

Our work is also related to the large body of literature on assortment optimization. Our underlying choice model is MNL, which is credited to Luce (1959) and McFadden (1973), with the additional interpretation that a product's mean utility depends on whether its features are observed or not. A similar tree taxonomy was used by Li et al. (2015) in their d-level nested logit model to describe product features, where each vertex represents a feature that is shared amongst leaves in its subtree. The important difference is that their tree described a customer's choice process whereas our tree is used to update product preferences. In Li et al. (2015), a customer decides on the feature she likes and shrinks the assortment from which she is willing to purchase from as she moves from the root to a leaf. In our model, a customer always consider the entire assortment available either online or offline, depending on her type.

Finally, the existence of offline and online customers is related to the mixture of MNL model (MMNL) studied by Bront et al. (2009) and Rusmevichientong et al. (2014). In MMNL, multiple customer types consider the same assortment, but each type has different product preferences. In our model, all customers can have the same preferences, but consider different assortments based on their purchasing channel. Our FPTAS uses techniques from Désir et al. (2014)'s paper on capacitated MMNL, where each product has a capacity requirement and the assortment's capacity cannot exceed a budget. For each customer type, they created a geometric grid on the numerator and denominator of the expected revenue function. A point on the grid lower-bounds the expected revenue to be earned from each customer type. They gave a dynamic program that finds the minimum capacity assortment which satisfies the constraints imposed by the grid, if such an assortment exists. Our FPTAS creates a similar geometric grid on the numerator and denominator of the offline expected revenue, and we maximize the online expected revenue.

1.3. Organization

In Section 2, we describe the OCA problem under the features tree model. In particular, we describe how products are related by features on the features tree and how an online customer would update her product preferences based on the in-store assortment. In Section 3, we focus on the showroom setting with only online customers and we give a polynomial-time algorithm. In Section 4, we present the FPTAS for the general setting. In order to consider the practical performance of our FPTAS, we also provide an efficient method to upper-bound the optimal expected revenue in Section 5. In Section 6, we present extensions where the retailer can choose both the online and physical stores' assortments.

We provide numerical experiments in our last two sections. In Section 7, we test the modeling power of our features tree model when the underlying ground-truth model allows for more general features relationship across products. In Section 8, we assess the practical performance of our FPTAS against the upper-bound described in Section 5. Finally, we conclude in Section 9.

2. The Model

We consider a retailer operating an online store and a physical (offline) store. There are n products in the online store, denoted $N = \{1, \ldots, n\}$. Product i generates a revenue of π_i when it is purchased by a customer. All products are offered online and the retailer's decision is to select an assortment $S \subseteq N$ for the physical store, which we call the display assortment or the in-store assortment.

Products have features, and we use a features tree T to describe how features are shared amongst products. The vertices of the tree correspond to features of products, and the leaves correspond to products. The feature at a leaf is the final feature differentiating the corresponding product from the other children of its parent. The path from a leaf to the root gives all the features that uniquely defines the product. See Figure 1 for an example.

To describe the structure of the tree, we introduce notations to describe its parent-child relationships. We assume without loss of generality that our features tree T is a binary tree.

ASSUMPTION 1. *The features tree T is a binary tree so that every non-leaf vertex has exactly two children. A binary tree T with n leaves always has $2n-1$ vertices, hence we index the leaves by $N=\{1,\dots,n\}$ and non-leaves by $\{n+1,\dots,2n-1\}$.*

If k is not the root, then we denote $p(k)$ as the parent of k in T. If k is not a leaf, then it has two children; we denote the left child by $\ell(k)$ and the right child by $r(k)$. For all vertices k, let $A(k)$ be the set of ancestors of vertex k and itself. Let $L(k)$ be the set of leaves in the subtree rooted at k. Then $L(k)$ is the set of products which share feature k. Recursively, these are defined as:

$$A(k)=\begin{cases}\{\text{root}\} & : k=\text{root}\\ A(p(k))\cup\{k\} & : k\neq\text{root}\end{cases},$$

$$L(k)=\begin{cases}\{k\} & : k\in N\\ L(\ell(k))\cup L(r(k)) & : k\notin N\end{cases}.$$

A feature k is observed if any product in $L(k)$ is made available in-store. When we offer assortment S in-store, the set of features demonstrated to customers is $\cup_{j\in S}A(j)$, and we say that feature $k\in\cup_{j\in S}A(j)$ is demonstrated in-store.

There are two types of customers in the market: offline and online customers. A customer's type determines the assortment she is purchasing from and her preferences, which are described by her preference weights for the products. An offline customer purchases only from the in-store assortment, and always associates a preference weight of $\hat{v}_i$ with product i. An online customer visits the physical store to observe the displayed products, but ultimately purchases from the full assortment N online. She updates her preferences based on the features of products in S and associates preference weight of $v_i(S)$ with product i. We first describe the process in which an online customer updates her preference weights, then define $v_i(S)$.

An online customer has initial preference weight w_i for product i, which is her preference weight when she does not have the opportunity to examine any features of product i. She also associates a boost or discount multiplier δ_k to vertex k in T, which represents her change in preference when feature k is observed. The customer updates her preference weight by multiplying w_i by δ_k if she sees feature k and k is a feature of product i. The reason behind using multiplicative updates will be clear when we justify our model in light of MNL later. A multiplier $\delta_k>1$ corresponds to a feature being more appealing than presumed (undervalued), $\delta_k < 1$ corresponds to a feature being less attractive than presumed (overvalued), and $\delta_k = 1$ corresponds to no changes in opinion.

To define $v_i(S)$, we introduce binary variables x_k for every vertex k in the features tree, such that $x_k = 1$ means that feature k is demonstrated in-store and $x_k = 0$ otherwise. The latter case is equivalent to saying that none of the products having feature k is available in-store. Given an in-store assortment S, the corresponding characteristic vector $x \in \{0,1\}^{2n-1}$ is:

$$x_k = \begin{cases} 1 & : k \in (\cup_{j\in S} A(j)) \\ 0 & : k \notin (\cup_{j\in S} A(j)) \end{cases}.$$

If feature k is a leaf, then the retailer must offer product k when feature k is demonstrated in-store. Hence the first n indices of x reveals the in-store assortment. Given a vector x, we can recover an assortment by taking the set of products such that $x_k = 1$ for $k \in N$.

If feature k is not a leaf, then it is demonstrated in-store if and only if at least one of the leaves in its subtree is available in-store. Such a leaf is in the subtree of either $\ell(k)$ or $r(k)$, and this means that at least one of features $\ell(k)$ or $r(k)$ is demonstrated. Hence x_k =1 if and only if $x_{\ell(k)} = 1$ or $x_{r(k)} = 1$. We denote $\mathcal{X}$ as the set of feasible characteristic vectors, such that:

$$\mathcal{X} = \left\{ x \;\middle|\; \begin{array}{ll} x_{\ell(k)} \le x_k & \forall k \notin N, \\ x_{r(k)} \le x_k & \forall k \notin N, \\ x_k \le x_{\ell(k)} + x_{r(k)} & \forall k \notin N, \\ x_k \in \{0,1\} & \forall k. \end{array} \right\}.$$

Hence we can write the online preference weight of product i as $v_i(S) = v_i(x)$ for $x \in \mathcal{X}$, where:

$$v_i(x) = w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k}.$$

An online customer either purchases a product in N or chooses the no-purchase option. An offline customer purchases exclusively from the in-store assortment x or chooses the no-purchase option. The no-purchase option refers to the customer's ability to leave the store without making a purchase, and is available regardless of her shopping channel. This option does not share features with products, and its preference weight does not change regardless of the in-store assortment. We denote the preference weight of the no-purchase option by w_0 for an online customer and $\hat{v}_0$ for an offline customer.

A customer's purchase probability of product i is proportional to the preference weight of product i in the assortment which she purchases from, using the structure in MNL. The purchase probability of product i for online customers, given the in-store assortment x, is:

$$P_i^{ON}(x) = \frac{v_i(x)}{w_0 + \sum_{j=1}^n v_j(x)} = \frac{w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k}}{w_0 + \sum_{j=1}^n w_j \cdot \prod_{k\in A(j)} \delta_k^{x_k}}.$$

The purchase probability of product i for offline customers is:

$$P_i^{PHY}(x) = \frac{\hat{v}_i x_i}{\hat{v}_0 + \sum_{j=1}^n \hat{v}_j x_j}.$$

We denote the expected revenue from online and offline customers by $\Pi^{ON}(x)$ and $\Pi^{PHY}(x)$ respectively, where

$$\Pi^{ON}(x) = \sum_{i=1}^n \pi_i P_i^{ON}(x), \quad \text{and} \quad \Pi^{PHY}(x) = \sum_{i=1}^n \pi_i P_i^{PHY}(x).$$

Let q be the fraction of online customers and $(1-q)$ be the fraction of offline customers. When $q = 0$, then all customers are offline customers who purchase from the in-store assortment x with preference weights $\hat{v}_i$ for product i and this reverts back to the MNL model. Hence we consider $q \in (0,1]$ in the OCA problem. If we display assortment x, then we obtain an expected revenue of:

$$\Pi(x) = q \cdot \Pi^{ON}(x) + (1-q) \cdot \Pi^{PHY}(x).$$

The assortment optimization problem is to choose an assortment x that maximizes the total expected revenue. The OCA problem can be formulated as the following optimization problem, where $\Pi^{ON}(x), \Pi^{PHY}(x)$ has been expanded out:

$$\max_{x \in \mathcal{X}} \Pi(x) = \max_{x \in \mathcal{X}} q \cdot \frac{\sum_{i=1}^n \pi_i w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i=1}^n w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}} + (1-q) \cdot \frac{\sum_{i=1}^n \pi_i \hat{v}_i x_i}{\hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i}.$$

To interpret our model as a utility-maximizing choice model, let $\mu_i(x)$ denote the utility of product i when assortment x is displayed in-store. Following the setup of MNL, a customer's utility of product i is the mean utility of product i plus a noise ϵ_i, which is generated by an independent, standard Gumbel random variable with mean 0. Before observing any features, online customers have a mean utility of $\ln w_i$ for product i. If feature k of product i is demonstrated in-store, then the mean utility of product i changes additively by $\ln \delta_k$, so that

$$\mu_i(x) = \left(\ln w_i + \sum_{k \in A(i)} x_k \ln \delta_k \right) + \epsilon_i = \ln \left(w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k} \right) + \epsilon_i.$$

The additive updates to the mean utility of product i translate to multiplicative updates to the preference weight of product i.

3. Showroom Setting

We begin by considering $q = 1$, so that we are in the showroom setting with only online customers. Let γ^* be the optimal expected revenue:

$$\gamma^* = \max_{x \in \mathcal{X}} \quad \frac{\sum_{i=1}^n \pi_i w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i=1}^n w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}}.$$

As in standard fractional combinatorial optimization theory, we can parametrize the objective function and find a fixed point to the parametrized problem. Specifically, suppose there exists an assortment $x \in \mathcal{X}$ that generates expected revenue greater or equal to γ, so that

$$\frac{\sum_{i=1}^{n} \pi_i w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i=1}^{n} w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}} \geq \gamma.$$

By rearranging this inequality, we observe that a display assortment x generates expected revenue greater or equal to γ if and only if the same assortment satisfies $\sum_{i=1}^{n} (\pi_i - \gamma) w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k} \geq w_0 \gamma$. By maximizing the left side of this new inequality over all assortments in $\mathcal{X}$, we obtain our parametrized problem below. Let $f(\gamma)$ denote the value of Problem (1) as a function of γ:

$$f(\gamma) = \max_{x \in \mathcal{X}} \quad \sum_{i=1}^{n} (\pi_i - \gamma) w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}. \tag{1}$$

Claim 1. *Given $f(\gamma)$ as defined above, let γ^* be the optimal expected revenue of the showroom setting. Then the following are true: i) $f(\gamma) > w_0 \gamma$ if $\gamma < \gamma^*$, ii) $f(\gamma) < w_0 \gamma$ if $\gamma > \gamma^*$, and iii) $f(\gamma) = w_0 \gamma$ if $\gamma = \gamma^*$.*

Finding γ^* requires a method to efficiently solve Problem (1) and a method to search for γ^* over possible values of γ. A feasible solution to Problem (1) satisfies exactly one of the following statements: either $x = \vec{0}$ or there exists $i \in N$ such that $x_i = 1$, which is equivalent to $x_{\text{root}} = 1$ because the root is featured whenever a product is offered in-store. Our dynamic program considers non-empty assortments and compares the result to $x = \vec{0}$ to compute $f(\gamma)$.

Let $V_\gamma(k)$ be the maximum contribution to the objective function of Problem (1) from all leaves in the subtree rooted at vertex k, given that feature k is demonstrated in-store. This requires $x_k = 1$ and restricts the objective function to summing over products in $L(k)$. Since $x_k = 1$, we know $x_{k'} = 1$ for all ancestors k' in $A(k)$. We use $\mathcal{X}_k$ to denote the constraints of $\mathcal{X} \cap \{x \mid x_{k'} = 1 \; \forall k' \in A(k)\}$. Since we only consider the contribution from the leaves in $L(k)$, we can focus on feasibility of the constraints in $\mathcal{X}$ related to vertices in the subtree of k. By this definition, $V_\gamma(k)$ is:

$$V_\gamma(k) = \max_{x \in \mathcal{X}_k} \sum_{i \in L(k)} (\pi_i - \gamma) w_i \cdot \prod_{k' \in A(i)} \delta_{k'}^{x_{k'}}.$$

If k is a leaf, then feasibility to $\mathcal{X}_k$ requires $V_\gamma(k) = (\pi_k - \gamma) w_k \cdot \prod_{k' \in A(k)} \delta_{k'}$. If k is the root, then $V_\gamma(\text{root})$ optimizes Problem (1) over all feasible solutions in $\mathcal{X}_{\text{root}} = \mathcal{X} \backslash \{\vec{0}\}$. The objective value of the remaining solution $x = \vec{0}$ is $\sum_{i=1}^{n} (\pi_i - \gamma) w_i$; hence our parametrized problem can be written as:

$$f(\gamma) = \max \left\{ V_\gamma(\text{root}), \; \sum_{i=1}^{n} (\pi_i - \gamma) w_i \right\}.$$

To efficiently compute the value of $V_\gamma(\text{root})$, we construct a dynamic program that solves the tree from the leaves up to the root, where each $V_\gamma(k)$ is related to the values at its children: $V_\gamma(\ell(k))$ and $V_\gamma(r(k))$. Since $x_k = 1$, we consider three cases: *i)* $x_{\ell(k)} = x_{r(k)} = 1$, *ii)* $x_{\ell(k)} = 1, x_{r(k)} = 0$, or *iii)* $x_{\ell(k)} = 0, x_{r(k)} = 1$. In each of these cases, the objective function and constraints can be broken up by the subtrees of $\ell(k)$ and $r(k)$, and we can optimize each subtree separately.

In cases (*ii*) and (*iii*), we need to consider the contribution to the objective function from the leaves in the subtree of the undemonstrated child of k. Consider case (*ii*) where $x_{r(k)} = 0$. Feature $r(k)$ and its descendants are not demonstrated in-store, but feature k and all its ancestors are demonstrated. The preference weight of product i in $L(r(k))$ is:

$$w_i \cdot \prod_{k' \in A(i)} \delta_{k'}^{x_{k'}} = w_i \cdot \prod_{k' \in A(k)} \delta_{k'}.$$

The same analysis holds for $i \in L(\ell(k))$ if we consider case (*iii*). To simplify notation, let Δ_k denote the product of all $\delta_{k'}$ such that k' is an ancestor of k. This is defined recursively, with the base case $\Delta_{p(\text{root})} = 1$:

$$\Delta_k = \prod_{k' \in A(k)} \delta_{k'} = \Delta_{p(k)} \cdot \delta_k \qquad \forall k = 1, \dots 2n-1.$$

For computational purpose, the size of Δ_k is still polynomial in the input sizes, as $\log \Delta_k = \sum_{k' \in A(k)} \log \delta_{k'} \le n \cdot \max_{k'} \log \delta_{k'}$. Based on the three cases above, we can rewrite $V_\gamma(k)$ as:

$$V_\gamma(k) = (\pi_k - \gamma) w_k \Delta_k \qquad \forall k \in N,$$

$$V_\gamma(k) = \max \left\{ \begin{array}{l} V_\gamma(\ell(k)) + V_\gamma(r(k)), \\ V_\gamma(\ell(k)) + \Delta_k \cdot \sum_{i \in L(r(k))} (\pi_i - \gamma) w_i, \\ \Delta_k \cdot \sum_{i \in L(\ell(k))} (\pi_i - \gamma) w_i + V_\gamma(r(k)) \end{array} \right\} \qquad \forall k \notin N.$$

The base cases are $k \in N$ and we solve the dynamic program from the leaves up to the root. For any fixed γ, computing $V_\gamma(\text{root})$ requires us to solve a dynamic program with $O(n)$ states and 3 decisions per state. We compare the value of $V_\gamma(\text{root})$ to the objective value when $x = \vec{0}$, to obtain $f(\gamma) = \max\{V_\gamma(\text{root}), \sum_{i=1}^n (\pi_i - \gamma) w_i\}$. Hence, we can compute $f(\gamma)$ in $O(n)$ operations.

Finally, we consider the number of operations necessary to find γ^* such that $f(\gamma^*) = w_0 \gamma^*$. The parametrized problem $f(\gamma)$ is monotone decreasing in γ, and one way to find our solution γ^* is via bisection search between an upper and lower bound on the online expected revenue. In order to bound the runtime of bisection search, we need to bound the smallest gap in expected revenue between two assortments, which is not a simple task. Another method is to apply Newton's method (Radzik (1998)). If the numerator and denominator of the online expected revenue can be written as linear functions, then the fixed point can be found in $O(n^2 \log^2 n)$ iterations of Newton's method because $\mathcal{X} \subseteq \{0,1\}^{2n-1}$. The next lemma shows that the numerator and denominator of the online expected revenue can be written as linear functions of $x \in \mathcal{X}$.

Lemma 1. *The preference weight of product i for an online customer, if she observes assortment x, can be written as a linear function of $x \in \mathcal{X}$:*

$$v_i(x) = w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k} = w_i \cdot \left(1 + \sum_{k \in A(i)} (\Delta_k - \Delta_{p(k)}) \cdot x_k\right).$$

Hence, we can write the online expected profit as:

$$\Pi^{ON}(x) = \frac{\sum_{i=1}^{n} \pi_i w_i + \sum_{k=1}^{2n-1} \left(\sum_{i \in L(k)} \pi_i w_i\right) \cdot \left(\Delta_k - \Delta_{p(k)}\right) \cdot x_k}{w_0 + \sum_{i=1}^{n} w_i + \sum_{k=1}^{2n-1} \left(\sum_{i \in L(k)} w_i\right) \cdot \left(\Delta_k - \Delta_{p(k)}\right) \cdot x_k}.$$

The proof is deferred to Online Appendix A.

Theorem 1. *In the showroom setting of the OCA problem under the features tree model, where all customers use the physical store as a showroom to observe features and purchase ultimately from the online assortment, we can compute an optimal display assortment in $O(n^3 \log^2 n)$ operations.*

Proof. As stated previously, finding the optimal assortment in the showroom setting requires computing $f(\gamma)$ and finding γ^* such that $f(\gamma^*) = w_0 \gamma^*$. It takes $O(n)$ operations to compute $f(\gamma)$ for each possible value of γ. Using Newton's method, we can find γ^* in $O(n^2 \log^2 n)$ iterations of computing $f(\gamma)$. Hence the total number of operations is $O(n^3 \log^2 n)$.

□

4. General Setting

Individually, optimizing the online expected revenue (showroom setting) and the offline expected revenue can be done in polynomial time. Since the objective function of the general setting is a sum of fractions, one should suspect that the OCA problem under the features tree model is NP-hard when $q \in (0,1)$, as is the case with many assortment optimization problems where the objective is a sum of fractions. We prove this in the next proposition. The proof is in Online Appendix A.

Proposition 1. *The general setting of the OCA problem under the features tree model, where $q \in (0,1)$, is NP-hard.*

In Subsection 4.1, we describe our FPTAS to compute an assortment guaranteeing $(1-\epsilon)$ of the optimal expected revenue. In order to run our FPTAS, our algorithm requires us to solve a parametrized problem, which will be done via dynamic programming in Subsection 4.2.

4.1. FPTAS for General Setting

Instead of optimizing over the sum of two fractions, we will extend the mathematical program in the showroom setting to incorporate bounds on the numerator and denominator of the offline expected revenue. Suppose x^* is the optimal solution to our problem and let $R^* = \sum_{i=1}^n \pi_i \hat{v}_i x_i^*$, $U^* = \hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i^*$. Denote the optimal expected revenue as $\Pi^* = \Pi(x^*)$.

For a pair $(R,U) \geq (0, \hat{v}_0)$, consider the following problem which optimizes the online expected revenue over $x \in \mathcal{X}$, subject to constraints on the numerator and denominator of the offline expected revenue based on inputs (R,U):

$$g(R,U) = \max_{x \in \mathcal{X}} \left\{ \frac{\sum_{i=1}^n \pi_i w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i=1}^n w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}} \;\middle|\; \sum_{i=1}^n \pi_i \hat{v}_i x_i \geq R,\ \hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i \leq U \right\}. \tag{2}$$

Solving Problem (2) at (R^*, U^*) gives us the characteristic vector of the optimal assortment. This is because x^* is a feasible solution with objective value $\Pi^{ON}(x^*)$. Any optimal solution x' to Problem (2) at (R^*, U^*) would earn offline expected revenue of $\Pi^{PHY}(x') \geq R^*/U^*$ because x' satisfies the two constraints above, and earn online expected revenue of $\Pi^{ON}(x') \geq \Pi^{ON}(x^*)$ because x' is optimal.

The problem is that we do not know R^* or U^*, and it is not clear that we can solve Problem (2) efficiently. Our strategy is to apply a geometric grid to the possible values of R and U and approximately compute $g(R,U)$ within this grid. The optimal solution on this grid will give us a FPTAS to our problem.

To set up our grid, let $\underline{R} = \min_{i \in N} \pi_i \hat{v}_i$ and $\overline{R} = \max_{i \in N} \pi_i \hat{v}$. Similarly, let $\underline{U} = \min_{i \in N \cup \{0\}} \hat{v}_i$ and $\overline{U} = \max_{i \in N \cup \{0\}} \hat{v}_i$. Then $\underline{R}$ and $n\overline{R}$ are the lower- and upper-bounds on the numerator of $\Pi^{PHY}(\cdot)$, assuming that at least one product is offered in-store. Similarly, $\underline{U}$ and $(n+1)\overline{U}$ are the lower- and upper-bounds of the denominator of $\Pi^{PHY}(\cdot)$. Given $\epsilon > 0$, our grid $\mathcal{K}_\epsilon$ is constructed as:

$$\begin{aligned}
\mathcal{K}_\epsilon^R &= \{\underline{R} \cdot (1+\epsilon)^d \mid \underline{R} \cdot (1+\epsilon)^d \leq (1+\epsilon) \cdot n\overline{R},\ d \in \mathbb{Z}_+\}, \\
\mathcal{K}_\epsilon^U &= \{\underline{U} \cdot (1+\epsilon)^d \mid \underline{U} \cdot (1+\epsilon)^d \leq (1+\epsilon) \cdot (n+1)\overline{U},\ d \in \mathbb{Z}_+\}, \text{ and} \\
\mathcal{K}_\epsilon &= \{(0, \hat{v}_0)\} \cup (\mathcal{K}_\epsilon^R \times \mathcal{K}_\epsilon^U).
\end{aligned}$$

Our grid $\mathcal{K}_\epsilon$ has $O\left(\frac{\log n\overline{R}/\underline{R}}{\epsilon} \cdot \frac{\log (n+1)\overline{U}/\underline{U}}{\epsilon}\right)$ points. Let $\mathcal{K}_{\text{feas}} \subseteq \mathcal{K}_\epsilon$ denote the set of points $(R,U) \in \mathcal{K}_\epsilon$ such that Problem (2) is feasible and $g(R,U)$ is well-defined.

PROPOSITION 2. *Consider an instance of the OCA problem under the features tree model with optimal expected revenue of* Π^* *and* $\epsilon < 1/6$*. For any* $(R,U) \in \mathcal{K}_{\text{feas}}$*, suppose we can compute a solution* $x^{R,U}$ *achieving online expected revenue* $\Pi^{ON}(x^{R,U}) \geq g(R,U)$ *and satisfying:*

$$\begin{aligned}
&\sum_{i=1}^n \pi_i \hat{v}_i x_i^{R,U} \geq (1-2\epsilon)R \\
&\hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i^{R,U} \leq (1+2\epsilon)U \\
&x^{R,U} \in \mathcal{X}.
\end{aligned}$$

Let $\bar{x} = \underset{(R,U) \in \mathcal{K}_{\text{feas}}}{\arg\max}\ \Pi(x^{R,U})$*. Then* $\Pi(\bar{x}) \geq (1-6\epsilon)\Pi^*$.

Proof. If Problem (2) is feasible at (R,U), then $x^{R,U}$ generates an expected revenue of

$$\begin{aligned}\Pi(x^{R,U}) &\geq q\cdot\Pi^{ON}(x^{R,U}) + (1-q)\cdot\frac{(1-2\epsilon)R}{(1+2\epsilon)U}\\ &\geq \frac{1-2\epsilon}{1+2\epsilon}\cdot\left(q\cdot\Pi^{ON}(x^{R,U}) + (1-q)\cdot\frac{R}{U}\right).\end{aligned}$$

If the optimal assortment is $x^*=\vec{0}$, then our solution to Problem (2) at $(0,\hat{v}_0)\in\mathcal{K}_\epsilon$ is optimal because $x^{0,\hat{v}_0}=\vec{0}$ is the unique solution at this grid point. Otherwise, there exists $(R',U')\in\mathcal{K}_\epsilon$ such that $R'\leq R^*<(1+\epsilon)R'$ and $U'\leq U^*<(1+\epsilon)U'$. We can use these points to upper-bound our optimal expected revenue. Since Problem (2) at $(R',(1+\epsilon)U')$ is a relaxation of Problem (2) at (R^*,U^*), Problem (2) must be feasible at $(R',(1+\epsilon)U')$ and hence $(R',(1+\epsilon)U')\in\mathcal{K}_{\text{feas}}$.

$$\begin{aligned}\Pi^* &= q\cdot g(R^*,U^*) + (1-q)\cdot\frac{R^*}{U^*}\\ &\leq q\cdot g(R',(1+\epsilon)U') + (1-q)\cdot\frac{(1+\epsilon)R'}{U'}\\ &\leq q\cdot\Pi^{ON}(x^{R',(1+\epsilon)U'}) + (1-q)\cdot(1+\epsilon)^2\cdot\frac{R'}{(1+\epsilon)U'}\\ &\leq (1+\epsilon)^2\left(q\cdot\Pi^{ON}(x^{R',(1+\epsilon)U'}) + (1-q)\cdot\frac{R'}{(1+\epsilon)U'}\right)\\ &\leq \frac{(1+\epsilon)^2(1+2\epsilon)}{1-2\epsilon}\Pi(x^{R',(1+\epsilon)U'})\\ &\leq \frac{1}{1-6\epsilon}\Pi(\bar{x}).\end{aligned}$$

The first inequality holds because we relaxed the right side of Problem (2). The second inequality holds by assumption, and the fourth inequality holds by the inequalities at the beginning of the proof. The last inequality holds by simplifying the ratio of ϵ's and the choice of $\bar{x}$.

$\square$

Our main challenge is to efficiently compute $x^{R,U}$ satisfying the assumptions of Proposition 2 for each $(R,U)\in\mathcal{K}_{\text{feas}}$. If we do the standard parametrization of the objective function, then we are looking at Problem (1) from the showroom setting, subject to the two additional knapsack-like constraints of Problem (2). This structure suggests rounding inputs $\pi_i\hat{v}_i$ and $\hat{v}_i$ based on R and U so that we can extend our dynamic program from the showroom setting to include the two knapsack-like constraints. For $i\in N$, we round the parameters as follows:

$$\tilde{\pi}_i^R = \left\lfloor\frac{\pi_i\hat{v}_i}{\epsilon R/n}\right\rfloor, \quad \text{and} \quad \tilde{v}_i^U = \left\lceil\frac{\hat{v}_i}{\epsilon U/(n+1)}\right\rceil.$$

We also round the offline preference weight of the no-purchase option and define Y_1, Y_2 as the polynomial-sized bounds on the rounded numerators and denominators of the offline expected revenue:

$$\tilde{v}_0^U = \left\lceil\frac{\hat{v}_0}{\epsilon U/(n+1)}\right\rceil, \quad Y_1 = \left\lfloor\frac{n}{\epsilon}\right\rfloor - n, \text{ and } \quad Y_2 = \left\lceil\frac{n+1}{\epsilon}\right\rceil + (n+1).$$

For each pair (R,U), we parametrize Problem (2) and use the rounded constraints to obtain Problem (3) with value $\tilde{f}(\gamma,R,U)$.

$$\tilde{f}(\gamma,R,U)=\max_{x\in\mathcal{X}}\left\{\sum_{i=1}^{n}(\pi_i-\gamma)w_i\cdot\prod_{k\in A(i)}\delta_k^{x_k}\;\middle|\;\sum_{i=1}^{n}\tilde{\pi}_i^R x_i\geq Y_1,\ \tilde{v}_0^U+\sum_{i=1}^{n}\tilde{v}_i^U x_i\leq Y_2\right\}. \tag{3}$$

Lemma 2. *Suppose x is a feasible solution to Problem (2) at (R,U). Then x is a feasible solution to Problem (3) at (γ,R,U) for any γ. Furthermore, any feasible x to Problem (3) satisfies $\sum_{i=1}^{n}\pi_i\hat{v}_i x_i\geq(1-2\epsilon)R$ and $\hat{v}_0+\sum_{i=1}^{n}\hat{v}_i x_i\leq(1+2\epsilon)U$.*

If x is feasible to Problem (2) at (R,U), then $x\in\mathcal{X}$, $\sum_{i=1}^{n}\pi_i\hat{v}_i x_i\geq R$, and $\hat{v}_0+\sum_{i=1}^{n}\hat{v}_i x_i\leq U$. We check for feasibility of the two knapsack-like constraints in Problem (3). Using the modified values for the numerator, we obtain the following chain of inequalities:

$$\sum_{i=1}^{n}\tilde{\pi}_i^R x_i=\sum_{i=1}^{n}\left\lfloor\frac{\pi_i\hat{v}_i}{\epsilon R/n}\right\rfloor x_i\geq\sum_{i=1}^{n}\left(\frac{\pi_i\hat{v}_i}{\epsilon R/n}-1\right)x_i\geq\frac{R}{\epsilon R/n}-\sum_{i=1}^{n}x_i\geq\left\lfloor\frac{n}{\epsilon}\right\rfloor-n=Y_1.$$

Similarly, we obtain the following chain of inequalities for the denominator:

$$\begin{aligned}\tilde{v}_0^U+\sum_{i=1}^{n}\tilde{v}_i^U x_i&=\left\lceil\frac{\hat{v}_0}{\epsilon U/(n+1)}\right\rceil+\sum_{i=1}^{n}\left\lceil\frac{\hat{v}_i}{\epsilon U/(n+1)}\right\rceil x_i\\&\leq\frac{\hat{v}_0}{\epsilon U/(n+1)}+1+\sum_{i=1}^{n}\left(\frac{\hat{v}_i}{\epsilon U/(n+1)}+1\right)x_i\\&\leq\frac{U}{\epsilon U/(n+1)}+1+\sum_{i=1}^{n}x_i\leq\left\lceil\frac{n+1}{\epsilon}\right\rceil+1+n=Y_2.\end{aligned}$$

Hence x is feasible to Problem (3) at (γ,R,U).

Next, if x is feasible to Problem (3) at (γ,R,U), then we have the following chain of inequalities:

$$\sum_{i=1}^{n}\pi_i v_i x_i\geq\frac{\epsilon R}{n}\cdot\sum_{i=1}^{n}\tilde{\pi}_i^R x_i\geq\frac{\epsilon R}{n}\cdot\left(\left\lfloor\frac{n}{\epsilon}\right\rfloor-n\right)\geq R-\frac{\epsilon R}{n}-\epsilon R\geq(1-2\epsilon)R.$$

Similarly, we have the following chain of inequalities when considering the denominator:

$$\hat{v}_0+\sum_{i=1}^{n}\hat{v}_i x_i\leq\frac{\epsilon U}{n+1}\left(\tilde{v}_0^U+\sum_{i=1}^{n}\tilde{v}_i^U x_i\right)\leq\frac{\epsilon U}{n+1}\cdot\left(\left\lceil\frac{n+1}{\epsilon}\right\rceil+1+n\right)\leq U+\frac{\epsilon U}{n+1}+\epsilon U\leq(1+2\epsilon)U.$$

Corollary 1. *Suppose $(R,U)\in\mathcal{K}_{\text{feas}}$. Then there exists $\gamma^{R,U}$ such that $\tilde{f}(\gamma^{R,U},R,U)=w_0\gamma^{R,U}$. Furthermore, the optimal solution $x^{R,U}$ of Problem (3) with inputs $(\gamma^{R,U},R,U)$ satisfies $\sum_{i=1}^{n}\pi_i\hat{v}_i x_i^{R,U}\geq(1-2\epsilon)R$ and $1+\sum_{i=1}^{n}\hat{v}_i x_i^{R,U}\leq(1+2\epsilon)U$, and ensures that $\Pi^{ON}(x^{R,U})\geq g(R,U)$.*

Proof. By Lemma 2, Problem (3) has a feasible solution because Problem (2) is feasible at (R,U). The value of $\tilde{f}(\gamma,R,U)$ is decreasing in γ and $\tilde{f}(0,R,U)>0$, so there exists $\gamma^{R,U}$ such that $\tilde{f}(\gamma^{R,U},R,U)=w_0\gamma^{R,U}$, with corresponding optimal solution $x^{R,U}$ to Problem (3) at $(\gamma^{R,U},R,U)$.

Furthermore, we can rearrange the objective function to observe that $\Pi^{ON}(x^{R,U}) = \gamma^{R,U}$. In particular, by definition of $\gamma^{R,U}$, we have:

$$\sum_{i=1}^{n}(\pi_i - \gamma^{R,U})w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k^{R,U}} = w_0\gamma^{R,U}.$$

By rearranging, we obtain:

$$\gamma^{R,U} = \frac{\sum_{i=1}^{n} \pi_i w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k^{R,U}}}{w_0 + \sum_{i=1}^{n} w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k^{R,U}}} = \Pi^{ON}(x^{R,U}).$$

Next, suppose x^g optimizes Problem (2) with inputs (R,U). Applying Lemma 2 again tells us that x^g is feasible to Problem (3) with inputs $(\gamma^{R,U}, R, U)$. Hence, its objective value in Problem (3) is upper-bounded by $w_0\gamma^{R,U}$

$$\sum_{i=1}^{n}(\pi_i - \gamma^{R,U})w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k^g} \le w_0\gamma^{R,U}.$$

By rearranging, we observe:

$$\gamma^{R,U} \ge \frac{\sum_{i=1}^{n} \pi_i w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k^g}}{w_0 + \sum_{i=1}^{n} w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k^g}} = g(R,U).$$

Therefore, $g(R,U) \le \Pi^{ON}(x^{R,U})$.

□

In summary, we set up our FPTAS for the general setting by creating a geometric grid on the numerator and denominator of the offline expected revenue. For each point (R,U) on the grid, a feasible solution to Problem (2) has offline expected revenue greater or equal to R/U. However, Problem (2) is hard to solve because of the knapsack-like constraints. Instead, our FPTAS only requires a slightly perturbed solution to Problem (2) which satisfies the conditions of Proposition 2. Then Lemma 2 and Corollary 1 tells us that we can parametrize a rounded version of Problem (2) to arrive at Problem (3); the optimal solution $x^{R,U}$ to the fixed point $\gamma^{R,U}$ of Problem (3) satisfies the conditions of Proposition 2. Since the parameters in the constraints of Problem (3) are integers of size $O(n/\epsilon)$, we can find its optimal solution via dynamic programming, which is described in the next subsection.

4.2. Solving the Parametrized Problem at a Grid-point

In order to solve Problem (3) for inputs (γ, R, U), we set up our dynamic program following the structure of Section 3. Before proceeding with the value function, there are two cases of $(R,U) \in \mathcal{K}_\epsilon$ that we have to consider. If $(R,U) = (0, \hat{v}_0)$, then the only feasible solution to Problem (3) is $x = \vec{0}$

because $\tilde{v}_i^U \geq 1$ for all $i \in N$. Hence, the characteristic vector of any non-empty assortment violates the second constraint of Problem (3) and the value of $g(0, \hat{v}_0)$ can be computed directly.

On the other hand, if $(R, U) \neq (0, \hat{v}_0)$, then $R > 0$ and $x = \vec{0}$ is infeasible to the first constraint of Problem (3). The feasible region of Problem (3) is contained in $\mathcal{X} \backslash \{(0, \hat{v}_0)\}$, which is equal to $\mathcal{X}_{\text{root}}$. We proceed to solve Problem (3) for $(R, U) \in \mathcal{K}_\epsilon \backslash \{(0, \hat{v}_0\}$ via dynamic programming.

Like the showroom setting, we set up a value function which considers the maximum contribution from the leaves in the subtree rooted at vertex k, given that feature k is displayed. In addition, we need to include two more states in our value function to recognize the impact of $L(k)$ to the two knapsack-like constraints. Let $\tilde{V}_\gamma^{R,U}(k, y_1, y_2)$ be the maximum contribution from leaves in $L(k)$ when feature k is demonstrated in-store, subject to the constraints of $\mathcal{X}_k$ and the leaves in $L(k)$ contributing at least y_1 to the first constraint of Problem (3) and at most y_2 to the second constraint. Mathematically, this is represented by:

$$\tilde{V}_\gamma^{R,U}(k, y_1, y_2) = \max_{x \in \mathcal{X}_k} \left\{ \sum_{i \in L(k)} (\pi_i - \gamma) w_i \cdot \prod_{k' \in A(i)} \delta_{k'}^{x_{k'}} \;\middle|\; \sum_{i \in L(k)} \tilde{\pi}_i^R x_i \geq y_1, \sum_{i \in L(k)} \tilde{v}_i^U x_i \leq y_2 \right\}.$$

By definition of our value function, the value of our parametrized problem at $(R, U) \in \mathcal{K}_\epsilon \backslash \{(0, \hat{v}_0\}$ is:

$$\tilde{f}(\gamma, R, U) = \tilde{V}_\gamma^{R,U}(\text{root}, Y_1, Y_2 - \tilde{v}_0^U).$$

There are two details that may not be obvious at first glance. First, the third input to the value function at the root is $Y_2 - \tilde{v}_0^U$ and not Y_2. Since the no-purchase option is not a decision, we exclude the preference weight of the no-purchase option from the second constraint of $\tilde{V}_\gamma^{R,U}(k, y_1, y_2)$ at a subtree. Instead, $\tilde{v}_0^U$ is a constant that can be moved to the right side of the second constraint in Problem (3). Second, when we compute $f(\gamma)$ in Problem (1) of Section 3, we compare the result of the dynamic program at $V_\gamma(\text{root})$ to the objective function at $x = \vec{0}$ because the feasible region $\mathcal{X}_{\text{root}}$ excludes the empty assortment. As stated previously, we do not have to consider $x = \vec{0}$ in Problem (3) because the feasible region is reduced to $\mathcal{X}_{\text{root}}$ when $(R, U) \neq (0, \hat{v}_0)$.

We can rewrite $\tilde{V}_\gamma^{R,U}(k, y_1, y_2)$ based on the contributions from the children of k. Since $x_k = 1$, there are three cases that we consider: *i)* $x_{\ell(k)} = x_{r(k)} = 1$, *ii)* $x_{\ell(k)} = 1, x_{r(k)} = 0$, and *iii)* $x_{\ell(k)} = 0, x_{r(k)} = 1$. In the first case, we look at all possible splits of the required y_1, y_2 across the trees rooted at $\ell(k)$ and $r(k)$. In the second case, since $x_{r(k)} = 0$, the leaves in $L(r(k))$ cannot contribute to the two constraints and the required y_1, y_2 have to be satisfied on the left subtree. A similar argument applies to the third case. Hence, $\tilde{V}_\gamma^{R,U}(k, y_1, y_2)$ can be rewritten as the following dynamic program where $y_1 \in \{0, \ldots, Y_1\}$ and $y_2 \in \{0, \ldots, Y_2 - \tilde{v}_0^U\}$:

$$\tilde{V}_\gamma^{R,U}(k, y_1, y_2) = \begin{cases} (\pi_k - \gamma) w_k \Delta_k & \text{if } y_1 \leq \tilde{\pi}_k^R, y_2 \geq \tilde{v}_k^U \\ -\infty & \text{otherwise} \end{cases} \qquad \forall k \in N,$$

$$\tilde{V}_\gamma^{R,U}(k, y_1, y_2) = \max \left\{ \begin{array}{l} \max\limits_{\substack{0 \le x_1 \le y_1 \\ 0 \le x_2 \le y_2}} \tilde{V}_\gamma^{R,U}(\ell(k), x_1, x_2) + \tilde{V}_\gamma^{R,U}(r(k), y_1 - x_1, y_2 - x_2), \\ \tilde{V}_\gamma^{R,U}(\ell(k), y_1, y_2) + \Delta_k \cdot \sum\limits_{i \in L(r(k))} (\pi_i - \gamma) w_i, \\ \Delta_k \cdot \sum\limits_{i \in L(\ell(k))} (\pi_i - \gamma) w_i + \tilde{V}_\gamma^{R,U}(r(k), y_1, y_2) \end{array} \right\} \qquad \forall k \notin N.$$

To summarize this section, our FPTAS for finding assortment $\bar{x}$ such that $\Pi(\bar{x}) \ge (1-\epsilon)\Pi^*$ is presented in Algorithm 1. The set $\tilde{\mathcal{K}}_{\text{feas}}$ on the second line contains all (R, U) such that Problem (3) is feasible at $(0, R, U)$. By Lemma 2, feasibility of Problem (2) implies feasibility of Problem (3), so we have $\mathcal{K}_{\text{feas}} \subseteq \tilde{\mathcal{K}}_{\text{feas}}$. The while loop implements Newton's method to find γ such that $\tilde{f}(\gamma, R, U) = w_0\gamma$. Our returned solution $\bar{x}$ satisfies $\Pi(\bar{x}) \ge \max_{(R,U)\in\mathcal{K}_{\text{feas}}} \Pi(x^{R,U}) \ge (1-\epsilon)\Pi^*$ where the first inequality is due to $\mathcal{K}_{\text{feas}} \subseteq \tilde{\mathcal{K}}_{\text{feas}}$ and the second inequality is due to Proposition 2.

Algorithm 1: FPTAS to find an assortment $\bar{x}$ such that $\Pi(\bar{x}) \ge (1-\epsilon)\Pi^*$

```
Input  : Instance of OCA under the features tree model with desired accuracy $\epsilon$
Output: Characteristic vector $\bar{x}$ such that $\Pi(\bar{x}) \ge (1-\epsilon)\Pi^*$
Set $\epsilon \leftarrow \epsilon/6$ and construct $\mathcal{K}_\epsilon$ ;
Initialize grid $\tilde{\mathcal{K}}_{\text{feas}} \leftarrow \{(0, \hat{v}_0)\}$ ;
Set $x^{0,\hat{v}_0} \leftarrow \vec{0}$ ;
for $(R,U) \in \mathcal{K}_\epsilon \setminus \{(0, \hat{v}_0)\}$ do
    Set $\gamma \leftarrow 0$ ;
    Solve Problem (3) for optimal $x$, if feasible ;
    if Problem (3) is feasible then
        Update $\tilde{\mathcal{K}}_{\text{feas}} \leftarrow \tilde{\mathcal{K}}_{\text{feas}} \cup \{(R,U)\}$ ;
        while $\tilde{f}(\gamma, R, U) > w_0\gamma$ do
            Update $\gamma \leftarrow \Pi^{ON}(x)$ ;
            Resolve Problem (3) for optimal $x$ and value $\tilde{f}(\gamma, R, U)$ ;
        end
        Set $x^{R,U} \leftarrow x$;
end
Return $\bar{x} = \arg\max_{(R,U)\in\tilde{\mathcal{K}}_{\text{feas}}} \Pi(x^{R,U})$.
```

THEOREM 2. *Suppose Π^* is the optimal expected revenue of the OCA problem under the features tree model. For any $\epsilon \in (0,1)$, there exists an algorithm that finds an in-store assortment x such that $\Pi(x) \ge (1-\epsilon)\Pi^*$ in $O\left(\frac{n^7 \log^2 n \log n\overline{R}/\underline{R} \cdot \log(n+1)\overline{U}/\underline{U}}{\epsilon^6}\right)$ operations.*

Proof. Corollary 1 gives an $x^{R,U}$ that satisfies Proposition 2 whenever Problem (2) is feasible at (R,U). By Proposition 2, the solution $\bar{x} = \arg\max_{(R,U)\in\mathcal{K}_{\text{feas}}} \Pi(x^{R,U})$ satisfies $\Pi(\bar{x}) \geq (1-6\epsilon')\Pi^*$ for $\epsilon' < 1/6$. So given any $\epsilon \in (0,1)$, set $\epsilon' = \epsilon/6$ to get a $(1-\epsilon)$-approximation.

There are $O\left(\frac{\log n\overline{R}/\underline{R}}{\epsilon} \cdot \frac{\log(n+1)\overline{U}/\underline{U}}{\epsilon}\right)$ pairs of $(R,U) \in \mathcal{K}_\epsilon$ for which we have to compute $g(R,U)$ or determine that it is infeasible. If Problem (3) is feasible at $(0,R,U)$, then we run Newton's method to find $\gamma^{R,U}$ such that $\tilde{f}(\gamma^{R,U},R,U) = w_0\gamma^{R,U}$, which takes $O(n^2\log^2 n)$ iterations. In total, the number of times that we need to compute $\tilde{V}^{R,U}_\gamma(\text{root},Y_1,Y_2-\tilde{v}^U_0)$ is $O\left(\frac{n^2\log^2 n \log n\overline{R}/\underline{R}\cdot\log(n+1)\overline{U}/\underline{U}}{\epsilon^2}\right)$.

Finally, we need to compute $\tilde{V}^{R,U}_\gamma(\text{root},Y_1,Y_2-\tilde{v}^U_0)$ in order to solve Problem (3) at (γ,R,U). We compute $\tilde{V}^{R,U}_\gamma(k,y_1,y_2)$ for $O(n)$ vertices in T and $O(n^2/\epsilon^2)$ pairs of (y_1,y_2) per vertex. For each state, we consider at most $O(n^2/\epsilon^2)$ ways to split (y_1,y_2) over the left and right children of k. Hence there are $O(n^5/\epsilon^4)$ operations to obtain $\tilde{V}^{R,U}_\gamma(\text{root},Y_1,Y_2-\tilde{v}^U_0)$. Therefore, we have a total runtime of $O\left(\frac{n^7\log^2 n\log n\overline{R}/\underline{R}\cdot\log(n+1)\overline{U}/\underline{U}}{\epsilon^6}\right)$.

□

In Online Appendix B, we reduce the runtime by a factor of $O(n^2/\epsilon^2)$ by solving Problem (3) via a fancier constrained longest path construction in a directed acyclic graph. In Theorem 3 in the online appendix, we compute the FPTAS assortment in $O\left(\frac{n^5\log^2 n\log n\overline{R}/\underline{R}\cdot\log(n+1)\overline{U}/\underline{U}}{\epsilon^4}\right)$ operations.

5. Upper Bound on Optimal Expected Revenue

In order to evaluate the performance of our FPTAS in practice, we need a reasonable upper-bound on the optimal expected revenue without enumerating all possible assortments. We use the grid $\mathcal{K}_\epsilon$ from Section 4 to consider a simpler problem, but we can construct it with a smaller $\epsilon > 0$ in order to tighten our upper-bound.

Let $g^{\text{LP}}(R,U)$ be the value of Problem (2) with the integrality constraint relaxed. Let $\mathcal{X}^{\text{LP}}$ denote the linear realization of $\mathcal{X}$ where we replace $x \in \{0,1\}^{2n-1}$ with $0 \leq x \leq 1$. Then $g^{\text{LP}}(R,U)$ is:

$$g^{\text{LP}}(R,U) = \max_{x\in\mathcal{X}^{\text{LP}}} \left\{ \frac{\sum_{i=1}^n \pi_i w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i=1}^n w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k}} \;\middle|\; \sum_{i=1}^n \pi_i\hat{v}_i x_i \geq R,\; \hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i \leq U \right\}. \tag{4}$$

We explain how $g^{\text{LP}}(R,U)$ can be computed for any $R \geq 0$, $U \geq \hat{v}_0$ after the following theorem, which describes the upper-bound on the optimal expected revenue Π^*. Recall that if we have the optimal characteristic vector x^*, then $R^* = \sum_{i=1}^n \pi_i\hat{v}_i x_i^*$ and $U^* = \hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i^*$.

Proposition 3. *Define an upper-bound at grid point* $(R,U) \in \mathcal{K}_{\text{feas}}$ *as* $\Pi^{\text{LP}}(R,U) = q \cdot g^{\text{LP}}(R,U) + (1-q)\cdot\frac{(1+\epsilon)^2 R}{U}$. *Then* $\max_{(R,U)\in\mathcal{K}_{\text{feas}}} \Pi^{\text{LP}}(R,U) \geq \Pi^*$.

Proof. There exists $(R',U') \in \mathcal{K}_\epsilon$ such that $R' \leq R^* < (1+\epsilon)R'$ and $U' \leq U^* < (1+\epsilon)U'$. From the proof of Proposition 2, we know $(R',(1+\epsilon)U') \in \mathcal{K}_{\text{feas}}$. Since Problem (4) is a relaxation of Problem (2), we have:

$$g^{\text{LP}}(R',(1+\epsilon)U') \geq g(R',(1+\epsilon)U') \geq g(R^*,U^*).$$

Furthermore, $(1+\epsilon)R'/U' \geq R^*/U^*$, and we obtain the following bound:

$$\begin{aligned}\max_{(R,U)\in\mathcal{K}_{\text{feas}}} \Pi^{\text{LP}}(R,U) &\geq \Pi^{\text{LP}}(R',(1+\epsilon)U') \\ &= q\cdot g^{\text{LP}}(R',(1+\epsilon)U') + (1-q)\cdot\frac{(1+\epsilon)^2R'}{(1+\epsilon)U'} \\ &\geq q\cdot g(R^*,U^*) + (1-q)\cdot\frac{R^*}{U^*} = \Pi^*.\end{aligned}$$

□

To compute $g^{\text{LP}}(R,U)$, we consider the parametrized form of Problem (4) with optimal value $f^{\text{LP}}(\gamma,R,U)$. We can also apply Lemma 1 so that the objective function of the parametrized problem is linear in x. The parametrized problem with optimal value $f^{\text{LP}}(\gamma,R,U)$ is:

$$f^{\text{LP}}(\gamma,R,U) = \tag{5}$$
$$\max_{x\in\mathcal{X}^{\text{LP}}}\left\{\sum_{i=1}^n(\pi_i-\gamma)w_i + \sum_{k=1}^{2n-1}\left(\sum_{i\in L(k)}(\pi_i-\gamma)\,w_i\right)\cdot\left(\Delta_k-\Delta_{p(k)}\right)\cdot x_k \;\middle|\; \sum_{i=1}^n\pi_i\hat{v}_ix_i \geq R,\ \hat{v}_0+\sum_{i=1}^n\hat{v}_ix_i\leq U\right\}.$$

Claim 1 applies with a slight modification, hence $\gamma = g^{\text{LP}}(R,U)$ if and only if $f^{\text{LP}}(\gamma,R,U) = w_0\gamma$.

Problem (5) is a linear program in x, so we can take the dual. The first term in the objective function, $\sum_{i=1}^n(\pi_i-\gamma)w_i$, is a constant and needs to be added back onto the dual's objective function. We use dual variables β_1,β_2 for the two knapsack-like constraints of Problem (5). Dual variable y_k is used for the first and second constraint of $\mathcal{X}_{\text{LP}}$ relating vertex k to its parent, and dual variable z_k is used for the third constraint relating vertex k to its children. Finally, we use dual variable u_k for the upper-bound on x_k. Then the dual linear program is:

$$\min \sum_{i=1}^n(\pi_i-\gamma)w_i + \sum_{k=1}^{2n-1}u_k - R\beta_1 + (U-\hat{v}_0)\,\beta_2$$

subject to

$$\begin{aligned}&-\pi_k\hat{v}_k\beta_1+\hat{v}_k\beta_2+y_k-z_{p(k)}+u_k \geq (\pi_k-\gamma)\,w_k\cdot\left(\Delta_k-\Delta_{p(k)}\right) && \forall k\in N,\\ &y_k-y_{\ell(k)}-y_{r(k)}+z_k-z_{p(k)}+u_k \geq \left(\sum_{i\in L(k)}(\pi_k-\gamma)\,w_k\right)\cdot\left(\Delta_k-\Delta_{p(k)}\right) && \forall k\notin N\cup\{\text{root}\},\\ &-y_{\ell(\text{root})}-y_{r(\text{root})}+z_{\text{root}}+u_{\text{root}} \geq \left(\sum_{i\in N}(\pi_k-\gamma)\,w_k\right)\cdot(\Delta_k-1) && ,\\ &y,z,u,\beta\geq 0.\end{aligned}$$

Let $\mathcal{Y}$ denote the set of constraints in the dual linear program. By strong duality, the dual linear program has optimal value $f^{\text{LP}}(\gamma,R,U)$ if Problem (5) is feasible. Since the dual is linear in γ, we can treat γ as a variable. We know $\gamma = g^{\text{LP}}(R,U)$ if and only if $f^{\text{LP}}(\gamma,R,U) = w_0\gamma$, so we add an extra constraint to the dual to set the objective function equal to to $w_0\gamma$. Finally, we can scale the objective function of the dual by $1/w_0$ to solve directly for the desired γ. Our resulting problem is:

$$\min_{(y,z,u,\beta)\in\mathcal{Y}} \left\{ \gamma \;\middle|\; \left(w_0 + \sum_{i=1}^{n} w_i \right)\gamma - \sum_{k=1}^{2n-1} u_k + R\beta_1 - (U - \hat{v}_0)\,\beta_2 = \sum_{i=1}^{n} \pi_i w_i,\ \gamma \geq 0 \right\}. \tag{6}$$

Hence $g^{\text{LP}}(R,U)$ can be computed efficiently by solving Problem (6) for all $(R,U)\in\mathcal{K}_{\text{feas}}$. The constraint $\gamma\geq 0$ does not modify the feasible region when $(R,U)\in\mathcal{K}_{\text{feas}}$, because $g^{\text{LP}}(R,U)\geq g(R,U)>0$. In particular, we know $(R,U)\notin\mathcal{K}_{\text{feas}}$ if Problem (6) has optimal value of 0 or is infeasible. In the former case, the dual linear program of Problem (5) has non-positive optimal value, because the solution (y,z,u,β) of Problem (6) is feasible to the dual at $\gamma=0$ with objective value 0. This contradicts $f^{\text{LP}}(0,R,U)>0$ whenever $(R,U)\in\mathcal{K}_{\text{feas}}$.

To summarize this section, we present the upper-bound computation in Algorithm 2. Similar to Algorithm 1, we initialize a set $\mathcal{K}_{\text{LP}}$ in the second line which contains all (R,U) such that Problem (6) is feasible and returns a positive value, so that $\mathcal{K}_{\text{feas}}\subseteq\mathcal{K}_{\text{LP}}$. The solution $\bar{\Pi}^{\text{LP}}$ returned by Algorithm 2 satisfies $\bar{\Pi}^{\text{LP}}\geq\max_{(R,U)\in\mathcal{K}_{\text{feas}}}\Pi^{\text{LP}}(R,U)\geq\Pi^*$ where the first inequality is due to $\mathcal{K}_{\text{feas}}\subseteq\mathcal{K}_{\text{LP}}$ and the second inequality is due to Proposition 3.

Algorithm 2: Upper-bound to measure the practical performance of our FPTAS

```
Input   : Instance of OCA under the features tree model with desired accuracy $\epsilon$
Output: Upper-bound value $\bar{\Pi}$ such that $\bar{\Pi}\geq\Pi^*$
Construct $\mathcal{K}_\epsilon$ ;
Initialize grid $\mathcal{K}_{\text{LP}}\leftarrow\{(0,\hat{v}_0)\}$ ;
Set $\Pi^{\text{LP}}(0,\hat{v}_0)\leftarrow q\cdot\frac{\sum_{i=1}^n\pi_i w_i}{w_0+\sum_{i=1}^n w_i}$ ;
for $(R,U)\in\mathcal{K}_\epsilon\backslash\{(0,\hat{v}_0)\}$ do
    Solve Problem (6) at $(R,U)$ for $\gamma$ ;
    if Problem (6) is feasible and $\gamma>0$ then
        Update $\mathcal{K}_{\text{LP}}\leftarrow\mathcal{K}_{\text{LP}}\cup\{(R,U)\}$ ;
        Set $g^{\text{LP}}(R,U)=\gamma$ and compute $\Pi^{\text{LP}}(R,U)$ ;
end
Return $\bar{\Pi}^{\text{LP}}=\arg\max_{(R,U)\in\mathcal{K}_{\text{LP}}}\Pi^{\text{LP}}(R,U)$.
```

6. Extensions

In this section, we allow the retailer to choose his online assortment as well as his in-store assortment. We discuss the extensions with respect to the showroom setting. The techniques can be applied to the dynamic program in Subsection 4.2 if we consider the general setting instead. Let S^{ON} denote the online assortment and S^{PHY} denote the in-store assortment.

6.1. Related Online and In-store Assortments

In the first variation, the physical store shows a subset of the products in the online store. Many retailers encourage customers to order online if their style is not available in-store. We introduce binary variables y_i for $i \in N$, where $y_i = 1$ if product i is offered online and 0 otherwise. Since $S^{PHY} \subseteq S^{ON}$, we require $y_i \geq x_i$. The mathematical program for this setting is:

$$\max_{\substack{x \in \mathcal{X} \\ y \in \{0,1\}^n}} \left\{ \frac{\sum_{i=1}^n \pi_i w_i y_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i=1}^n w_i y_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}} \,\middle|\, x_i \leq y_i \; \forall i \in N \right\}. \tag{7}$$

We can parametrize Problem (7) in γ to obtain:

$$f^{\text{sub}}(\gamma) = \max_{\substack{x \in \mathcal{X}, \\ y \in \{0,1\}^n}} \left\{ \sum_{i=1}^n (\pi_i - \gamma) w_i y_i \cdot \prod_{k \in A(i)} \delta_k^{x_k} \,\middle|\, x_i \leq y_i \; \forall i \in N \right\}.$$

Following the strategy of Section 3, we seek γ such that $f^{\text{sub}}(\gamma) = w_0 \gamma$. We define value functions at each vertex to consider its subtree's contribution to the parametrized problem, because we can still split decisions over the two children of each non-leaf vertex. In particular, we can add the new constraints to $V_\gamma(k)$ to obtain a new value function $V_\gamma^{\text{sub}}(k)$:

$$V_\gamma^{\text{sub}}(k) = \max_{\substack{x \in \mathcal{X}_k, \\ y \in \{0,1\}^{|L(k)|}}} \left\{ \sum_{i \in L(k)} (\pi_i - \gamma) w_i y_i \cdot \prod_{k' \in A(i)} \delta_{k'}^{x_{k'}} \,\middle|\, x_i \leq y_i \; \forall i \in L(k) \right\}.$$

If k is a leaf, then $x_k = 1$ implies $y_k = 1$ and the base case has the same value as before. The difference lies in deciding which products in $L(k)$ to offer online when none of the products in $L(k)$ are displayed. That is, $x_k = 0$ for $k \notin N$. Since $x_k = 0$ implies $x_i = 0$ for all $i \in L(k)$, y_i can be 0 or 1 for each leaf in the subtree. We take $y_i = 1$ if and only if $(\pi_i - \gamma) w_i \cdot \Delta_k$ is non-negative, so that product i only increases the value function. This occurs when $\pi_i \geq \gamma$. Let $(\cdot)^+$ denote $\max\{0, \cdot\}$. Applying this argument to the left and right subtree of k when k is not a leaf, we obtain the following dynamic program:

$$\begin{aligned} V_\gamma^{\text{sub}}(k) &= (\pi_k - \gamma) w_k \Delta_k & \forall k \in N, \\ V_\gamma^{\text{sub}}(k) &= \max \left\{ \begin{array}{l} V_\gamma^{\text{sub}}(\ell(k)) \;+\; V_\gamma^{\text{sub}}(r(k)), \\ V_\gamma^{\text{sub}}(\ell(k)) \;+\; \Delta_k \cdot \displaystyle\sum_{i \in L(r(k))} (\pi_i - \gamma)^+ w_i, \\ \Delta_k \cdot \displaystyle\sum_{i \in L(\ell(k))} (\pi_i - \gamma)^+ w_i \;+\; V_\gamma^{\text{sub}}(r(k)) \end{array} \right\} & \forall k \notin N. \end{aligned}$$

The parametrized problem takes value $f^{\text{sub}}(\gamma) = \max\{V_\gamma^{\text{sub}}(\text{root}), \sum_{i=1}^n (\pi_i - \gamma)^+ w_i\}$. For each γ, S^{PHY} is identified by reaching the base cases and $S^{ON} = S^{PHY} \cup \{i \mid \pi_i \geq \gamma\}$.

The number of operations to compute $V_\gamma^{\text{sub}}(\text{root})$ is the same as the original $V_\gamma(\text{root})$, but transforming the numerator and denominator of the objective function into linear functions to measure the number of iterations of Newton's method is more complicated. Simply applying Lemma 1 to the objective function in Problem (7) is not sufficient and results in:

$$\frac{\sum_{i=1}^n \pi_i w_i y_i + \sum_{k=1}^{2n-1} \left(\Delta_k - \Delta_{p(k)}\right) \cdot \sum_{i \in L(k)} \pi_i w_i x_k y_i}{w_0 + \sum_{i=1}^n w_i y_i + \sum_{k=1}^{2n-1} \left(\Delta_k - \Delta_{p(k)}\right) \cdot \sum_{i \in L(k)} w_i x_k y_i}.$$

The numerator and denominator of the objective function have quadratic terms $x_k y_i$. To turn the numerator and denominator into linear functions, we introduce variables z_{ki} for every vertex k and $i \in N$ such that $z_{ki} = x_k y_i$. The relationship can be rewritten as $z_{ki} \le y_i$, $z_{ki} \le x_k$, and $z_{ki} \ge x_k + y_i - 1$ to obtain the following mathematical program:

$$\max_{\substack{x \in \mathcal{X} \\ y \in \{0,1\}^n \\ z \in \{0,1\}^{n(2n-1)}}} \left\{ \frac{\sum_{i=1}^n \pi_i w_i y_i + \sum_{k=1}^{2n-1} \left(\Delta_k - \Delta_{p(k)}\right) \cdot \sum_{i \in L(k)} \pi_i w_i z_{ki}}{w_0 + \sum_{i=1}^n w_i y_i + \sum_{k=1}^{2n-1} \left(\Delta_k - \Delta_{p(k)}\right) \cdot \sum_{i \in L(k)} w_i z_{ki}} \;\middle|\; \begin{array}{c} x_i \le y_i,\ z_{ki} \le y_i, \\ z_{ki} \le x_k,\ z_{ki} \ge x_k + y_i - 1 \end{array} \right\}.$$

The numerator and denominator of the objective function can be written as linear functions, but the feasible region becomes a subset of $\{0,1\}^{O(n^2)}$ and we have an upper-bound of $O(n^4 \log^2 n)$ iterations of Newton's method to find γ such that $f^{\text{sub}}(\gamma) = w_0 \gamma$. Hence we increase our runtime by a factor of $O(n^2)$ compared to Theorem 1.

6.2. Independent Online and In-store Assortments

In the second variation, we allow the online and in-store assortments to be independent of each other. Some stores offer in-store-only sales products that are not available online. The mathematical program is the same as Problem (7), but with the constraint $x_i \le y_i$ removed. This recognizes that a product can be offered in-store ($x_i = 1$) but not online ($y_i = 0$). The parametrized problem also drops this constraint, so the value function $V_\gamma^{\text{ind}}(k)$ is:

$$V_\gamma^{\text{ind}}(k) = \max_{\substack{x \in \mathcal{X}_k, \\ y \in \{0,1\}^{|L(k)|}}} \left\{ \sum_{i \in L(k)} (\pi_i - \gamma) w_i y_i \cdot \prod_{k' \in A(i)} \delta_{k'}^{x_{k'}} \right\}.$$

If k is not a leaf, then the dynamic program follows the structure of $V_\gamma^{\text{sub}}(k)$ because we choose the online assortment in the subtree that is not offered in-store immediately and postpone the decision in the subtree that offers at least one product in-store. The difference lies in the leaves because we can set $y_i = 0$ even when $x_i = 1$. Hence, we set $y_i = 1$ when $(\pi_i - \gamma) w_i \Delta_i$ is non-negative, so that we offer product i online when its per-unit revenue is at least the online expected revenue:

$$V_\gamma^{\text{ind}}(k) = (\pi_k - \gamma)^+ w_k \Delta_k \qquad \forall k \in N,$$

$$V_\gamma^{\text{ind}}(k) = \max \left\{ \begin{array}{l} V_\gamma^{\text{ind}}(\ell(k)) + V_\gamma^{\text{ind}}(r(k)), \\ V_\gamma^{\text{ind}}(\ell(k)) + \Delta_k \cdot \sum_{i \in L(r(k))} (\pi_i - \gamma)^+ w_i, \\ \Delta_k \cdot \sum_{i \in L(\ell(k))} (\pi_i - \gamma)^+ w_i + V_\gamma^{\text{ind}}(r(k)) \end{array} \right\} \qquad \forall k \notin N.$$

The parametrized problem takes value $f^{\text{ind}}(\gamma) = \max\{V_\gamma^{\text{ind}}(\text{root}), \sum_{i=1}^n (\pi_i - \gamma)^+ w_i\}$. For each γ, S^{PHY} is identified by reaching the base cases and $S^{ON} = \{i \mid \pi_i \geq \gamma\}$. The runtime analysis is the same as Subsection 6.1 where we require $S^{PHY} \subseteq S^{ON}$. The number of operations to compute $V_\gamma^{\text{sub}}(\text{root})$ and $V_\gamma^{\text{ind}}(\text{root})$ are the same, and we have the same objective function.

7. Numerical Study: Modeling Power of Features Tree Model

Dzyabura and Jagabathula (2017) had demonstrated empirical evidence supporting their choice model, but it is NP-hard to compute the optimal assortment even under the showroom setting. We show that we can use a features tree to approximate their model.

We use a generalization of Dzyabura and Jagabathula (2017)'s model as the ground-truth model. We test two aspects of our features tree model: its ability to approximate the true purchase probabilities and its ability to select an assortment that obtains a high percentage of the true optimal expected revenue. We also test against a third model, the benchmark model, to measure the performance of our features tree model against a simple, non-features-based choice model in the omnichannel retail setting. We run our tests for the showroom setting to focus on online customers.

We describe the ground-truth and benchmark models in this section but defer the mathematical details to Online Appendix C. For consistent notation across the three models, we use set notation S to describe the assortment being displayed in-store. Under the features tree model, the probability that a customer chooses product i is $P_i^{\text{T}}(S)$ and the expected revenue is $\Pi^{\text{T}}(S)$.

7.1. Ground-Truth and Benchmark Models

The ground-truth model that we test against is a generalization of Dzyabura and Jagabathula (2017)'s model. Suppose there are L feature classes and K feature values per class. Each product is created by a combination of one feature value per feature class. However, we do not require that a product exists for every combination, so that $n < K^L$ is possible. The ground-truth model allows any two products to share feature values over multiple classes.

In the ground-truth model, product i has initial preference weight w_i. A feature is denoted by its class-value pair, (ℓ, k), where $1 \leq \ell \leq L$ and $1 \leq k \leq K$, and is associated with a multiplier $\delta_{\ell,k}$. Similar to the features tree model, a feature is seen when at least one of the products with that feature is displayed in-store. If feature (ℓ, k) is seen and is a feature of product i, then we update the preference weight of product i by multiplying w_i by $\delta_{\ell,k}$. After updating preference weights, customers make their purchase decisions from the full assortment N and the no-purchase option according to MNL. Given a display assortment S, the probability that customers choose product i is denoted $P_i^{\text{G}}(S)$ and the expected revenue is denoted $\Pi^{\text{G}}(S)$ under the ground-truth model.

As a benchmark model, we consider a simple extension of MNL to the omnichannel retail setting by dropping the features dependence between products. Each product's preference weight depends only on whether it is displayed in-store. In this case, product i has preference weight v_i if it is seen in-store and w_i otherwise. Customers then make their purchase decisions from the full assortment N and the no-purchase option according to MNL by applying the appropriate preference weights. Given a display assortment S, the probability that customers choose product i is denoted $P_i^{\mathrm{B}}(S)$ and the expected revenue is denoted $\Pi^{\mathrm{B}}(S)$ under the benchmark model.

7.2. Test of Predictive Ability

First, we test the features tree model's ability to approximate the true purchase probabilities if customers make purchase decisions according to the ground-truth model. To test the predictive ability of our features tree model, we generate many instances of the ground-truth model. For each instance, we generate purchase data by randomly selecting display assortments and the resulting purchase under the ground-truth model. We fit a features tree to each instance, and for a second set of display assortments, measure the difference between purchase probabilities under the features tree model and the ground-truth model.

7.2.1. Setup An instance of the ground-truth model consists of L feature classes and K feature values per class, where $L \in \{2,3\}$ and $K \in \{4,5,6\}$. The retailer offers $n = 15$ products, which are selected uniformly at random out of the K^L possible combinations of features. The features of product i is denoted $A(i)$. Each feature is associated with a utility value, and a product's utility is the sum of its features' utilities. Feature (ℓ,k) has utility $\mu_{\ell,k}$, such that $e^{\mu_{\ell,k}}$ is generated uniformly over $[0,2]$. Following the interpretation of preference weights in MNL, the preference weight of product i is the exponential of its utility, such that $w_i = e^{\sum_{(\ell,k)\in A(i)} \mu_{\ell,k}}$. Finally, each feature (ℓ,k) is also associated with a multiplier $\delta_{\ell,k}$, which is generated uniformly over $[0.1, 1.9]$. The preference weight of the no-purchase option w_0 is chosen so that a customer who sees the full assortment N on display will walk out empty-handed with probability 0.1.

To obtain data from an instance of the ground-truth model, we generate $D = 2500$ assortments to display in-store. For each $d = 1, \ldots, D$, an assortment S_d is generated from a binomial distribution such that each product is displayed with probability 0.1. From each assortment S_d, we generate the customer's purchase decision i_d, which is either the no-purchase option or one of the n products, according to the purchase probabilities under the ground-truth model.

The structure of the features tree depends on how we order the features. We start with $L!$ features trees, where the layers of each tree corresponds to a permutation of the L feature classes. The parameters of each features tree are computed by solving the features tree's corresponding maximum likelihood estimator. As the maximum likelihood estimator might not be concave, we use Ipopt (Wächter and Biegler (2006)) to find a local optimal solution. We choose the features tree with the highest log-likelihood when we substitute its estimated parameters back into its log-likelihood function. Similarly, we estimate the parameters of the benchmark model by finding a local optimal solution to its maximum likelihood estimator.

To test the performance of the features tree model and the benchmark model, we generate another $D = 2500$ assortments. The new assortments S'_d are generated in the same manner as the previous set of assortments, for $d = 1, \dots, D$. We test the performance of the features tree model by measuring the mean absolute error, MAE^{T}, between the purchase probability of each product under the ground-truth model and the features tree model. Similarly, we denote the mean absolute error of the benchmark model as MAE^{B}:

$$\text{MAE}^{\text{T}} = \frac{1}{Dn} \sum_{d=1}^{D} \sum_{i=1}^{n} \left| P_i^{\text{T}}(S'_d) - P_i^{\text{G}}(S'_d) \right|, \quad \text{and} \quad \text{MAE}^{\text{B}} = \frac{1}{Dn} \sum_{d=1}^{D} \sum_{i=1}^{n} \left| P_i^{\text{B}}(S'_d) - P_i^{\text{G}}(S'_d) \right|.$$

In addition to measuring the performance of the features tree model against the ground-truth model, we also compare the performance of the features tree model against the benchmark model. We measure the reduction in prediction error as:

$$\text{ReduceError} = \frac{\text{MAE}^{\text{B}} - \text{MAE}^{\text{T}}}{\text{MAE}^{\text{B}}} \times 100\%.$$

7.2.2. Results For each $L \in \{2, 3\}$ and $K \in \{4, 5, 6\}$, we generate $M = 100$ instances of the ground-truth model and fit a features tree model and a benchmark model according to the procedure above. For each instance $m = 1, \dots, M$, we compute MAE_m^{T}, MAE_m^{B}, and ReduceError_m. We report the average of these measures, as well as the 5th and the 95th percentile, in Table 1.

		Tree MAE^{T}			Benchmark MAE^{B}			ReduceError		
L	K	5th perc.	95th perc.	avg.	5th perc.	95th perc.	avg.	5th perc.	95th perc.	avg.
2	4	0.0065	0.0156	0.0105	0.0085	0.0201	0.0141	1.61%	52.82%	24.53%
2	5	0.0058	0.0170	0.0105	0.0082	0.0208	0.0138	3.27%	44.89%	22.89%
2	6	0.0059	0.0160	0.0093	0.0076	0.0192	0.0123	4.05%	42.93%	22.76%
3	4	0.0082	0.0215	0.0132	0.0102	0.0265	0.0168	2.99%	42.17%	20.53%
3	5	0.0076	0.0213	0.0127	0.0098	0.0268	0.0164	4.43%	55.89%	21.33%
3	6	0.0070	0.0220	0.0123	0.0099	0.0260	0.0154	2.91%	46.77%	20.13%

Table 1 Mean Absolute Error in Estimating Purchase Probability for Features Tree and Benchmark Models

Overall, our features tree model performs well in estimating the purchase probabilities of the ground-truth model. On average, we make an error of 0.0114 when estimating the purchase probability of a product. At the 95th percentile, our MAE increases to 0.0189. The features tree model significantly outperforms the benchmark model, by reducing the error in estimating purchase probabilities by 22% on average.

In Figure 2, we present the results for $L \in \{2, 3\}$ and $K = 5$ by plotting MAE_m^{T} against MAE_m^{B} for $m = 1, \dots, M$. The plots for the other (L, K) pairs look similar and are not presented. The diagonal

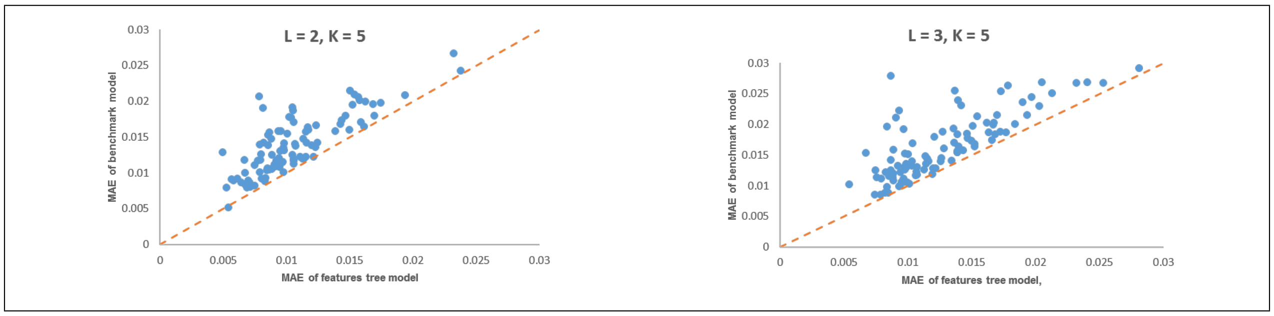

Figure 2: Comparison of the mean absolute error when using the features tree model versus the benchmark-model to approximate the ground-truth model.

$y = x$ has been plotted as a dashed line. Most of the points lie above the diagonal, which implies that on an instance-by-instance basis, the benchmark model incurs a higher mean absolute error than the features tree model. When the benchmark model outperforms the features tree model, its mean absolute error is only slightly smaller than the mean absolute error of the features tree model. In contrast, there are some instances in which the mean absolute error of the features tree model is significantly smaller than the mean absolute error of the benchmark model.

7.3. Test of Assortments Selected by Features Tree

The main advantage of the features tree model over the ground-truth model is its computation tractability. Hence we want to know how the optimal assortment obtained from a fitted features tree would perform against the true optimal assortment.

7.3.1. Setup For each instance of the ground-truth model in the previous subsection, we generate $J = 250$ scenarios by generating a new set of product prices for each scenario. The price of each product is generated uniformly at random over $[1, 10]$. As the set of prices change for each scenario, we denote the expected revenue function in scenario j as $\Pi^{\mathrm{G}}_j(\cdot)$, because we are concerned with the expected revenue of each assortment under the ground-truth model.

For scenario $j = 1, \dots, J$, we compute the optimal assortment S^{T}_j under the features tree model, using the fitted features tree that was selected in the earlier test of predictive ability. We also compute the optimal solution S^{B}_j under the fitted benchmark model and the true optimal solution S^*_j by enumeration. We compare the expected revenue under the ground-truth model for each of the assortments S^{T}_j, S^{B}_j, and S^*_j. For each instance, we measure the percentage of the true optimal expected revenue earned by assortments S^{T}_j and S^{B}_j in scenarios $j = 1, \dots J$, and then average over its J scenarios. We denote this measure by $\mathrm{Earn}^{\mathrm{T}}$ and $\mathrm{Earn}^{\mathrm{B}}$ respectively:

$$\mathrm{Earn}^{\mathrm{T}} = \frac{1}{J} \sum_{j=1}^{J} \frac{\Pi^{\mathrm{G}}_j\left(S^{\mathrm{T}}_j\right)}{\Pi^{\mathrm{G}}_j\left(S^*_j\right)} \times 100\%, \quad \text{and} \quad \mathrm{Earn}^{\mathrm{B}} = \frac{1}{J} \sum_{j=1}^{J} \frac{\Pi^{\mathrm{G}}_j\left(S^{\mathrm{B}}_j\right)}{\Pi^{\mathrm{G}}_j\left(S^*_j\right)} \times 100\%.$$

We also measure the improvement that we obtain from using the features tree model over the benchmark model over all J scenarios of that instance:

$$\text{ImproveRev} = \frac{1}{J}\sum_{j=1}^{J} \frac{\Pi_j^{\text{G}}\left(S_j^{\text{T}}\right) - \Pi_j^{\text{G}}\left(S_j^{\text{B}}\right)}{\Pi_j^{\text{G}}\left(S_j^{\text{B}}\right)} \times 100\%.$$

7.3.2. Results We reuse the $M = 100$ instances of the ground-truth model from the previous subsection, as well as the fitted features tree and benchmark models. For each instance $m = 1,\dots,M$, we computed the value of Earn_m^{T}, Earn_m^{B}, and ImproveRev_m, where each instance averages the performance of its J scenarios. We report the average of these measures, as well as the 5th and the 95th percentile, in Table 2.

		Tree Earn^{T}			Benchmark Earn^{B}			ImproveRev		
L	K	5th perc.	95th perc.	avg.	5th perc.	95th perc.	avg.	5th perc.	95th perc.	avg.
2	4	92.20%	98.28%	95.87%	87.40%	97.01%	92.49%	0.63%	10.26%	4.00%
2	5	89.68%	98.31%	95.36%	85.94%	96.49%	92.33%	0.80%	8.32%	3.58%
2	6	92.53%	98.49%	96.09%	88.14%	97.09%	93.30%	1.11%	7.20%	3.22%
3	3	87.31%	97.86%	93.95%	81.47%	96.26%	90.28%	0.47%	11.86%	4.58%
3	4	87.84%	96.92%	93.53%	82.79%	95.80%	89.96%	0.24%	9.96%	4.47%
3	5	86.97%	96.97%	93.24%	80.89%	94.46%	89.50%	0.60%	14.30%	4.82%
3	6	89.24%	97.72%	93.97%	85.25%	95.41%	90.85%	0.60%	9.19%	3.85%

Table 2 Percentage of Optimal Expected Revenue Earned by Using Features Tree and Benchmark Models

Our features tree model performs quite well in choosing display assortments which capture a high percentage of the optimal expected revenue under the ground-truth model. On average, we capture more than 94% of the true optimal expected revenue. On the low end of performance at the 5th percentile, we capture 89% of the optimal expected revenue on average. Furthermore, we outperform the benchmark model by an average increase of 4% in expected revenue.

In Figure 3, we present the results for $L \in \{2,3\}$ and $K = 5$ by plotting the values of Earn_m^{T} against the values of Earn_m^{B} for $m = 1,\dots,M$. The plots for the other (L,K) pairs look similar and are not presented. The diagonal $y = x$ has been plotted as a dashed line. Most of the points in the plots lie below the diagonal, which implies that on an instance-by-instance basis, the features tree model earns a higher expected revenue than the benchmark model when averaged over the J scenarios of that instance. Even when the benchmark model outperforms the features tree model, it is by a very small margin.

To conclude this section, our tests show that the features tree model is able to approximate the ground-truth model quite well, both in terms of estimating purchase probabilities and selecting the optimal assortment to display in-store. The ground-truth model is a generalization of Dzyabura and Jagabathula (2017)'s model and hence computationally difficult to optimize over. Our features tree model is a good substitute in that it is computationally tractable while achieving a 94% of the optimal expected revenue on average.

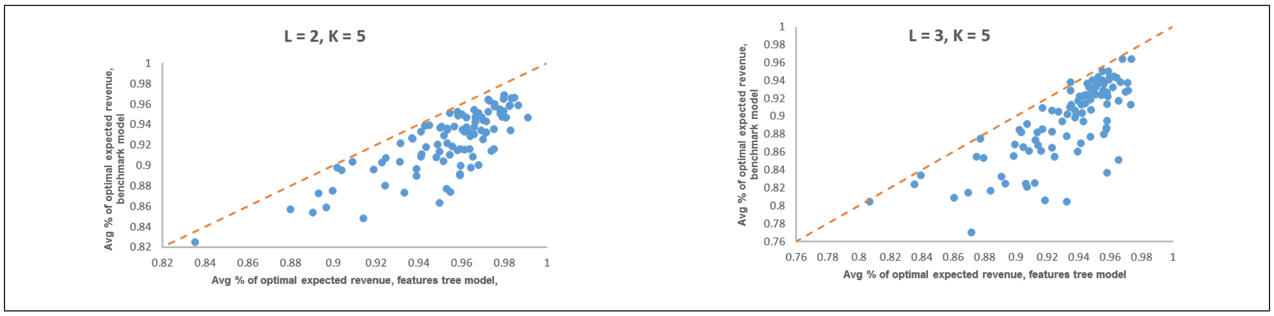

Figure 3: Average percentage of optimal expected revenue obtained when using the features tree model versus the benchmark model to compute the optimal assortment.

8. Numerical Study: Performance of FPTAS

We test our FPTAS to assess its practical performance compared to the its theoretical guarantee. We generate many instances of the OCA problem under a features tree model, and compare the expected revenue earned by our FPTAS against the upper-bound described in Section 5.

8.1. Setup

We assume that offline and online customers have the same preference weight for product i if they both see it. In other words, $\hat{v}_i = w_i \cdot \prod_{k\in A(i)} \delta_k$ and $\hat{v}_0 = w_0$. This focuses our test on the omnichannel aspect so that offline and online customers only differ in terms of what they are willing to purchase.

For each instance, we generate 32 products. Each product's price π_i is generated uniformly over $[1, 10]$. Each product's initial preference weight w_i is generated uniformly over $[1, 5]$.

Recall from Assumption 1 that our features tree T is a binary tree. Suppose vertex k is not a leaf nor the root, and feature k has multiplier $\delta_k = 1$. Then we can contract the edge between k and its parent on T to obtain an equivalent features tree that is not binary. We use this idea to construct a general features tree. First, we set up a balanced binary tree with 32 products as leaves. For a leaf $i \in N$, a product-specific multiplier δ_i is generated uniformly over $[0.1, 1.9]$. For each of the 31 non-leaf vertices, its feature multiplier δ_k is set to 1 with probability β and generated uniformly over $[0.1, 1.9]$ with probability $1-\beta$. Here, β is a parameter we vary in $\{0, 0.05, 0.1, 0.2\}$.

The no-purchase option is chosen so that a customer who sees the full assortment will refuse to purchase with probability $\eta \in \{0.05, 0.1\}$. Hence we set $w_0 = \hat{v}_0 = \eta \cdot \sum_{i=1}^{n} \hat{v}_i/(1-\eta)$.

For each instance, we vary the fraction of online customers q. For each value of $q \in \{0.2, 0.4, 0.6, 0.8\}$, let x^q denote the assortment returned by our FPTAS with a performance guarantee of $\epsilon = 1/2$. Let $\Pi^q(\cdot)$ denote the expected revenue function and $\bar{\Pi}^q$ denote the upper-bound from Section 5. We measure the percentage of the upper-bound earned by our FPTAS solution:

$$\text{EarnLP}^q = \frac{\Pi^q(x^q)}{\bar{\Pi}^q} \times 100\%.$$

In our code, we used the constrained longest path formulation in Online Appendix B to take advantage of the $O(n^2/\epsilon^2)$ improvement in runtime.

8.2. Results

For each combination of $(\eta, \beta) \in \{0.05, 0.1\} \times \{0, 0.05, 0.1, 0.2\}$, we generate $M = 50$ instances of the problem. For each instance m, we compute EarnLP_m^q and report the results in Table 3. Each row describes a combination of $(\eta, \beta) \in \{0.05, 0.1\} \times \{0, 0.05, 0.1, 0.2\}$. The proceeding four blocks summarize the average, minimum, and maximum of EarnLP_m^q for varying values of $q \in \{0.2, 0.4, 0.6, 0.8\}$. Our code computes the FPTAS assortment for all values of q at the same time, so only one average runtime is reported in each row. We run our FPTAS with a performance guarantee of $\epsilon = 1/2$, but make our grid 30 times finer ($\epsilon = 1/60$) when we construct the upper-bound.

η	β	$q = 0.2$			$q = 0.4$			$q = 0.6$			$q = 0.8$			runtime
		avg.	min.	max.	avg.	min.	max.	avg.	min.	max.	avg.	min.	max.	(min)
5%	0%	97.3%	93.0%	99.3%	98.3%	96.0%	99.4%	98.9%	97.1%	99.7%	99.5%	96.4%	99.8%	6.67
5%	5%	97.1%	93.7%	99.2%	98.2%	96.3%	99.4%	98.9%	97.3%	99.7%	99.6%	97.5%	99.9%	6.99
5%	10%	97.3%	92.9%	99.4%	98.2%	95.1%	99.5%	98.9%	97.2%	99.7%	99.6%	98.8%	99.9%	6.95
5%	20%	96.9%	93.6%	99.2%	98.2%	95.8%	99.5%	98.9%	97.0%	99.6%	99.5%	98.5%	99.9%	6.54
10%	0%	96.8%	93.2%	99.3%	97.9%	95.9%	99.6%	98.9%	97.7%	99.7%	99.6%	99.0%	99.9%	5.11
10%	5%	96.8%	93.9%	99.2%	98.1%	94.9%	99.2%	99.1%	96.7%	99.7%	99.6%	98.5%	99.9%	4.81
10%	10%	97.1%	93.5%	99.3%	98.2%	95.8%	99.3%	99.0%	97.5%	99.6%	99.7%	99.1%	99.9%	4.73
10%	20%	96.8%	93.3%	99.2%	98.1%	96.6%	99.4%	99.0%	97.4%	99.7%	99.5%	96.1%	99.9%	4.74

Table 3 Average, worst-case, and best-case performance of FPTAS on 50 instances of OCA under features-tree model with $\epsilon = 1/2$. Time reported is for computing the FPTAS solution.

Our FPTAS achieves at least 96% of the upper-bound on average, even though we ran our FPTAS with a performance guarantee of 1/2. The upper-bound uses a linear relaxation of our OCA problem and is not tight. In the worst case above, our FPTAS achieves 92.9% of the upper-bound. The average runtime of our FPTAS is 5.82 minutes, which translates to 349 seconds of CPU time.

9. Conclusion

We considered the assortment optimization problem of an omnichannel retailer, who must consider how his in-store assortment affects customers' preferences when they purchase from the full assortment in his online store. Our features tree is a simple and intuitive method to group products by features; this structure allowed us to develop a FPTAS to compute an approximately optimal assortment in the general setting via dynamic programming. We considered two extensions where the retailer chooses both his online and in-store assortments, and showed that they can be solved efficiently via a small modification to our dynamic program.

We fitted our features tree model to data simulated from a generalized model. We showed that the fitted tree was able to approximate the true purchase probabilities of products with a mean absolute error of 0.01 and select assortments achieving 94% of the optimal expected revenue on average. When we ran our FPTAS over numerical examples with a performance guarantee of 1/2 of the optimal expected revenue, we achieved 96% of the optimal expected revenue on average.

One future research direction is pricing in the omnichannel environment. Some retailers post different prices online and in-store, or offer promotions over one channel and not the other. Although previous papers have considered pricing for one product, pricing has not been considered in terms of assortments. Rather than simply using prices to modify customers' utilities, as is standard in MNL literature, perhaps an OCA pricing problem should also modify the fraction of customers purchasing online or in-store based on the average prices in the channels. This would represent the omnichannel environment more accurately, where customers are not loyal to one channel.

Assortment optimization in the omnichannel environment is quite new, and there could be better ways to explain how customers update preferences when they can research on either or both channels. Our model assumed that customers gather information from both channels and update preferences strictly based on the features tree. More general ways to update product preferences can be examined. Other choice models, like the nested logit model or the Markov chain model, could also be modified for the omnichannel setting.

Finally, parameter estimation is an important aspect of assortment optimization since representing customers' preferences accurately is a critical step before solving for the optimal assortment. In our numerical experiments, we used the local optimal solutions to the maximum likelihood estimators. A future research direction is to obtain real-world data from an omnichannel retailer and test our model and estimator on the data to evaluate the usefulness of the features tree model in practice.

References

Albadvi, A. and Shahbazi, M. (2009), 'A hybrid recommendation technique based on product category attributes', *Expert Systems with Applications* **36**(9), 11480–11488.

Avery, J., Steenburgh, T. J., Deighton, J. and Caravella, M. (2012), 'Adding bricks to clicks: Predicting the patterns of cross-channel elasticities over time', *Journal of Marketing* **76**(3), 96–111.

Bachrach, D. G., Ogilvie, J., Rapp, A. and Calamusa IV, J. (2016), *More Than a Showroom: Strategies for Winning Back Online Shoppers*, Springer.

Balakrishnan, A., Sundaresan, S. and Zhang, B. (2014), 'Browse-and-switch: Retail-online competition under value uncertainty', *Production and Operations Management* **23**(7), 1129–1145.

Bell, D., Gallino, S. and Moreno, A. (2014), 'How to win in an omnichannel world', *MIT Sloan Management Review* **56**(1), 45.

Bell, D., Gallino, S. and Moreno, A. (2015), 'Showrooms and information provision in omni-channel retail', *Production and Operations Management* **24**(3), 360–362.

Bell, D., Gallino, S. and Moreno, A. (2017), 'Offline showrooms in omnichannel retail: Demand and operational benefits', *Management Science* .

Blanchet, J., Gallego, G. and Goyal, V. (2016), 'A markov chain approximation to choice modeling', *Operations Research* **64**(4), 886–905.

Blue Nile (2018), 'FAQs: About Blue Nile', https://www.bluenile.com/contact-us/faq. Accessed: 2018-05-29.

Bront, J. J. M., Méndez-Díaz, I. and Vulcano, G. (2009), 'A column generation algorithm for choice-based network revenue management', *Operations Research* **57**(3), 769–784.

Cho, Y. H., Kim, J. K. and Kim, S. H. (2002), 'A personalized recommender system based on web usage mining and decision tree induction', *Expert systems with Applications* **23**(3), 329–342.

Désir, A., Goyal, V. and Zhang, J. (2014), 'Near-optimal algorithms for capacity constrained assortment optimization'.

Dzyabura, D. and Jagabathula, S. (2017), 'Offline assortment optimization in the presence of an online channel', *Management Science* .

Feldman, J. and Topaloglu, H. (2017), 'Revenue management under the markov chain choice model', *Operations Research* **65**(5), 1322–1342.

Fornari, E., Fornari, D., Grandi, S., Menegatti, M. and Hofacker, C. F. (2016), 'Adding store to web: migration and synergy effects in multi-channel retailing', *International Journal of Retail & Distribution Management* **44**(6), 658–674.

Gangurde, S. R. and Akarte, M. M. (2015), 'Segmentation based product design using preferred features', *Benchmarking* **22**(6), 1096–1114.

Gao, F. and Su, X. (2016*a*), 'Omnichannel retail operations with buy-online-and-pick-up-in-store', *Management Science* **63**(8), 2478–2492.

Gao, F. and Su, X. (2016*b*), 'Online and offline information for omnichannel retailing', *Manufacturing & Service Operations Management* **19**(1), 84–98.

Li, G., Rusmevichientong, P. and Topaloglu, H. (2015), 'The d-level nested logit model: Assortment and price optimization problems', *Operations Research* **63**(2), 325–342.

Li, H. and Huh, W. T. (2011), 'Pricing multiple products with the multinomial logit and nested logit models: Concavity and implications', *Manufacturing & Service Operations Management* **13**(4), 549–563.

Luce, R. D. (1959), *Individual Choice Behavior a Theoretical Analysis*, John Wiley and sons.

McFadden, D. (1973), 'Conditional logit analysis of qualitative choice behavior'.

Radzik, T. (1998), 'Fractional combinatorial optimization', *Handbook of Combinatorial Optimization* pp. 429–478.

Rusmevichientong, P., Shmoys, D., Tong, C. and Topaloglu, H. (2014), 'Assortment optimization under the multinomial logit model with random choice parameters', *Production and Operations Management* **23**(11), 2023–2039.

Wächter, A. and Biegler, L. T. (2006), 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming', *Mathematical programming* **106**(1), 25–57.

Ziegler, C.-N., Lausen, G. and Schmidt-Thieme, L. (2004), Taxonomy-driven computation of product recommendations, *in* 'Proceedings of the thirteenth ACM international conference on Information and knowledge management', ACM, pp. 406–415.

Appendix A: Proofs Omitted from Paper

Proof of Claim 1: We only prove the first case because the other two cases are similar. Let x^* be the optimal solution in the showroom setting, such that

$$\gamma^* = \frac{\sum_{i=1}^n \pi_i w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k^*}}{w_0 + \sum_{i=1}^n w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k^*}} > \gamma.$$

We can multiply both sides by the denominator as $w_0 > 0$ and rearrange terms to obtain:

$$\sum_{i=1}^n (\pi_i - \gamma) w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k^*} > w_0 \gamma.$$

The left side of the above inequality is upper-bounded by $f(\gamma)$.

□

Proof of Lemma 1: Since x is integral and $x_{p(k)} \geq x_k$ for all $k \in A(i)\backslash\{\text{root}\}$, either $x_i = 1$, $x_{\text{root}} = 0$, or there exists some $\bar{k}$ such that $x_{\bar{k}} = 0$ and $x_{p(\bar{k})} = 1$ for $\bar{k} \in A(i)\backslash\{\text{root}\}$. The result clearly holds for the first two cases, so we proceed to the third case. Then $x_k = 1$ for all $k \in A(p(\bar{k}))$ and $x_k = 0$ for $k \in A(i)\backslash A(p(\bar{k}))$, and the right side simplifies to:

$$\begin{aligned} w_i + w_i \cdot \sum_{k\in A(i)} (\Delta_k - \Delta_{p(k)}) \cdot x_k &= w_i + w_i \cdot \sum_{k\in A(p(\bar{k}))} (\Delta_k - \Delta_{p(k)}) \\ &= w_i + w_i \cdot \Delta_{p(\bar{k})} - w_i \\ &= w_i \prod_{k\in A(p(\bar{k}))} \delta_k \\ &= w_i \prod_{k\in A(i)} \delta_k^{x_k}. \end{aligned}$$

Therefore, we can rewrite $P_i^{ON}(x)$ and the expected online revenue as a fraction of linear functions.

$$P_i^{ON}(x) = \frac{w_i + w_i \cdot \sum_{k\in A(i)} (\Delta_k - \Delta_{p(k)}) x_k}{w_0 + \sum_{j=1}^n \Big(w_j + w_j \cdot \sum_{k\in A(j)} (\Delta_k - \Delta_{p(k)}) x_k \Big)},$$

$$\begin{aligned} \sum_{i\in N} \pi_i P_i^{ON}(x) &= \frac{\sum_{i=1}^n \pi_i w_i + \sum_{i=1}^n \pi_i w_i \cdot \sum_{k\in A(i)} (\Delta_k - \Delta_{p(k)}) x_k}{w_0 + \sum_{i=1}^n w_i + \sum_{i=1}^n w_i \cdot \sum_{k\in A(i)} (\Delta_k - \Delta_{p(k)}) x_k} \\ &= \frac{\sum_{i=1}^n \pi_i w_i + \sum_{k=1}^{2n-1} \Big(\sum_{i\in L(k)} \pi_i w_i \Big) \cdot \big(\Delta_k - \Delta_{p(k)} \big) \cdot x_k}{w_0 + \sum_{i=1}^n w_i + \sum_{k=1}^{2n-1} \Big(\sum_{i\in L(k)} w_i \Big) \cdot \big(\Delta_k - \Delta_{p(k)} \big) \cdot x_k}. \end{aligned}$$

The last equality switches the summation from products to vertices in the features tree T.

□

Proof of Proposition 1: The OCA problem under the features tree model is in NP: given characteristic vector x, we can easily determine if "$\Pi(x) \geq K$" for any K. By contradiction, suppose that for any instance of our problem and any K, we can determine if the optimal expected revenue is at least K. We will show NP-hardness via a reduction from the partition problem.

In the partition problem, we are given integers $\{c_1, \ldots, c_n\}$ such that $\sum_{i=1}^n c_i = 2C$ for some integer C. The problem requires us to determine if there exists a subset $S \subseteq N$ such that $\sum_{i\in S} c_i = C$. Our proof constructs an instance of the OCA problem such that an optimal assortment will give us a solution to the partition problem if it exists.

The portion of online customers is $q = \frac{2C^2+C}{5C^2+4C+1} + \frac{1}{2}$, and the portion of offline customers is $1-q = \frac{3C^2+3C+1}{5C^2+4C+1} - \frac{1}{2}$. It is easy to see that $q \in (0,1)$ for any $C \geq 1$.

Construct n products, each with revenue $\pi_i = 1$. An online customer associates an initial preference weight $w_i = 2c_i$ to product i. It will be easier to consider a general tree rather than a binary tree for the purpose of this proof. Our tree has $n+1$ vertices, consisting of n leaves and a root. Each leaf has a multiplier $\delta_i = 1/2$, and the root has a multiplier $\delta_{\text{root}} = 1$. Since $\delta_{\text{root}} = 1$, it never affects online preference weights and we can ignore x_{root}. Our feasible region reduces to $\mathcal{X} = \{0,1\}^n$. Given $x \in \{0,1\}^n$, the resulting preference weights of online customers for product i is $v_i(x) = 2c_i \left(\frac{1}{2}\right)^{x_i} = c_i(2-x_i)$. The second equality can be obtained by applying Lemma 1. An offline customer has preference weight $\hat{v}_i = c_i$ for product i. Both customer types have a preference weight of 1 for the no-purchase option.

We claim that a partition of $\{c_i \mid i \in N\}$ exists if and only if the optimal expected revenue is greater or equal to $K = (5C^2+2C)/(5C^2+4C+1)$. The expected revenue of $x \in \{0,1\}^n$ is:

$$\begin{aligned}
\Pi(x) &= q \cdot \frac{\sum_{i=1}^n \pi_i v_i(x)}{1+\sum_{i=1}^n v_i(x)} + (1-q) \cdot \frac{\sum_{i=1}^n \pi_i \hat{v}_i x_i}{1+\sum_{i=1}^n \hat{v}_i x_i} \\
&= q \cdot \frac{\sum_{i=1}^n c_i(2-x_i)}{1+\sum_{i=1}^n c_i(2-x_i)} + (1-q) \cdot \frac{\sum_{i=1}^n c_i x_i}{1+\sum_{i=1}^n c_i x_i} \\
&= q \cdot \frac{4C - \sum_{i=1}^n c_i x_i}{1+4C-\sum_{i=1}^n c_i x_i} + (1-q) \cdot \frac{\sum_{i=1}^n c_i x_i}{1+\sum_{i=1}^n c_i x_i}.
\end{aligned}$$

The last line is true because $\sum_{i=1}^n c_i = 2C$. Let $y = 2C - \sum_{i=1}^n c_i x_i$ and rewrite the expected revenue as a function $f(y)$, with q expanded out:

$$f(y) = \left(\frac{2C^2+C}{5C^2+4C+1} + \frac{1}{2}\right) \cdot \frac{2C+y}{1+2C+y} + \left(\frac{3C^2+3C+1}{5C^2+4C+1} - \frac{1}{2}\right) \cdot \frac{2C-y}{1+2C-y}.$$

The first and second derivatives are:

$$f'(y) = \left(\frac{2C^2+C}{5C^2+4C+1} + \frac{1}{2}\right) \cdot \frac{1}{(1+2C+y)^2} - \left(\frac{3C^2+3C+1}{5C^2+4C+1} - \frac{1}{2}\right) \cdot \frac{1}{(1+2C-y)^2}, \text{ and}$$

$$f''(y) = -\left(\frac{2C^2+C}{5C^2+4C+1} + \frac{1}{2}\right) \cdot \frac{2}{(1+2C+y)^3} - \left(\frac{3C^2+3C+1}{5C^2+4C+1} - \frac{1}{2}\right) \cdot \frac{2}{(1+2C-y)^3}.$$

Since $f''(y) < 0$ for all $y \in [0, 2C]$, any solution $\bar{y}$ such that $f'(\bar{y}) = 0$ must be a unique maximum over this interval. We can verify that $f'(C) = 0$ and that:

$$\begin{aligned}
f(C) &= \left(\frac{2C^2+C}{5C^2+4C+1} + \frac{1}{2}\right) \cdot \frac{3C}{1+3C} + \left(\frac{3C^2+3C+1}{5C^2+4C+1} - \frac{1}{2}\right) \cdot \frac{C}{1+C} \\
&= \frac{5C^2+2C}{5C^2+4C+1} = K.
\end{aligned}$$

The optimal expected revenue is upper-bounded by $f(C) = K$ and this is a unique maximum on the interval $[0, 2C]$. Rearranging $C = y = 2C - \sum_{i=1}^n c_i x_i$ gives us $C = \sum_{i=1}^n c_i x_i = \sum_{i \in S} c_i$. Hence the optimal expected revenue is greater or equal to K if and only if there exists some partition S such that $\sum_{i \in S} c_i = C$. This shows the OCA problem under the features tree model is NP-hard in the general setting.

□

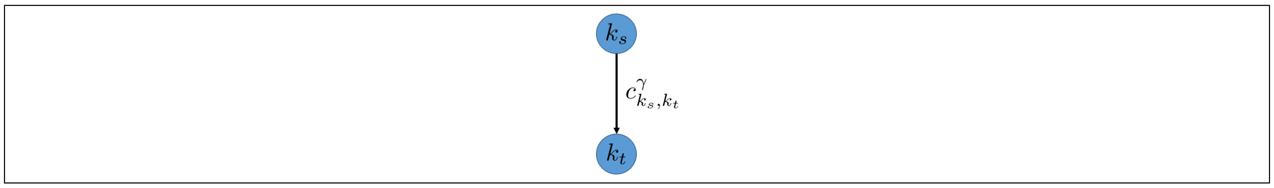

Figure 4: G_k when $k \in N$, with arc cost $c^{\gamma}_{k_s,k_t} = (\pi_k - \gamma) w_k \Delta_k$.

Appendix B: Faster Algorithm via Longest Path in a Directed Acyclic Graph

One of the biggest contribution to the high runtime of our dynamic program in the general setting is the number of decisions that are considered. In this section, we show a constrained longest path problem on a directed acyclic graph (DAG) that is equivalent to solving Problems (1) and (3). This structure reduces the number of decisions per state and significantly reduces the runtime of our FPTAS by a factor of $O(n^2/\epsilon^2)$. We explain the intuition at the end of the section.

In the constrained longest path problem, we are given a graph G with a source s and a sink t. The arcs of G have costs c, and L sets of weights $d^1, \dots d^L$ with corresponding budget D^l. Let $c(P)$ and $d^l(P)$ denote the length of P under costs c and weights d^l respectively: $c(P) = \sum_{(u,v)\in P} c_{u,v}$ and $d^l(P) = \sum_{(u,v)\in P} d^l_{u,v}$. Then the problem is to find a s-t path P that maximizes $c(P)$ subject to $d^l(P) \le D_l$ for all $l = 1, \dots, L$.

We proceed by showing the longest path construction for Problem (1). We present a DAG G that has a bijection between every s-t path and solution $x \in \mathcal{X}$. As a result, we can efficiently compute $f(\gamma)$ by finding the longest path in G via the standard dynamic program. The extension to the general setting simply requires putting two set of weights on the arcs of G to form a constrained longest path problem. Even though the constrained longest path problem is NP-hard in general, our instance can be solved efficiently because we use the rounded parameters from Problem (3), which are integers of size $O(n/\epsilon)$.

B.1. Showroom Setting

To solve Problem (1), we construct a directed graph $G = (V, E)$ with arc costs c^γ based on our features tree T and parametrized for γ. For every vertex k in T, we will have two vertices in G: k_s and k_t. We also add a source s and a sink t into G. Hence, $V(G) = \{s, t\} \cup \{k_s, k_t | \, k \in V(T)\}$.

Let G_k be the subgraph of G induced by the set of vertices k'_s, k'_t where k' is a feature in the subtree rooted at k. The subgraphs are built recursively from the leaves of T up to the root. We start from $k \in N$ and create arc (k_s, k_t) with cost $c^{\gamma}_{k_s,k_t} = (\pi_k - \gamma) w_k \Delta_k$. So G_k has two vertices and one arc (see Figure 4).

To build G_k where $k \notin N$, recall that k has two children: $\ell = \ell(k)$ and $r = r(k)$. Both G_ℓ and G_r have already been built, and we add five arcs to complete G_k (see Figure 5). In particular, arcs (k_s, ℓ_s) and (k_s, r_s) go from the source of G_k to the sources of G_ℓ and G_r respectively. Arcs (ℓ_t, k_t) and (r_t, k_t) go from the sinks of G_ℓ and G_r to the sink of G_k. We also join the sink of G_ℓ to the source of G_r with arc (ℓ_t, r_s). Arcs (k_s, ℓ_s), (ℓ_t, r_s), and (r_t, k_t) have 0 costs. The other two arcs have costs:

$$c^{\gamma}_{k_s,r_s} = \Delta_k \cdot \left(\sum_{i \in L(\ell)} (\pi_i - \gamma) w_i \right), \quad \text{and} \quad c^{\gamma}_{\ell_t,k_t} = \Delta_k \cdot \left(\sum_{i \in L(r)} (\pi_i - \gamma) w_i \right).$$

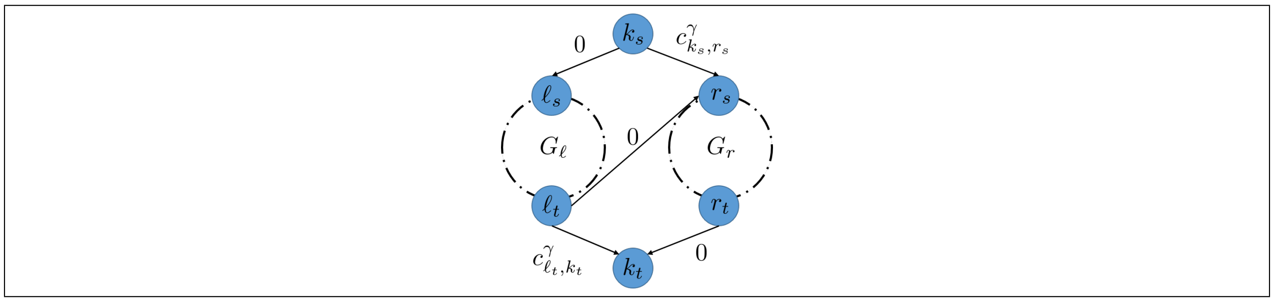

Figure 5: G_k when $k \notin N$, with arc costs $c^{\gamma}_{k_s,r_s} = \Delta_k \cdot \left(\sum_{i\in L(\ell)}(\pi_i - \gamma)w_i\right)$ and $c^{\gamma}_{\ell_t,k_t} = \Delta_k \cdot \left(\sum_{i\in L(r)}(\pi_i - \gamma)w_i\right)$.

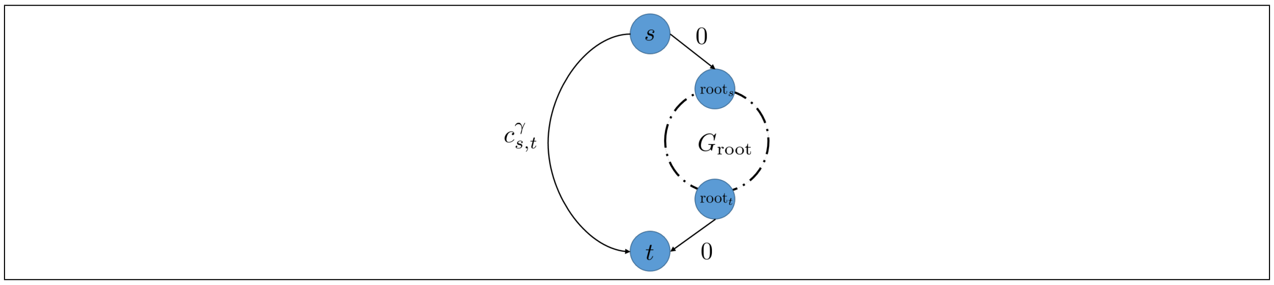

Figure 6: DAG G, with arc cost $c^{\gamma}_{s,t} = \sum_{i=1}^{n}(\pi_i - \gamma)w_i$.

Intuitively, a unit of flow entering G_k corresponds to feature k being demonstrated in-store. This unit of flow can do one of three things. First, it can pass through G_ℓ and then G_r, which corresponds to both children features being demonstrated in-store. Second, it can pass through G_ℓ but not G_r. In this case, the cost of $c^{\gamma}_{\ell_t,k_t}$ is incurred, which accounts for the fixed contribution by the right subtree when the left feature is demonstrated but not the right feature. Third, it can pass through G_r but not G_ℓ, in which case the cost of $c^{\gamma}_{k_s,r_s}$ is incurred, and accounts for the fixed contribution by the left subtree. It is not possible for the unit of flow to neither enter G_ℓ nor G_r, which corresponds to demonstrating at least one child feature.

Finally, we connect G_{root} to the source s and sink t by adding arcs (s, root_s), (root_t, t), and (s,t) (see Figure 6). The first two arcs have 0 costs and the third arc has $c^{\gamma}_{s,t} = \sum_{i=1}^{n}(\pi_i - \gamma)w_i$. Our graph G has $|V(G)| = 2 \cdot (2n-1) + 2 = O(n)$ vertices and $|E(G)| = n + 5 \cdot (n-1) + 3 = O(n)$ arcs. Hence G can be created in polynomial time.

The construction of G implies a bijection between solutions $x \in \mathcal{X}$ and s-t paths P in G, such that $x_k = 1$ if and only if $(k_s, k_t) \in P$ for $k \in N$. The empty assortment maps to the single-edge path $P = (s,t)$. Furthermore the objective value of x in Problem (1) is equal to the length of the path P under costs c^{γ}.

As G is a DAG, it is well-known that the longest path can be computed via dynamic programming. Let $C_\gamma(u)$ be the cost of the longest s-u path in G under cost vector c^{γ}. Solving Problem (1) for the optimal solution x requires us to compute the longest path P on G, which gives us $f(\gamma) = C_\gamma(t)$. Since $C_\gamma(t)$ already considers displaying the empty assortment via the (s,t) arc, this solution does not have to be computed separately as in Section 3. We solve the following dynamic program to compute $C_\gamma(t)$:

$$C_\gamma(s) = 0,$$

$$C_\gamma(u) = \max_{v:(v,u)\in E(G)} C_\gamma(v) + c^\gamma_{v,u} \qquad \forall u \neq s.$$

Computing $C_\gamma(t)$ involves computing $C_\gamma(u)$ at $O(n)$ states. There are at most two decisions to consider for each $C_\gamma(u)$, because each vertex $u \in V(G)$ has at most two incoming arcs. Hence, we can compute $C_\gamma(t)$ in $O(n)$ operations. Since we can compute $f(\gamma)$ efficiently, we can use Newton's method to search for γ^* such that $f(\gamma^*) = w_0\gamma^*$ in $O(n^2 \log^2 n)$ iterations of solving $C_\gamma(t)$. Our total runtime remains $O(n^3 \log^2 n)$ in the showroom setting.

B.2. General Setting

Next, we move onto the problem of solving Problem (3) for $\tilde{f}(\gamma, R, U)$ when inputs have been rounded appropriately for any (R, U) pair. In order to incorporate the two constraints of Problem (3) into our longest path problem, we add two sets of weights, d^1 and d^2, to our graph G and turn it into a constrained longest path problem. We will ensure that our path length is at least Y_1 under weights d^1 and no longer than Y_2 under weights d^2. Due to the bijection between s-t paths in G and solutions $x \in \mathcal{X}$, we know that $x_k = 1$ if and only if the arc (k_s, k_t) is in the path P for $k \in N$. The obvious way to construct d^1 and d^2 in G_{root} are:

$$d^1_{u,v} = \begin{cases} \tilde{\pi}^R_k & : (u,v) = (k_s, k_t), k \in N \\ 0 & : \text{otherwise} \end{cases},$$
$$d^2_{u,v} = \begin{cases} \tilde{v}^U_k & : (u,v) = (k_s, k_t), k \in N \\ 0 & : \text{otherwise} \end{cases}.$$

To incorporate the weight of the no-purchase option, $\tilde{v}^U_0$, into the path length under d^2, we add weights to the three arcs in $E(G)\backslash E(G_{\text{root}})$:

$$d^1_{u,v} = 0, \qquad \text{for } (u,v) = (s,t), (s, \text{root}_s), (\text{root}_t, t),$$
$$d^2_{u,v} = \begin{cases} \tilde{v}^U_0 & : (u,v) = (\text{root}_t, t), (s,t) \\ 0 & : (u,v) = (s, \text{root}_s) \end{cases}.$$

An s-t path that ensures $d^1(P) \geq Y_1$ and $d^2(P) \leq Y_2$ corresponds to a solution x such that the constraints of Problem (3) are satisfied. Since Y_1 and Y_2 are integers of the order $O(n/\epsilon)$, we can set up a dynamic program to compute the longest s-t path under costs c^γ subject to the length requirements $d^1(P) \geq Y_1$ and $d^2(P) \leq Y_2$. Let $\tilde{C}^{R,U}_\gamma(u, y_1, y_2)$ denote the length of the longest s-u path P_u such that $d^1(P_u) \geq y_1$ and $d^2(P_u) \leq y_2$. Then $\tilde{f}(\gamma, R, U) = \tilde{C}^{R,U}_\gamma(t, Y_1, Y_2)$. Our dynamic program is as follows for $y_1 \in \{0, \ldots, Y_1\}$ and $y_2 \in \{0, \ldots, Y_2\}$:

$$\tilde{C}^{R,U}_\gamma(s, y_1, y_2) = \begin{cases} 0: & y_1 \leq 0, y_2 \geq 0 \\ -\infty: & \text{otherwise} \end{cases},$$
$$\tilde{C}^{R,U}_\gamma(u, y_1, y_2) = \max_{v:(v,u)\in E(G)} \left\{ c^\gamma_{v,u} + \tilde{C}^{R,U}_\gamma(v, y_1 - d^1_{v,u}, y_2 - d^2_{v,u}) \right\} \qquad \forall u \neq s.$$

When we compute $\tilde{C}^{R,U}_\gamma(u, y_1, y_2)$, we may move backward to a state where the second or third input is negative. This happens if $y_1 - d^1_{v,u} < 0$ or $y_2 - d^2_{v,u} < 0$. In the former case, the state (u, y_1, y_2) with $y_1 < 0$ means that $d^1(P_u) < 0$ and our requirement under weights d^1 has been satisfied. To reduce the number of states in our dynamic program, we can set $\tilde{C}^{R,U}_\gamma(u, y_1, y_2) = \tilde{C}^{R,U}_\gamma(u, 0, y_2)$ whenever $y_1 < 0$, because we do not have to consider weights d^1 at either states.

In the latter case, the state (u, y_1, y_2) with $y_2 < 0$ means that $d^2(P_u) < 0$ and we violate our requirement under d^2. Furthermore, the base case ensures that $\tilde{C}_\gamma^{R,U}(u, y_1, y_2) = -\infty$ if we continue to run the dynamic program, regardless of the value of y_1. Hence, we can set $\tilde{C}_\gamma^{R,U}(u, y_1, y_2) = -\infty$ whenever $y_2 < 0$ to reduce the number of states in our dynamic program.

One final difference between the path-based dynamic program and the original dynamic program is that we were computing $\tilde{V}_\gamma^{R,U}(\text{root}, Y_1, Y_2 - \tilde{v}_0^U)$, whereas we are now computing $\tilde{C}_\gamma^{R,U}(t, Y_1, Y_2)$. We do not need to adjust Y_2 by $\tilde{v}_0^U$ in the path-based dynamic program because its value has been incorporated into G by arcs (s, t) and (root_t, t).

THEOREM 3. *Suppose* Π^* *is the optimal expected revenue of the OCA problem under the features tree model. For any* $\epsilon \in (0, 1)$*, there exists an algorithm that finds an in-store assortment* x *such that* $\Pi(x) \geq (1 - \epsilon)\Pi^*$ *in* $O\left(\frac{n^5 \log^2 n \log n\overline{R}/\underline{R} \cdot \log(n+1)\overline{U}/\underline{U}}{\epsilon^4}\right)$ *operations.*

Proof. The $(1 - \epsilon)$-approximation follows from the proof of Theorem 2, where we take $\epsilon' = \epsilon/6$ when creating our grid $\mathcal{K}_\epsilon$ and rounding our values.

Also similar to the proof of Theorem 2, the number of times that we need to compute $\tilde{C}_\gamma^{R,U}(t, Y_1, Y_2)$ is $O\left(\frac{n^2 \log^2 n \log n\overline{R}/\underline{R} \cdot \log(n+1)\overline{U}/\underline{U}}{\epsilon^2}\right)$.

To compute $\tilde{C}_\gamma^{R,U}(t, Y_1, Y_2)$, there are $O(n)$ vertices in G and $O(n^2/\epsilon^2)$ pairs of (y_1, y_2) for each vertex. So there are $O(n^3/\epsilon^2)$ states. For each state, the dynamic program considers at most two decisions as there are at most two incoming arcs per vertex. Therefore, finding the longest constrained path in G takes $O(n^3/\epsilon^2)$ operations and the total runtime of the modified algorithm is $O(\frac{n^5 \log^2 n \log n\overline{R}/\underline{R} \cdot \log(n+1)\overline{U}/\underline{U}}{\epsilon^4})$ operations.

□

Compared to Theorem 2, the number of operations decreased by a factor of $O(n^2/\epsilon^2)$. This is because we reduce the number of decisions at each state from $O(n^2/\epsilon^2)$ to $O(1)$, while increasing the number of states by a factor of 2. When we run the original dynamic program in Subsection 4.2 and consider state (k, y_1, y_2), we immediately decide on the allocation of (y_1, y_2) between the two children of k. This is wasteful because we do not consider the actual revenues and offline preference weights of products in the subtree, and different ways of splitting (y_1, y_2) could have the same results. Using the new dynamic program, we only decide on displaying one or both children features of k when we are at (k_t, y_1, y_2), not how we split (y_1, y_2) over each subtree. The actual allocation of (y_1, y_2) is delayed until we reach the parent of a product vertex, when we decide whether to offer the left, right, or both products. The graph G sorts vertices in the features tree according to a depth-first-search, which allows us to consider the consumption of budgets (y_1, y_2) product by product, rather than subtree by subtree like the original dynamic program.

This constrained longest path construction can be used to represent the parametrized version of Problem (2) at (R, U) before rounding parameters $\pi\hat{v}$ and $\hat{v}$. We could have simply used $\pi\hat{v}$ and $\hat{v}$ in the construction of d^1 and d^2, replacing the path constraints with $d^1(P) \geq R$ and $d^2(P) \leq U$. We present the rounded version directly to show how we can compute a solution in polynomial time.

With regards to the extensions in Section 6, where the retailer chooses both online and in-store assortments, the earlier modifications can be incorporated by applying the non-negativity function to arc costs whenever the corresponding terms are non-negative in the earlier dynamic program.

Appendix C: Ground-Truth and Benchmark Models

In this section, we give the details underlying the ground-truth and benchmark models, against which we test our features tree model.

C.1. Ground-Truth Model

Our ground-truth model for testing the modeling power of our features tree model is a generalization of the model introduced by Dzyabura and Jagabathula (2017). We do not require the online store to offer a product for every possible combination of features. For L feature classes and K feature values per class, our ground-truth model selects $n \le L^K$ products as the full assortment.

As a result of this modification, our ground-truth model differs from Dzyabura and Jagabathula (2017) because it cannot be optimized over features. Rather, we require a separate set of variables to describe which products and features are on display. Product i has initial preference weight w_i and its preference weight is updated when the features of product i are seen in-store. We use binary variables $x \in \{0,1\}^n$ to describe whether products are on display, so that $x_i = 1$ if product i is on display and 0 otherwise.

We denote features by their class-value pair (ℓ, k). We use $y \in \{0,1\}^{L \cdot K}$ to indicate which features are seen. Each feature (ℓ, k) has a boost or discount multiplier $\delta_{\ell,k}$. To be consistent with notation in our features tree model, let $A(i)$ denote the features of product i and $L(\ell, k)$ denote all the products with feature (ℓ, k). Following the same logic in Section 2, we have $y_{\ell,k} = 1$ if and only if $x_i = 1$ for some $i \in L(\ell, k)$, and the set of feasible characteristic vectors (x, y) describing the features and products on display can be denoted by $\mathcal{X}^{\mathrm{G}}$, where:

$$\mathcal{X}^{\mathrm{G}} = \left\{ (x,y) \;\middle|\; \begin{array}{ll} x_i \le y_{\ell,k} & \forall i,\ (\ell,k) \in A(i), \\ y_{\ell,k} \le \sum_{i \in L(\ell,k)} x_i & \forall (\ell,k), \\ y_{\ell,k} \in \{0,1\} & \forall (\ell,k), \\ x_i \in \{0,1\} & \forall i. \end{array} \right\}.$$

Given an assortment x, the vector y is uniquely defined. The final preference weight of product i can be written as $w_i \cdot \prod_{(\ell,k)\in A(i)} \delta_{\ell,k}^{y_{\ell,k}}$ and the probability of selling product i in assortment x is:

$$P_i^{\mathrm{G}}(x,y) = \frac{w_i \cdot \prod_{(\ell,k)\in A(i)} \delta_{\ell,k}^{y_{\ell,k}}}{w_0 + \sum_{j=1}^n w_j \cdot \prod_{(\ell,k)\in A(j)} \delta_{\ell,k}^{y_{\ell,k}}}.$$

Hence the assortment optimization problem in the showroom setting under the ground-truth model is to maximize expected revenue $\Pi^{\mathrm{G}}(x,y)$ over $\mathcal{X}^{\mathrm{G}}$, where where $\Pi^{\mathrm{G}}(x,y)$ is defined as:

$$\Pi^{\mathrm{G}}(x,y) = \frac{\sum_{i=1}^n \pi_i w_i \cdot \prod_{(\ell,k)\in A(i)} \delta_{\ell,k}^{y_{\ell,k}}}{w_0 + \sum_{i=1}^n w_i \cdot \prod_{(\ell,k)\in A(i)} \delta_{\ell,k}^{y_{\ell,k}}}.$$

C.2. Benchmark Model

Our benchmark model is the simplest model that incorporates the notion of having different preference weights for products, depending on whether they are on display. In the benchmark model, we ignore features relationship and product i simply has a preference weight of v_i when it is displayed and w_i otherwise.

We use binary variables $x \in \{0,1\}^n$ to describe the assortment being display in-store. That is, $x_i = 1$ if product i is on display and 0 otherwise. The preference weight of product i is $v_i x_i + w_i \cdot (1 - x_i)$ and the probability of selling product i when assortment x is offered is:

$$P_i^{\mathrm{B}}(x) = \frac{v_i x_i + w_i \cdot (1 - x_i)}{w_0 + \sum_{j=1}^n v_j x_j + w_j \cdot (1 - x_j)}.$$

Hence the assortment optimization problem in the showroom setting under the benchmark model is to maximize expected revenue $\Pi^{\mathrm{B}}(x)$ over $\{0,1\}^n$, where $\Pi^{\mathrm{B}}(x)$ is defined as:

$$\Pi^{\mathrm{B}}(x) = \frac{\sum_{i=1}^n \pi_i \left(v_i x_i + w_i \cdot (1 - x_i)\right)}{w_0 + \sum_{i=1}^n v_i x_i + w_i \cdot (1 - x_i)}.$$