MANUFACTURING & SERVICE OPERATIONS MANAGEMENT
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 1523-4614 | EISSN 1526-5498 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000
© 0000 INFORMS

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

Venus Lo
Department of Management Sciences, City University of Hong Kong, Hong Kong SAR, venus.hl.lo@cityu.edu.hk

Huseyin Topaloglu
School of Operations Research and Information Engineering, Cornell Tech, New York, New York 10011, 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 among the products that are offered in their preferred store. However, customers who purchase from the online store can first test out products offered in the physical store. These customers revise their preferences for online products based on the features that are shared with the in-store products. 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. Unlike traditional assortment optimization, the physical store's assortment influences 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 correspond to features of the product. Customers choose among the products within their store's assortment according to the multinomial logit model. We consider two settings: either all customers purchase online after viewing products in the physical store, 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 closely approximate the case where products have arbitrary combinations of features without a tree structure, and that our FPTAS performs remarkably well. **Managerial Implications:** We characterize conditions under which it is optimal to display expensive products with under-rated features, and expose inexpensive products with over-rated features.

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

1. Introduction

Traditionally, retailers operated on a single channel, either as offline physical stores or online stores. As online shopping has become ubiquitous, a customer can use multiple channels to research and purchase products (Bachrach et al. (2016)). In a practice known as showrooming, a customer can test out products at a local retailer before purchasing online. In response, retailers have started to sell on multiple channels. Best Buy started as a physical store, but it has progressed to operating an online store and even offers online-only products. Diamonds retailer Blue Nile started as an online store, but it has opened showrooms to display its products (Blue Nile (2018)). Literature refers to this phenomenon as an omnichannel retail environment because retailers must operate multiple channels as a cohesive unit. Since products may share features, customers can try out the products that are displayed in-store, and modify their preferences of online products based on shared features. This leads to the study of assortment optimization from an omnichannel viewpoint.

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. Products have features and we describe similarities among products by their shared features. We organize features onto a tree so that each vertex corresponds to a feature. The path from a leaf to the root gives a set of features that uniquely defines a product, so that a leaf also corresponds to a product. We refer to this tree as the features tree. Figure 1 presents twelve products from M.A.C Cosmetics organized onto a features tree. Product 4 is described by the path from the fourth leaf to the root, so that product 4 is a lipstick from the Cremesheen line in the Peach Blossom colour.

Two products share a feature if the feature's vertex is a common ancestor of the products' leaves on the tree. On Figure 1, products 1 and 2 share the feature of belonging in the Lustre line, and products 3 to 5 share the feature of belonging in the Cremesheen line. Products 1 to 7 share the feature of being a lipstick. Although we describe each product using three features, each feature can be the combination of several sub-features. The products within a product line have the same composition of ingredients, and they inherently share the same glossiness, smoothness, and effectiveness with respect to being a long-lasting and moisturizing product. The overall performance of these sub-features can be attributed to the umbrella feature of belonging to the same product line. Hence, the Lustre vertex represents all of the sub-features shared by products 1 and 2.

There are two types of customers: offline and online. Offline customers visit the physical store to purchase from the assortment that is offered in-store. Online customers visit the physical store to test out the products before purchasing from the full assortment online. By trying out the products that are available in-store, online customers can evaluate whether products' features are over- or under-rated relative to their online descriptions. These customers update their preferences for online-only products based on the features that are shared with the displayed products.

In Figure 1, suppose the retailer only offers product 4 in the physical store, which is a Peach Blossom, Cremesheen lipstick. Online customers visit the physical store and try out product 4 to see whether the product performs as advertised. Seeing this lipstick would allow customers to determine if the online depiction of the colour is accurate, and if not, whether it is better or worse than expected. Furthermore, the Cremesheen line is described online as a creamy and semi-glossy lipstick, but customers may determine that the lipstick is glossier than advertised upon testing product 4. Online customers can update their opinion on products 3 and 5 with respect to all the sub-features that are inherent to the Cremesheen line. They can also update their opinion on products 1 to 7 for the overall quality of lipsticks, and all the products for the quality of the M.A.C. brand. Customers do not update their opinion on the colour of any products other than product 4, because they cannot evaluate the colours relative to their online depiction. The features tree is best for describing products that can be categorized by levels of distinctive features (e.g. product line), such that the features in the lower levels of the tree are different across categories.

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, in order to maximize the total expected revenue.

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 a set of features. We describe similarities among products by their shared features using the features tree, so that two products share a feature if the feature's vertex is a common ancestor of the products' leaves on the tree. Offline and online customers purchase according to the multinomial logit model (MNL), and the preference weight of product i depends on the customer's purchasing channel. An offline customer chooses among the products in the physical store's assortment, and her preference weights for the products are given as input parameters and fixed. An online customer visits the physical store to test out features on the displayed products before purchasing from the full assortment online. Her preference weights are functions of the in-store assortment, and 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. We call this problem the OmniChannel Assortment optimization (OCA) problem under the features tree model.

We show that the OCA problem under the 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 customers as the "general" setting. This is common among traditional retailers that subsequently started their own online store, such as Best Buy. The physical store serves as a display front, and targets 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

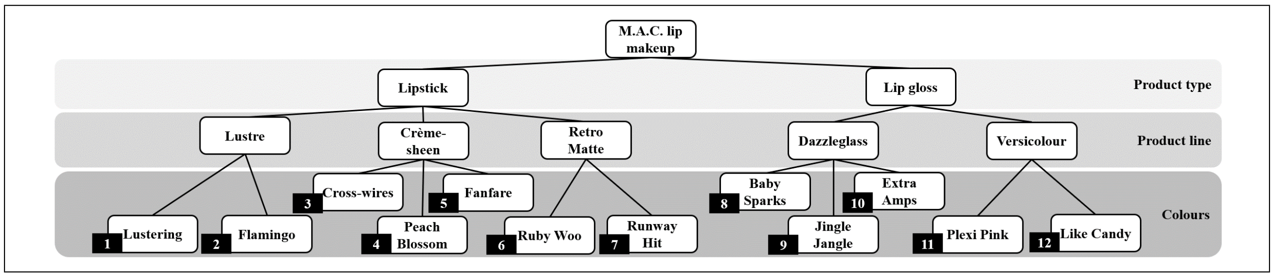
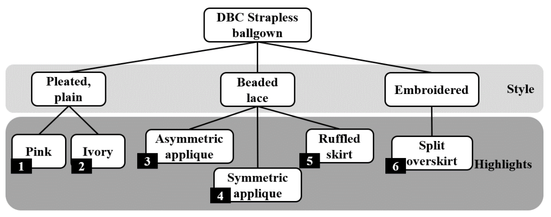


Figure 1: Twelve lip products from M.A.C Cosmetics (2019*a*,*b*), labeled by black squares. Level 2 lists the product types, level 3 lists the product lines, and level 4 lists the colours in each line.

is common among historically online retailers that subsequently opened their own showrooms. At Blue Nile, a customer can see sample rings and test out different sizes and settings in a physical store, but she can only order her customized ring online. We present an algorithm that finds the optimal display assortment with runtime polynomial in the number of products. We give sufficient conditions under which it is optimal to display a revenue-ordered subset of products with under-rated features and a reverse revenue-ordered subset of products with over-rated features.

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 of the offline customers. For each point in our grid, we solve an auxiliary optimization problem where we maximize the expected revenue from online customers subject to constraints defined by the grid point. We use dynamic programming to compute an approximately optimal assortment for the auxiliary problem.

Literature Review: To the best of our knowledge, Dzyabura and Jagabathula (2017) are the first to study an omnichannel assortment optimization problem. They consider feature classes with several feature values per feature class. A product is created by combining one feature value per feature class. The retailer offers all the products that may result as combinations of feature values in his online store. We refer to their model as the list-of-features model. In Figure 1, the feature classes could be product types, product lines, and colours. The novelty is that they optimize over the sets of feature values to display, and recover the optimal assortment from the set of feature values. In the showroom setting, the revenue-maximizing assortment can be computed in polynomial-time when products have unit-revenue, and they present a FPTAS for arbitrary revenues. In the general setting, various assortments can demonstrate the same set of feature values but earn different offline expected revenue. They restrict the space of feasible assortments to the largest assortments described by sets of feature values, and give a FPTAS when the size of the feature classes is fixed. In contrast, the OCA problem is not NP-hard under our features tree model in the showroom setting, and our FPTAS does not depend on the size of feature classes in the general setting. Dzyabura



(a) Features tree

(DBC Strapless ballgown)	Pleated, plain	Beaded lace	Embroi-dered
Pink	√	√	√
Ivory	√	√	√
Asymmetric applique	√	√	√
Symmetric applique	√	√	√
Ruffled skirt	√	√	√
Split overskirt	√	√	√

(b) List-of-features

(DBC Strapless ballgown)	Pleated, plain	Beaded lace	Embroi-dered
Pink	√		
Ivory	√		
Asymmetric applique		√	
Symmetric applique		√	
Ruffled skirt		√	√
Split overskirt			√

(c) Generalized model

Figure 2: Comparison of the features tree model with the list-of-features model, and a generalization of the two models. Six strapless ballgowns are priced under $400 in David's Bridal Collection.

and Jagabathula (2017) provide empirical evidence that customers update preferences by features via a field experiment, where participants rank bags before and after observing similar bags.

Under the list-of-features model, products can share feature values over any feature classes without following a tree structure. However, a product exists for ***every*** combination of feature values, so that products must be fully customizable. In Figure 1, Dzyabura and Jagabathula (2017) would require that the same product lines be carried in both lipsticks and lip gloss, and that the same colours be available under each product line. In reality, the five product lines do not overlap in colours (M.A.C Cosmetics (2019*a*,*b*)). This is not due to naming conventions, as each product line offers a different number of colour options, and slight variations in shades of reds are important to customers of lip makeup products. Furthermore, lipsticks in the Retro Matte product line have a matte finish, and this product line cannot be offered under lip gloss because lip gloss have a glossy finish by definition.

To further stress the differences in the models, we consider wedding gowns in Figure 2. David's Bridal Collection offers six strapless ballgowns that are under $400 if we remove duplicates from plus and petite sizes (David's Bridal (2020)), which we organize into a features tree (Figure 2a). Gowns are typically in white and ivory, so that there is very little to evaluate from seeing these colours. We omit this detail unless it differentiates gowns with atypical colours (e.g. products 1 and 2). The list-of-features model requires that all 18 combinations of style and highlights be available (Figure 2b), so that the retailer must offer a pleated, plain gown with symmetric appliques. This is an incompatible combination because appliques are made from lace and beading, and a gown with appliques must have some beaded lace. Customization is also limited, because designers may refuse to construct a gown with an unattractive combination of features. The retailer cannot offer a pink, embroidered gown unless a designer is willing to produce the gown. Customers may also find that a certain highlight works better with one style than another. The features tree model permits customers' evaluation of a lower-level feature to be conditional on a higher-level feature. Furthermore, the features tree model is flexible with respect to the products that are offered by the retailer, and it is suitable for a diverse assortment of products which differentiate along prominent features. The list-of-features model is flexible with respect to how products are related by features, but it assumes that the products can be fully customized and is suitable for comparing very similar products along precise features like size and colour. Neither model is a generalization of the other. Both models are special cases of the generalized model (Figure 2c), which allows products to share features arbitrarily and allows the retailer to offer only some of the products created from combinations of features. In Figure 2c, the retailer offers a ruffled skirt in two styles, rather than in one style (Figure 2a) or in all of the styles (Figure 2b). The two special cases contribute to a better understanding of the difficulty of the OCA problem under the generalized model.

There is evidence in the psychology literature that customers do not always compare products by considering them as lists of features (Goldstone (1994)). Market segmentation, recommender systems, and psychology literature use trees to categorize products, providing ample support for our model (Albadvi and Shahbazi (2009), Cho et al. (2002), Ziegler et al. (2004)). Another way to interpret the vertices on the features tree is to consider each vertex as a subset of products, or category, which all share the associated feature. Rosch et al. (1976) show that the tree structure is not arbitrary, and that the top, middle, and bottom levels of the tree represent superordinate, basic, and subordinate categories. Basic categories are abstract enough that the category names are commonly used to refer to the products, and specific enough to visualize a standard product within the category. Superordinate categories are not specific enough to be informative when describing a product, and subordinate categories are too specific when trying to describe a product quickly (Rosch et al. (1976), Murphy and Brownell (1985)). Markman and Wisniewski (1997) find that products in different basic categories have many alignable differences and are easy to compare. Products in different superordinate categories have many nonalignable differences and cannot be directly compared. In Figure 1, lipstick is a basic category, lip makeup is a superordinate category, and the Lustre line is a subordinate category. Lipsticks can be compared with lip gloss, but lip makeup is not compared with eye makeup. Whether a category is basic could depend on customer expertise and the products being categorized (Rosch et al. (1976)). Silhouettes (e.g. ballgown vs. mermaid) could be a basic categories in the superordinate category of wedding gowns. However, customers typically focus on gowns with a specific silhouette when they visit bridal salons, and silhouettes could form the superordinate categories in this application.

When a new product is introduced to the market, customers may categorize the product by the presence or absence of distinctive features, or by similarity in appearance with familiar products. The retailer can train customers to categorize the new product using either methods (Yamauchi and Markman (1998)): by explicitly categorizing the product via advertisement (Moreau et al. (2001)), or by teaching customers to look for distinctive features (Noseworthy and Goode (2011)).

We assume that the preference weight of an online product is modified only to the extent of the features that it shares with an in-store product, according to the structure of the features tree. Hence, customers do not make inferences across different categories. In Figure 1, a customer cannot evaluate the colour of any lip gloss when she sees a lipstick. Several studies support this assumption and multi-category inferences occur only under exceptional circumstances: when customers are confused about a product or when the retailer encourages multi-category inferences (Moreau et al. (2001), Noseworthy and Goode (2011), Gregan-Paxton et al. (2005), Murphy and Ross (2010)).

Other works have studied challenges in the omnichannel environment. Harsha et al. (2019) study the price optimization problem when prices affect the fraction of customers purchasing from each store. Gao and Su (2016*a*) study the profitability of the buy-online-pickup-in-store (BOPS) option, where customers strategically choose between shipping cost and the hassle of traveling to the store. Gao and Su (2016*b*) study a retailer who uses both channels to encourage customers to visit the physical store by reducing their risk of stock-out. The latter two papers consider a single product, but we integrate a choice model so that demand for each product depends on the assortment.

Empirical evidence supports the importance of studying an omnichannel retail environment. Warby Parker is a glasses retailer which used to sell exclusively online, and has a sampling program for customers to try glasses for five days. Bell et al. (2015) and Bell et al. (2017) find that online sales increased and returns decreased in cities where Warby Parker opened a showroom to let customers try out glasses before ordering online. Fornari et al. (2016) and Avery et al. (2012) study online retailers who opened physical stores, and find that sales increased in the long-run when customers have an additional channel to research products. In the reverse direction, customers like to purchase "high-touch" products in-store. By offering the BOPS option, Bell et al. (2014) find that the retailer benefits from higher store traffic and the opportunity to cross-sell products.

Our work is 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. In the d-level nested logit model, Li et al. (2015) use the categorization view of the tree to describe product features. The important difference is that their tree describes a customer's choice process, whereas our tree describes how a customer revises her 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 as she moves from the root to a leaf. In our model, a customer always considers the entire assortment available either online or offline, depending on her type.

The existence of offline and online customers is related to the mixture of MNL (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 preferences. In our model, online and offline customers consider different assortments, but they can have the same or different preference weights for products that are offered in both channels. Our FPTAS uses techniques from Désir et al. (2014)'s work on capacitated assortment optimization under MMNL, where each product has a capacity requirement and the assortment's capacity cannot exceed a budget. For each customer type, they create a geometric grid on the numerator and denominator of the expected revenue function. A point on the grid lower-bounds the expected revenue from each customer type. They give a dynamic program that finds the minimum capacity assortment which satisfies the constraints imposed by the grid, if such an assortment exists. We create a geometric grid on the numerator and denominator of the offline expected revenue, and we maximize the online expected revenue.

Organization: In Section 2, we describe the OCA problem under the features tree model, and explain how the preferences of online customers are updated based on the in-store assortment. In Section 3, we focus on the showroom setting with only online customers. We present a polynomial-time algorithm, and give conditions for a revenue-ordered or a reverse revenue-ordered display assortment to be optimal. In Section 4, we present the FPTAS for the general setting. To evaluate the practical performance of our FPTAS, we present an efficient method to upper-bound the optimal expected revenue in Section 5. We discuss extensions in Section 6. We allow the retailer to choose both the online and in-store assortments, consider more general product relationships, and limit the size of the in-store assortment. We provide numerical experiments in our last two sections. In particular, in Section 7, we test the modeling power of our features tree model when the ground-truth model allows products to arbitrarily share features. In Section 8, we assess the practical performance of our FPTAS. We conclude in Section 9. All omitted proofs are deferred to Online Appendix A.

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 $\mathcal{N} = \{1, \ldots, n\}$. 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 $\mathcal{S} \subseteq \mathcal{N}$ 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 among products. The vertices of the tree, denoted by $\mathcal{V}$, correspond to the features of the products. The path from a leaf to the root gives all the features that uniquely defines a product. Hence, each leaf corresponds to a product and the set of leaves in T is exactly $\mathcal{N}$. See Figures 1 and 2a for examples. To characterize the structure of the tree, we introduce notations to describe its parent-child relationships. If $k \in \mathcal{V}$ is not the root vertex, then we denote its parent by $p(k)$. Let $A(k)$ be the set of ancestors of vertex k and itself, so that $A(\text{root}) = \{\text{root}\}$ and $A(k) = A(p(k)) \cup \{k\}$ if

$k \neq$ root. We can interpret $A(i)$ as the set of features of product i. Let $L(k)$ be the set of leaves in the subtree rooted at vertex k, that is, $L(k) = \{i \in \mathcal{N} \mid k \in A(i)\}$. Then $L(k)$ is the set of products which share feature k. We say that feature k is displayed if any of the products in $L(k)$ are displayed in-store. For an in-store assortment $\mathcal{S}$, the set of features displayed to customers is $\cup_{j\in\mathcal{S}} A(j)$.

There are two types of customers: offline and online customers. A customer's type determines the assortment that 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 $\mathcal{S}$, and always associates a preference weight $\hat{v}_i > 0$ with product i. An online customer visits the physical store to observe the displayed products, but ultimately purchases from the full assortment $\mathcal{N}$ online. Her preferences depend on the features displayed in-store by assortment $\mathcal{S}$, and she associates preference weight $v_i(\mathcal{S})$ with product i. We first describe the update process, and then define $v_i(\mathcal{S})$.

An online customer has initial preference weight $w_i > 0$ for product i, which is her preference weight when she does not see any features of product i. Each vertex k is associated with a multiplier $\delta_k > 0$. If she sees feature k of product i, then her preference weight is updated by multiplying w_i with δ_k. We interpret $\delta_k > 1$ as the increase in preference weight from seeing a feature that is more appealing than suggested by its online description (a good feature). Similarly, $\delta_k < 1$ corresponds to a feature being less attractive (a bad feature), and $\delta_k = 1$ corresponds to no changes in opinion (an indifferent feature). When k represents an aggregate of sub-features, as in Figure 1, then δ_k is the overall change from all of the sub-features. Both w_i's and δ_k's are deterministic input parameters.

Without loss of generality, we may assume that our features tree is a binary tree, so that every non-leaf vertex k has a left child $\ell(k)$ and a right child $r(k)$. If a general features tree has a vertex k with more than two children, then we can convert the tree into a binary tree by introducing an auxiliary vertex k' with $\delta_{k'} = 1$. We can reduce the degree of vertex k by setting the auxiliary vertex k' as the parent of the second through last children of k, and making k' the new second child of k. Conversely, if vertex k has only one child, then we can contract the edge between k and its child. Repeated application of this process allows us to obtain a binary features tree.

Assumption 1. *The features tree T is a binary tree. Given n products, T has $2n-1$ vertices. Hence, we index the leaves by $\mathcal{N} = \{1,\ldots,n\}$ and non-leaf vertices by $\mathcal{V}\backslash\mathcal{N} = \{n+1,\ldots,2n-1\}$.*

To define $v_i(\mathcal{S})$, we assign binary variable x_k to $k \in \mathcal{V}$, so that $x_k = 1$ means that feature k is displayed in-store and $x_k = 0$ otherwise. Given an in-store assortment $\mathcal{S}$, its characteristic vector is $x \in \{0,1\}^{2n-1}$ such that $x_k = \mathbb{1}[k \in \cup_{j\in\mathcal{S}} A(j)]$, with $\mathbb{1}[\cdot]$ being the indicator function. Since $\cup_{j\in\mathcal{S}} A(j)$ is the set of displayed features, x_k takes the value of 1 if and only if k is a feature of a displayed product. Furthermore, if feature k is a leaf, then the retailer offers product k in-store whenever feature k is displayed. Since the first n indices of x reveals the in-store assortment and we can recover an assortment by taking $x_k = 1$ for $k \in \mathcal{N}$, we also refer to x as the assortment.

If feature k corresponds to a non-leaf vertex, then it is displayed if and only if at least one of the products in its subtree is available in-store. The corresponding leaf is in the subtree of either $\ell(k)$ or $r(k)$, which implies that at least one of features $\ell(k)$ or $r(k)$ is displayed. 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:

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

With a slight abuse of notation, we can write the online preference weight of product i as $v_i(\mathcal{S}) = v_i(x)$ for $x \in \mathcal{X}$, where $v_i(x) = w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}$.

The no-purchase option is the customer's ability to leave the store without making a purchase, and is available regardless of her type. This option does not have features, and its preference weight does not change regardless of the in-store assortment. The no-purchase option, also denoted as product 0, has preference weight $w_0 > 0$ for an online customer and $\hat{v}_0 > 0$ for an offline customer. An online customer purchases a product in $\mathcal{N}$ or chooses the no-purchase option. An offline customer purchases exclusively from the in-store assortment x or chooses the no-purchase option.

A customer's purchase probability for product i is proportional to the preference weight of product i in the assortment that she purchases from, using the structure in MNL. Given x, the purchase probabilities of product i for online and offline customers are respectively:

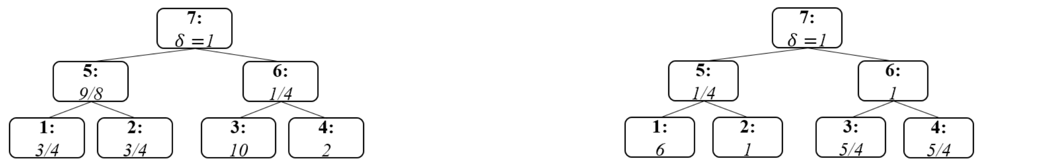
$$P_i^{ON}(x) = \frac{v_i(x)}{w_0 + \sum_{j=1}^n v_j(x)}, \qquad \text{and} \qquad P_i^{PHY}(x) = \frac{\hat{v}_i x_i}{\hat{v}_0 + \sum_{j=1}^n \hat{v}_j x_j}.$$

The online and offline expected revenue are denoted by $\Pi^{ON}(x)$ and $\Pi^{PHY}(x)$, where $\Pi^{ON}(x) = \sum_{i=1}^n \pi_i P_i^{ON}(x)$ and $\Pi^{PHY}(x) = \sum_{i=1}^n \pi_i P_i^{PHY}(x)$.

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 standard MNL. Hence we consider $q \in (0,1]$. If we display assortment x, then the expected revenue is $\Pi(x) = q \cdot \Pi^{ON}(x) + (1-q) \cdot \Pi^{PHY}(x)$. The OCA problem is to choose an assortment x that maximizes the total expected revenue, and can be formulated as the following optimization problem, where we expand out $\Pi^{ON}(x)$ and $\Pi^{PHY}(x)$:

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

We can interpret our model as an extension to MNL. Suppose online customers have a mean utility of $\ln w_i$ for product i if none of its features are displayed. A customer's utility for product i



(a) Product $4 = \arg\min_{i \in \mathcal{N}} \prod_{k \in A(i)} \delta_k$ (b) Product $1 = \arg\max_{i \in \mathcal{N}} \prod_{k \in A(i)} \delta_k$

Figure 3: Two features trees with $\pi_i = w_i = 1$ for all $i \in \mathcal{N}$, with $\mathcal{N} = \{1,2,3,4\}$ and optimal assortment $\{3,4\}$. Bold numbers denote feature k, and italicized numbers denote multipliers δ_k.

is its mean utility plus a noise ϵ_i, which is generated by an independent, standard Gumbel random variable with mean 0. If feature k is displayed, then the mean utility for product i changes additively by $\ln \delta_k$: $\ln w_i + \ln \delta_k = \ln(w_i \cdot \delta_k)$, and this translates to a multiplicative update to product i's preference weights. Hence, δ_k reflects the change in opinion of the general population from seeing a feature versus reading about it online, rather than the resolution of uncertainty for an individual.

3. Showroom Setting

We begin with the showroom setting where $q = 1$ in Problem (1). We briefly explain why the structural results of Dzyabura and Jagabathula (2017) do not apply to our model. Then, we present an algorithm which computes the optimal showroom assortment in polynomial runtime.

3.1. Challenges of Showroom Setting with Unit-Revenues

Under Dzyabura and Jagabathula (2017)'s list-of-features model, the optimal assortment can be computed efficiently only when $\pi_i = 1$ for all $i \in \mathcal{N}$ in the showroom setting. Under these conditions, maximizing expected revenue is equivalent to maximizing the sum of the preference weights over all products. Recall that their problem reduces to deciding which feature values to display. The optimal assortment is recovered by displaying all the products that may be constructed from the optimal set of feature values. If the optimal assortment is not the empty set, then all good ($\delta_k > 1$) and indifferent ($\delta_k = 1$) feature values are displayed. If a feature class has only bad feature values ($\delta_k < 1$), then the feature value with the largest δ_k is displayed in order to minimize the discount on products' preference weights. Hence, whether each feature value is displayed can be decided independently. In contrast, the features tree model limits the flexibility in which the retailer can display features, and it may be suboptimal to display good features or hide bad features.

In the problem instance given by Figure 3a, the optimal assortment is the set of products $\{3,4\}$, and the retailer displays features $\{3,4,6,7\}$. If we blindly apply Dzyabura and Jagabathula (2017)'s result, then the retailer should display features $\{3,4,5,7\}$, which is infeasible. The retailer must display products 1 or 2 in order to display feature 5. The increase in preference weights from displaying the good feature 5 does not compensate the discount of seeing either of the bad features

1 or 2. In contrast, feature 6 is displayed because the increase in preference weights from its children features makes up for its discount. We consider a realistic example based on reviews of the lipsticks in Figure 1 (Adrienne Söndag (2019)). Feature 5 could be a line of moisturizing lipsticks with unappealing colours, and feature 6 could be a line of dry lipsticks with very appealing colours. The retailer may not have a product with both of the good features.

When $\pi_i = 1$ for all $i \in \mathcal{N}$, the optimal set of feature values can be computed greedily under the list-of-features model. In contrast, the problem instances given by Figure 3 demonstrate that we cannot compute the optimal assortment by greedily adding features or products. In Figure 3a, the optimal assortment is not $\mathcal{N}$, but it includes the worst feature ($\delta_6 = 1/4$) and the product with the most decrease in preference weight (product 4). In Figure 3b, the optimal assortment is non-empty, but it excludes the best feature ($\delta_1 = 6$) and the product with the most increase in preference weight (product 1). Surprisingly, if products have arbitrary revenues, then assortment optimization under Dzyabura and Jagabathula (2017)'s model is NP-hard, whereas we present an efficient algorithm to compute the optimal showroom assortment in the next subsection.

3.2. Computing the Optimal Showroom Assortment

We return to studying the showroom setting with arbitrary revenues $\pi_i \geq 0$, so that the goal is to optimize the retailer's expected revenue. 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, we can parameterize the objective function and find a fixed point to the parametric problem. Specifically, suppose there exists an assortment $x \in \mathcal{X}$ that generates expected revenue greater or equal to γ, so that $\sum_{i=1}^n \pi_i w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k} / (w_0 + \sum_{i=1}^n w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}) \geq \gamma$. 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 assortments in $\mathcal{X}$, we obtain the parametric problem as a function of γ:

$$f(\gamma) = \max_{x \in \mathcal{X}} \quad \sum_{i=1}^n (\pi_i - \gamma) w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}. \tag{2}$$

Claim 1. *Given $f(\gamma)$ as defined above, let γ^* be the optimal expected revenue of the showroom setting. Then: 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 $\gamma = \gamma^*$.*

To find γ^*, we need to efficiently solve Problem (2) for any γ and search for γ^* over possible values of γ. If x is a feasible solution to Problem (2), then either $x = \vec{0}$ or there exists $i \in \mathcal{N}$ such that $x_i = 1$. The latter case is equivalent to $x_{\text{root}} = 1$. For any fixed γ, we give a dynamic program that computes the optimal non-zero solution to Problem (2) and compares the result to $x = \vec{0}$.

Let $V_\gamma(k)$ be the maximum contribution to the objective function of Problem (2) from all leaves in the subtree rooted at vertex k, given that feature k is displayed 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 $k' \in A(k)$. Moreover, since we only consider the contribution from the products in $L(k)$, we can focus on feasibility of the constraints in $\mathcal{X}$ related to vertices in the subtree of k and use $\mathcal{X}_k$ to denote $\mathcal{X} \cap \{x \mid x_{k'} = 1 \, \forall k' \in A(k)\}$. By this definition, $V_\gamma(k)$ can be written as:

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

If k is the root, then $V_\gamma(\text{root})$ optimizes Problem (2) 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 parametric problem can be rewritten as $f(\gamma) = \max\{V_\gamma(\text{root}), \sum_{i=1}^n (\pi_i - \gamma) w_i\}$.

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. If k is a leaf, then feasibility to $\mathcal{X}_k$ implies that $V_\gamma(k) = (\pi_k - \gamma) w_k \cdot \prod_{k' \in A(k)} \delta_{k'}$. Otherwise, $V_\gamma(k)$ is related to the values at its children: $V_\gamma(\ell(k))$ and $V_\gamma(r(k))$. Since the value of $V_\gamma(k)$ assumes that $x_k = 1$, we consider three cases: *i)* $x_{\ell(k)} = x_{r(k)} = 1$, *ii)* $x_{\ell(k)} = 1, x_{r(k)} = 0$, and *iii)* $x_{\ell(k)} = 0, x_{r(k)} = 1$. The objective function and constraints can be separated over 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 subtree of the child feature that is not displayed. Consider case (*ii*) where $x_{r(k)} = 0$. Feature $r(k)$ and its descendants are not displayed in-store, but feature k and all its ancestors are displayed. Hence, the preference weight of product i in $L(r(k))$ is $w_i \cdot \prod_{k' \in A(k)} \delta_{k'}$. The same analysis holds for $i \in L(\ell(k))$ in case (*iii*). To simplify notation, let Δ_k denote the product of all $\delta_{k'}$ for $k' \in A(k)$, with the convention $\Delta_{p(\text{root})} = 1$. Then $\Delta_k = \prod_{k' \in A(k)} \delta_{k'} = \Delta_{p(k)} \cdot \delta_k$ for all k.

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

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

The base cases are $k \in \mathcal{N}$ 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 runtime of finding γ^* such that $f(\gamma^*) = w_0 \gamma^*$. The parametric problem $f(\gamma)$ is monotone decreasing in γ, and one way to find γ^* is via bisection search between upper- and lower-bounds on the online expected revenue. 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 of x, then the fixed point can be found in $O(n^2 \log^2 n)$ iterations of Newton's method because $\mathcal{X} \subseteq \{0,1\}^{2n-1}$.

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 $x \in \mathcal{X}$, so that $v_i(x) = w_i \cdot \left(1 + \sum_{k \in A(i)} (\Delta_k - \Delta_{p(k)}) \cdot x_k\right)$. Hence, we can write the online expected profit as:*

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

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. We need to compute $f(\gamma)$ and find γ^* such that $f(\gamma^*) = w_0 \gamma^*$. We can compute $f(\gamma)$ in $O(n)$ operations for any 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 $O(n^3 \log^2 n)$. $\square$

3.3. Managerial Insights

To discuss managerial insights for the showroom setting, we consider reasonable restrictions on the values of the δ_k's. We revert to the more natural form of a general features tree rather than its binary tree representation. We need to consider each product's revenue before deciding whether we should increase its preference weight, because we only want to increase the attractiveness of the expensive products. By imposing certain structures on the features tree, a revenue-ordered display assortment can be optimal. We define a revenue-ordered tree as follows:

Definition 1. A revenue-ordered features tree (ROFT) is a features tree such that for all non-leaf vertices k, and for all $i \in L(\ell(k))$ and $j \in L(r(k))$, we have $\pi_i > \pi_j$.

Pictorially, this is a features tree where the products on the left have higher revenues than products on the right. This definition is a technical way of saying that products which are closer in revenue have more features in common. Hence, the structure of a ROFT is actually quite natural.

Suppose a retailer is new to the market and his products are under-rated by his customers ($\delta_k \geq 1$ for all k). Intuitively, he should display the most expensive products so that they receive the largest improvement in preference weights. However, if his products look attractive online but are sub-par in reality ($\delta_k \leq 1$ for all k), then he should display the least expensive products so that they receive the largest discounts in preference weights. We prove these statements for a ROFT.

Proposition 1. *Suppose the features tree is a ROFT and the retailer operates in the showroom setting. Let S^* denote the optimal display assortment. If $\delta_k \geq 1$ for all $k \in \mathcal{V}$, then S^* is either revenue-ordered and has the form $\{1, 2, \ldots, i^*\}$ for some i^*, or $S^* = \emptyset$. If $\delta_k \leq 1$ for all $k \in \mathcal{V}$, then S^* is either reverse revenue-ordered and has the form $\{i^*, i^*+1, \ldots, n\}$ for some i^*, or $S^* = \emptyset$.*

To extend Proposition 1, suppose that each product either has no bad features or no good features. Let $\mathcal{N}_+ = \{i \in \mathcal{N} \mid \delta_k \geq 1 \, \forall k \in A(i)\}$ and $\mathcal{N}_- = \{i \in \mathcal{N} \mid \delta_k \leq 1 \, \forall k \in A(i), \, \exists k \in A(i) \text{ s.t. } \delta_k < 1\}$ partition $\mathcal{N}$. Observe that products $i \in \mathcal{N}_+$ and $j \in \mathcal{N}_-$ only share indifferent features (i.e. $\delta_k = 1$ for all $k \in A(i) \cap A(j)$). A high-level feature like the brand may have no impact on preferences if quality is inconsistent across the brand's products. We show that it is optimal to display a revenue-ordered subset of $\mathcal{N}_+$ with a reverse revenue-ordered subset of $\mathcal{N}_-$.

Theorem 2. *Suppose the features tree is a ROFT and the retailer operates in the showroom setting. If the full assortment $\mathcal{N}$ can be partitioned into $\mathcal{N}_+$ and $\mathcal{N}_-$, then the optimal assortment S^* is either $\emptyset$, or there exists i^* such that $S^* = \{i \in \mathcal{N}_+ \mid i \leq i^*\} \cup \{i \in \mathcal{N}_- \mid i > i^*\}$.*

As a special case, suppose our features tree is not a ROFT, but $\delta_k = 1$ for all $k \notin \mathcal{N}$. Each product's preference weight only changes when it is seen by customers. We can redraw the features tree with only two levels, so that the leaf vertices are children of the root vertex. A two-level tree can always be drawn as a ROFT, and we can apply Theorem 2 to compute the optimal display assortment.

On the other hand, sometimes the high-level features can have large impact on customers' preferences, whereas low-level features are details that only have small impact. Mathematically, let $k_1, \ldots, k_F$ be F features such that $L(k_1), \ldots, L(k_F)$ partition the products in $\mathcal{N}$. Let $\mathcal{V}^{\text{low}} = \mathcal{V} \setminus \left(\cup_{f=1}^{F} A(k_f)\right)$ denote the set of low-level features. If δ_k are close to 1 for all $k \in \mathcal{V}^{\text{low}}$, then we can consider a contracted problem where we decide to display one or zero product from each $L(k_f)$. In particular, we can contract the the vertices in the subtree rooted at k_f into a super-product with revenue $\sum_{i \in L(k_f)} \pi_i w_i / \sum_{i \in L(k_f)} w_i$, initial preference weight $\sum_{i \in L(k_f)} w_i$, and feature multiplier δ_{k_f}. If x^o is the optimal solution to the contracted problem, then we can recover a display assortment x^ϵ via the following procedure. If $x^o_{k_f} = 0$, then set $x^\epsilon_i = 0$ for all $i \in L(k_f)$. If $x^o_{k_f} = 1$, then randomly select a product i from $L(k_f)$ to include in the display assortment and set $x^\epsilon_i = 1$.

Proposition 2. *Let $\epsilon \geq 0$. Suppose the retailer operates in the showroom setting and $\delta_k \in [1 - \frac{\epsilon}{4}, 1 + \frac{\epsilon}{4}]$ for all $k \in \mathcal{V}^{\text{low}}$. Let $\bar{L} = \max_{\substack{k_f : f = 1, \ldots, F \\ i \in L(k_f)}} |A(i) \setminus A(k_f)|$. If an assortment x^ϵ is recovered from the optimal solution x^o of the contracted problem, as described above, then $\Pi(x^\epsilon) \geq (1 - \bar{L}\epsilon)\Pi^*$.*

Proposition 2 implies that the retailer does not need to be too concerned about the exact product being displayed from each $L(k_f)$, because low-level features that have small impact can be omitted from the model with a small loss in expected revenue. Algorithmically, we can compute a display assortment with reasonable performance in $O(F^3)$ runtime.

The usefulness of Proposition 2 depends on the shape of the features tree. In reality, we expect that the features tree is shallow because the depth of the tree corresponds to the number of levels that customers use to categorize products. This could be reflected by the number of filters that customers can use in the online store to find their products. For example, David's Bridal has filters for price, size, silhouette, sleeves, length, colour, neckline, style, and brand ($\bar{L} \le \max_{i\in\mathcal{N}} |A(i)| \le 9$). Prices do not change customers' preferences regardless of whether this feature is seen in-store or online. To obtain the perfect fit, customers would order their size based on body measurements and have the dress altered and tailored. Hence, these two filters are irrelevant for the features tree. Filters that are used to differentiate customer segments with disjoint consideration sets can be removed and a separate tree can be constructed for each customer segment. The resulting features tree would be shallow and the values of $\bar{L}, F$ should be quite small relative to the value of n.

4. General Setting

When $q \in (0,1)$, the OCA problem under the features tree model is NP-hard. We prove this in the next proposition. As such, we present a FPTAS to compute an assortment which guarantees $(1-\epsilon)$-fraction of the optimal expected revenue in Subsection 4.1. Our FPTAS requires us to solve a parametric problem, which is done via dynamic programming in Subsection 4.2.

Proposition 3. *The general setting of the OCA problem under the features tree model, where $q \in (0,1)$, is NP-hard.*

4.1. FPTAS for General Setting

Instead of optimizing over the sum of two fractions in Problem (1), we 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 in-store assortment, and let $R^* = \sum_{i=1}^n \pi_i \hat{v}_i x_i^*$ and $U^* = \hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i^*$. Denote the optimal expected revenue as $\Pi^* = \Pi(x^*)$.

Consider the problem of optimizing the online expected revenue, subject to constraints on the numerator and denominator of the offline expected revenue based on inputs $(R,U) \ge (0, \hat{v}_0)$:

$$g(R,U) = \max_{x\in\mathcal{X}} \left\{ \frac{\sum_{i=1}^n \pi_i w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i=1}^n w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k}} \;\middle|\; \sum_{i=1}^n \pi_i \hat{v}_i x_i \ge R,\ \hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i \le U \right\}. \tag{3}$$

An optimal solution x' to Problem (3) at (R^*, U^*) earns offline expected revenue of $\Pi^{PHY}(x') \ge R^*/U^*$ because x' satisfies the two constraints above. Furthermore, x' earns online expected revenue of $\Pi^{ON}(x') \ge \Pi^{ON}(x^*)$ because x' and x^* are both feasible solutions. Hence $\Pi(x') \ge \Pi^{ON}(x^*) + R^*/U^* = \Pi(x^*)$, and solving Problem (3) at (R^*, U^*) gives us the optimal in-store assortment.

We do not know R^* or U^*, and we do not have an algorithm to solve Problem (3) 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. Let $\underline{R} = \min_{i\in\mathcal{N}} \pi_i \hat{v}_i$ and $\overline{R} = \max_{i\in\mathcal{N}} \pi_i \hat{v}_i$. Similarly, let $\underline{U} = \min_{i\in\mathcal{N}\cup\{0\}} \hat{v}_i$ and $\overline{U} = \max_{i\in\mathcal{N}\cup\{0\}} \hat{v}_i$. When there is at least one product offered in-store, then $\underline{R}$ and $n\overline{R}$ are the lower- and upper-bounds on the numerator of $\Pi^{PHY}(\cdot)$. Similarly, $\underline{U}$ and $(n+1)\overline{U}$ are the lower- and upper-bounds on the denominator of $\Pi^{PHY}(\cdot)$. Given $\epsilon > 0$, our grid is constructed as $\mathcal{K}_\epsilon = \{(0,\hat{v}_0)\} \cup (\mathcal{K}^R_\epsilon \times \mathcal{K}^U_\epsilon)$, where:

$$\mathcal{K}^R_\epsilon = \{\underline{R}\cdot(1+\epsilon)^d \mid \underline{R}\cdot(1+\epsilon)^d \le (1+\epsilon)\cdot n\overline{R},\ d\in\mathbb{Z}_+\}, \text{ and}$$
$$\mathcal{K}^U_\epsilon = \{\underline{U}\cdot(1+\epsilon)^d \mid \underline{U}\cdot(1+\epsilon)^d \le (1+\epsilon)\cdot (n+1)\overline{U},\ d\in\mathbb{Z}_+\}.$$

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 (3) is feasible and $g(R,U)$ is well-defined.

Proposition 4. *Consider an instance of the OCA problem under the features tree model with optimal expected revenue* Π^* *and* $\epsilon < 1/6$. *For any* $(R,U)\in\mathcal{K}_{\text{feas}}$, *suppose we can compute a solution* $x^{R,U}\in\mathcal{X}$ *that achieves an online expected revenue* $\Pi^{ON}(x^{R,U}) \ge g(R,U)$ *and satisfies:*

$$\sum_{i=1}^n \pi_i\hat{v}_i x_i^{R,U} \ge (1-2\epsilon)R, \quad \text{and} \quad \hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i^{R,U} \le (1+2\epsilon)U.$$

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

The only feasible solution to Problem (3) at $(0,\hat{v}_0)$ is $x=\vec{0}$, and this solution satisfies the assumptions of Proposition 4. The challenge is to efficiently compute $x^{R,U}$ satisfying the assumptions of Proposition 4 for each $(R,U)\in\mathcal{K}_{\text{feas}}\backslash\{(0,\hat{v}_0)\}$. If we parameterize the objective function of Problem (3), we get Problem (2) from the showroom setting, subject to two knapsack-like constraints. Our goal is to incorporate the knapsack constraints into the dynamic program from Subsection 3.2, by considering how the products in the left and right subtrees contribute to these constraints. To ensure that the state space and decision space are not too large, we round the parameters of these constraints to integers of size at most $O(n/\epsilon)$:

$$\tilde{\pi}^R_i = \left\lfloor \frac{\pi_i\hat{v}_i}{\epsilon R/n} \right\rfloor \ \forall i\in\mathcal{N}, \quad \text{and} \quad \tilde{v}^U_i = \left\lceil \frac{\hat{v}_i}{\epsilon U/(n+1)} \right\rceil \ \forall i\in\mathcal{N}\cup\{0\}.$$

We define $Y_1 = \lfloor \frac{n}{\epsilon} \rfloor - n$ and $Y_2 = \lceil \frac{n+1}{\epsilon} \rceil + (n+1)$ as the polynomial-sized bounds on the rounded numerator and denominator of the offline expected revenue. Finally, we parameterize the objective function of Problem (3) and use the rounded constraints to obtain Problem (4):

$$\tilde{f}(\gamma,R,U) = \max_{x\in\mathcal{X}} \left\{ \sum_{i=1}^n (\pi_i-\gamma)w_i \cdot \prod_{k\in A(i)} \delta_k^{x_k} \ \middle|\ \sum_{i=1}^n \tilde{\pi}^R_i x_i \ge Y_1,\ \tilde{v}^U_0 + \sum_{i=1}^n \tilde{v}^U_i x_i \le Y_2 \right\}. \tag{4}$$

Following the strategy of Subsection 3.2, we need to solve Problem (4) and find $\gamma^{R,U}$ such that $\tilde{f}(\gamma^{R,U}, R, U) = w_0 \gamma^{R,U}$. In the next subsection, we solve Problem (4) via dynamic programming. Meanwhile, since the parameters on the knapsack constraints have been rounded, a feasible assortment to Problem (4) does not guarantee an offline expected revenue of R/U. Fortunately, the next lemma and corollary prove that such an assortment still guarantees an offline expected revenue of $(1-2\epsilon)R/(1+2\epsilon)U$. Furthermore, at the fixed point $\gamma^{R,U}$, the online expected revenue of our assortment is lower-bounded by $g(R,U)$, and satisfies the conditions of Proposition 4.

LEMMA 2. *Suppose x is a feasible solution to Problem (3) at $(R,U) \in \mathcal{K}_{\text{feas}}$. Then x is a feasible solution to Problem (4) at (γ, R, U) for any γ. Furthermore, any feasible x to Problem (4) satisfies $\sum_{i=1}^n \pi_i \hat{v}_i x_i \geq (1-2\epsilon)R$ and $\hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i \leq (1+2\epsilon)U$.*

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 (4) 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 $\hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i^{R,U} \leq (1+2\epsilon)U$, and ensures that $\Pi^{ON}(x^{R,U}) \geq g(R,U)$.*

In summary, our FPTAS creates a geometric grid $\mathcal{K}_\epsilon$ on the numerator and denominator of the offline expected revenue. A feasible solution to Problem (3) at $(R,U) \in \mathcal{K}_\epsilon$ guarantees an offline expected revenue greater or equal to R/U. Problem (3) is hard to solve because of the knapsack constraints, but our FPTAS only requires a perturbed solution which satisfies the conditions of Proposition 4. By Lemma 2 and Corollary 1, we can parameterize a rounded version of Problem (3) to arrive at Problem (4). The optimal solution $x^{R,U}$ to the fixed point $\gamma^{R,U}$ of Problem (4) satisfies the conditions of Proposition 4. The parameters in the constraints of Problem (4) are integers of size $O(n/\epsilon)$, so we use dynamic programming to find its optimal solution in the next subsection.

4.2. Solving the Parametric Problem at a Grid-point

As explained previously, we focus on solving Problem (4) at (γ, R, U) such that $(R,U) \in \mathcal{K}_\epsilon \backslash \{(0, \hat{v}_0)\}$. If $(R,U) \neq (0, \hat{v}_0)$ and $\epsilon < 1/6$, then we have $Y_1 > 0$ and $x = \vec{0}$ is infeasible to Problem (4). The feasible region of Problem (4) is contained in $\mathcal{X} \backslash \{\vec{0}\} = \mathcal{X}_{\text{root}}$. We proceed to solve Problem (4) for $(R,U) \in \mathcal{K}_\epsilon \backslash \{(0, \hat{v}_0)\}$ via dynamic programming.

Like the showroom setting, our value function considers the maximum contribution from the leaves in the subtree rooted at vertex k, given that feature k is displayed. We need to include two more states in our value function to recognize the impact of $L(k)$ to the two knapsack constraints. Let $\tilde{V}_\gamma^{R,U}(k, y_1, y_2)$ be the maximum contribution from products in $L(k)$ when feature k is displayed, subject to the constraints of $\mathcal{X}_k$. Moreover, we require that the products in $L(k)$ contribute *i)* at least y_1 to the first constraint of Problem (4) and *ii)* at most y_2 to the second constraint:

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

The value of our parametric problem at γ and $(R,U) \neq (0, \hat{v}_0)$ 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. 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, $\tilde{v}_0^U$ is a constant that can be moved to the right side of the second constraint in Problem (4). Second, when we compute $f(\gamma)$ in Problem (2) of Subsection 3.2, we compare the result of the dynamic program at $V_\gamma(\text{root})$ to the objective value at $x = \vec{0}$ because the feasible region $\mathcal{X}_{\text{root}}$ excludes the empty assortment. As stated previously, the feasible region of Problem (4) 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. We consider three cases: *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 consider all splits of y_1, y_2 across the trees rooted at $\ell(k)$ and $r(k)$. In the second case, the products in $L(r(k))$ do not contribute to the knapsack 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 $y_1 \in \{0, \dots, Y_1\}$ and $y_2 \in \{0, \dots, Y_2 - \tilde{v}_0^U\}$:

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

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

To summarize this section, our FPTAS for finding assortment $\bar{x}$ such that $\Pi(\bar{x}) \geq (1-\epsilon)\Pi^*$ is presented in Algorithm 1. The set $\tilde{\mathcal{K}}_{\text{feas}}$ on the second line contains all (R,U) such that Problem (4) is feasible at $(0,R,U)$. By Lemma 2, feasibility of Problem (3) implies feasibility of Problem (4), 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 solution $\bar{x}$ satisfies $\Pi(\bar{x}) \geq \max_{(R,U) \in \mathcal{K}_{\text{feas}}} \Pi(x^{R,U}) \geq (1-\epsilon)\Pi^*$, where the first inequality is due to $\mathcal{K}_{\text{feas}} \subseteq \tilde{\mathcal{K}}_{\text{feas}}$. The second inequality is due to Proposition 4 and $\epsilon' = \epsilon/6$.

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 $O\left(\frac{n^7 \log^2 n \log(n\overline{R}/\underline{R}) \cdot \log((n+1)\overline{U}/\underline{U})}{\epsilon^6}\right)$ operations.*

Proof. 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 $(R,U) \in \mathcal{K}_\epsilon$. If Problem (4) is feasible at $(0,R,U)$, then Newton's method finds $\gamma^{R,U}$ such that $\tilde{f}(\gamma^{R,U}, R, U) = w_0 \gamma^{R,U}$ in $O(n^2 \log^2 n)$ iterations. In total, we solve our dynamic program $O\left(\frac{n^2 \log^2 n \log(n\overline{R}/\underline{R}) \cdot \log((n+1)\overline{U}/\underline{U})}{\epsilon^2}\right)$ times.

We compute $\tilde{V}_\gamma^{R,U}(k, y_1, y_2)$ for $O(n)$ vertices and $O(n^2/\epsilon^2)$ pairs of (y_1, y_2) per vertex. For each state, we consider $O(n^2/\epsilon^2)$ ways to split (y_1, y_2) over the children of k. There are $O(n^5/\epsilon^4)$ operations to obtain $\tilde{V}_\gamma^{R,U}(\text{root}, Y_1, Y_2 - \tilde{v}_0^U)$. Our total runtime is $O\left(\frac{n^7 \log^2 n \log(n\overline{R}/\underline{R}) \cdot \log((n+1)\overline{U}/\underline{U})}{\epsilon^6}\right)$. $\square$

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

```
Input  : Instance of OCA problem under the features tree model with desired accuracy $\epsilon$
Output: Assortment $\bar{x}$ such that $\Pi(\bar{x}) \geq (1-\epsilon)\Pi^*$
Set $\epsilon' \leftarrow \epsilon/6$ and construct $\mathcal{K}_\epsilon$ using $\epsilon'$ ;
Initialize grid $\tilde{\mathcal{K}}_{\text{feas}} \leftarrow \{(0, \hat{v}_0)\}$ and set $x^{0,\hat{v}_0} \leftarrow \vec{0}$ ;
for $(R,U) \in \mathcal{K}_\epsilon \backslash \{(0, \hat{v}_0)\}$ do
    Set $\gamma \leftarrow 0$ ;
    Solve Problem (4) for optimal $x$, if feasible ;
    if Problem (4) 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 (4) for optimal $x$ and value $\tilde{f}(\gamma, R, U)$ ;
        end
        Set $x^{R,U} \leftarrow x$;
end
Return $\bar{x} = \arg\max_{(R,U) \in \tilde{\mathcal{K}}_{\text{feas}}} \Pi(x^{R,U})$.
```

In Online Appendix B, we reduce the runtime by a factor of $O(n^2/\epsilon^2)$ by solving Problem (4) as a constrained longest path problem on a directed acyclic graph. This allows us to 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 in Theorem 4.

5. Upper-Bound on Optimal Expected Revenue

To evaluate the practical performance of our FPTAS, it is useful to efficiently compute an upper-bound on the optimal expected revenue. We can compare the expected revenue from the assortment obtained by our FPTAS with the upper-bound to get a sense of the optimality gap of our solution.

To construct our upper-bound, let $\mathcal{X}^{\text{LP}}$ denote the relaxation of $\mathcal{X}$ where we replace $x \in \{0,1\}^{2n-1}$ with $x \in [0,1]^{2n-1}$. Using our grid $\mathcal{K}_\epsilon$, let $g^{\text{LP}}(R,U)$ denote the optimal objective value when we relax the integrality constraint of Problem (3) at (R,U):

$$g^{\text{LP}}(R,U) = \max_{x \in \mathcal{X}^{\text{LP}}} \left\{ \frac{\sum_{i=1}^n \pi_i w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i=1}^n w_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}} \;\middle|\; \sum_{i=1}^n \pi_i \hat{v}_i x_i \geq R,\ \hat{v}_0 + \sum_{i=1}^n \hat{v}_i x_i \leq U \right\}. \qquad (5)$$

We can use $g^{\text{LP}}(R,U)$ to obtain an upper-bound on the optimal total expected revenue Π^*. Recall that $\mathcal{K}_{\text{feas}} \subseteq \mathcal{K}_\epsilon$ is the set of points on our geometric grid such that Problem (3) is feasible. Since Problem (5) is a relaxation of Problem (3), Problem (5) is also feasible at $(R,U) \in \mathcal{K}_{\text{feas}}$.

Proposition 5. *For* $(R,U) \in \mathcal{K}_{\text{feas}}$*, define* $\Pi^{\text{LP}}(R,U) = q \cdot g^{\text{LP}}(R,U) + (1-q) \cdot \frac{(1+\epsilon)^2 R}{U}$. *Then* $\max_{(R,U) \in \mathcal{K}_{\text{feas}}} \Pi^{\text{LP}}(R,U) \geq \Pi^*$.

To compute $g^{\text{LP}}(R,U)$, we consider the parametric form of Problem (5) with optimal objective value $f^{\text{LP}}(\gamma,R,U)$. We can also apply Lemma 1 so that the objective function of the parametric problem is linear in x. The parametric problem with optimal objective value $f^{\text{LP}}(\gamma,R,U)$ is:

$$f^{\text{LP}}(\gamma,R,U)= \tag{6}$$
$$\max_{x\in\mathcal{X}^{\text{LP}}}\left\{\sum_{i=1}^{n}(\pi_i-\gamma)w_i+\sum_{k=1}^{2n-1}\left(\sum_{i\in L(k)}(\pi_i-\gamma)\,w_i\right)\cdot\left(\Delta_k-\Delta_{p(k)}\right)\cdot x_k \;\middle|\; \sum_{i=1}^{n}\pi_i\hat{v}_i x_i\ge R,\ \hat{v}_0+\sum_{i=1}^{n}\hat{v}_i x_i\le U\right\}.$$

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 (6) is a linear program in x, so we can write its dual linear program. 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 constraints of Problem (6). 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:

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

where

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

By strong duality, the dual linear program has optimal objective value $f^{\text{LP}}(\gamma,R,U)$ if Problem (6) is feasible. Since the dual is linear in γ, we 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 a constraint to set the objective value of the dual to $w_0\gamma$. Finally, we scale the objective function by $1/w_0$ to solve for the desired γ, and obtain:

$$\min_{(y,z,u,\beta)\in\mathcal{Y}}\left\{\gamma\;\middle|\;\left(w_0+\sum_{i=1}^{n}w_i\right)\gamma-\sum_{k=1}^{2n-1}u_k+R\beta_1-(U-\hat{v}_0)\,\beta_2=\sum_{i=1}^{n}\pi_i w_i,\ \gamma\ge 0\right\}. \tag{7}$$

Hence $g^{\text{LP}}(R,U)$ can be computed by solving Problem (7) for all $(R,U)\in\mathcal{K}_{\text{feas}}$. The constraint $\gamma\ge 0$ does not modify the feasible region when $(R,U)\in\mathcal{K}_{\text{feas}}$, because $g^{\text{LP}}(R,U)\ge g(R,U)>0$.

We summarize the 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 (7) is feasible and returns a positive value. Algorithm 2 returns an upper-bound $\bar{\Pi}^{\text{LP}}$ such that $\bar{\Pi}^{\text{LP}}\ge\max_{(R,U)\in\mathcal{K}_{\text{feas}}}\Pi^{\text{LP}}(R,U)\ge\Pi^*$, where the first inequality is due to $\mathcal{K}_{\text{feas}}\subseteq\mathcal{K}_{\text{LP}}$ and the second inequality is due to Proposition 5.

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

```
Input  : Instance of OCA problem 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 $\mathcal{K}_{\text{LP}} \leftarrow \{(0,\hat{v}_0)\}$ and 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 (7) at $(R,U)$ for $\gamma$ ;
    if Problem (7) 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

We consider three extensions. First, we allow the retailer to choose both his online assortment and his in-store assortment in the showroom setting. Second, we extend the features tree model so that the leaves of the features tree represent subsets of similar products which are arbitrarily related by features. Third, we consider a cardinality constraint for the physical store in the general setting.

6.1. Related Online and In-store Assortments

Suppose the physical store displays a subset of the products from the online store, and we make assortment decisions in both channels. Many retailers encourage customers to order online if their desired product is unavailable in-store. We discuss this extension with respect to the showroom setting, but the techniques can be applied to the dynamic program in Subsection 4.2 for the general setting. Let $\mathcal{S}^{ON}$ and $\mathcal{S}^{PHY}$ denote the online and in-store assortments respectively. We introduce binary variables y_i for $i \in \mathcal{N}$, where $y_i = 1$ if product i is offered online and 0 otherwise. Since $\mathcal{S}^{PHY} \subseteq \mathcal{S}^{ON}$, we require $y_i \geq x_i$. The mathematical program for this extension is:

$$\max_{\substack{x \in \mathcal{X} \\ y \in \{0,1\}^n}} \left\{ \frac{\sum_{i=1}^n \pi_i w_i y_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i=1}^n w_i y_i \cdot \prod_{k \in A(i)} \delta_k^{x_k}} \,\middle|\, x_i \leq y_i \ \forall i \in \mathcal{N} \right\}. \tag{8}$$

Let $(\cdot)^+$ denote $\max\{0,\cdot\}$. Following the strategy of Subsection 3.2, we can write the parametric problem $f^{\text{sub}}(\gamma)$ for Problem (8) and find its fixed point γ. Furthermore, we can construct value functions at each vertex k to consider its subtree's contribution to the parametric problem when $x_k = 1$. We add the new constraint to $V_\gamma(k)$ to obtain the new value function $V_\gamma^{\text{sub}}(k)$:

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

Then the parametric problem takes value $f^{\text{sub}}(\gamma) = \max\{V_\gamma^{\text{sub}}(\text{root}), \sum_{i=1}^n (\pi_i - \gamma)^+ w_i\}$. The first term in $f^{\text{sub}}(\gamma)$ represents the case that $\mathcal{S}^{PHY} \neq \emptyset$. The second term represents the case that $\mathcal{S}^{PHY} = \emptyset$, and is optimized by taking $y_i = \mathbb{1}[\pi_i - \gamma \geq 0]$.

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 these products are not displayed in-store. If $x_k = 0$, then $x_i = 0$ for all $i \in L(k)$, and $y_i = \mathbb{1}[\pi_i - \gamma \geq 0]$ optimizes the value function. We have the following dynamic program:

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

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

The parametric 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, $\mathcal{S}^{PHY}$ is identified by reaching the base cases and $\mathcal{S}^{ON} = \mathcal{S}^{PHY} \cup \{i \mid \pi_i \geq \gamma\}$.

The number of operations to compute $V_\gamma^{\text{sub}}(\text{root})$ is the same as $V_\gamma(\text{root})$. In order to analyze the total runtime, we can rewrite the objective function using Lemma 1 and introduce variables z_{ki} for every $k \in \mathcal{V}$ and $i \in \mathcal{N}$ such that $z_{ki} = x_k y_i$. The equality can be rewritten as $z_{ki} \leq y_i$, $z_{ki} \leq x_k$, and $z_{ki} \geq x_k + y_i - 1$. Hence, the feasible region becomes a subset of $\{0,1\}^{O(n^2)}$ and we need $O(n^4 \log^2 n)$ iterations of Newton's method to arrive at our fixed point. Compared to Theorem 1, the runtime increases by a factor of $O(n^2)$. Using the same techniques, we can study the case where the online and in-store assortments are independent by removing $x_i \leq y_i$ from Problem (8).

6.2. Extending the Features Tree: Leaves as Subsets of Similar Products

The features tree is best for modeling a diverse assortment of products, which can be differentiated by prominent features like product line and brand. We extend the model so that each leaf represents a set of very similar products which arbitrarily share precise features. In Figure 1, the retailer can offer the same colour in multiple product lines of lipsticks, but not across lipsticks and lip gloss.

Let I denote the leaves of our features tree. For $i \in I$, let $N(i) \subseteq N$ be a partition of products such that all products in $N(i)$ share the features in $A(i)$, as well as other features arbitrarily. For $i, i' \in I$ and $i \neq i'$, product $j \in N(i)$ and product $j' \in N(i')$ only share the features in $A(i) \cap A(i')$. We use binary vector y^i to denote which products in $N(i)$ are in the display assortment, such that $y^i_j = \mathbb{1}[j \in S \cap N(i)]$. We continue to use binary vector $x \in \mathcal{X}$, such that $x_k = 1$ implies that feature k on the features tree is displayed. Since a leaf feature i is displayed if and only if a product in $N(i)$ is displayed, we add constraints $\sum_{j \in N(i)} y^i_j \geq x_i$ and $x_i \geq y^i_j$ for all $j \in N(i)$. These constraints ensure that $x_i = 1$ if and only if $y^i_j = 1$ for some $j \in N(i)$. Finally, let $w_j(y^i)$ denote the base preference weight of product $j \in N(i)$ as a result of the features that it shares with products in $N(i)$, excluding the features in $A(i)$ which are on the tree. The final preference weight of product j is $v_j(x, y^i) = w_j(y^i) \cdot \prod_{k \in A(i)} \delta_k^{x_k}$. The OCA problem in the showroom setting is:

$$\max_{\substack{x \in \mathcal{X} \\ y^i \in \{0,1\}^{|N(i)|} : \, i \in I}} \left\{ \frac{\sum_{i \in I} \left(\sum_{j \in N(i)} \pi_j w_j(y^i) \right) \cdot \prod_{k \in A(i)} \delta_k^{x_k}}{w_0 + \sum_{i \in I} \left(\sum_{j \in N(i)} w_j(y^i) \right)} \;\middle|\; \begin{array}{ll} \sum_{j \in N(i)} y_j^i \geq x_i & \forall i \in I, \\ x_i \geq y_j^i & \forall j \in N(i), i \in I \end{array} \right\}.$$

Suppose $|N(i)| \leq n'$ for some small n' and there is limited customization at the last level of the tree. We can use the algorithm in Subsection 3.2 and enumeration at the base case of the dynamic program. This increases the runtime in Theorem 1 by a factor of $O(2^{n'})$.

6.3. Cardinality Constraint for the Physical Store

In the general setting, we may want to impose a cardinality constraint on the number of products offered in the physical store. Due to the limited amount of space in his physical store, the retailer may be restricted to displaying C products in-store, so that $\sum_{i=1}^{n} x_i \leq C$. The geometric grid and parametrization techniques in Subsection 4.1 still apply. Since $C \leq n$ and C is an integer, we do not need to round the parameters of this constraint and we can incorporate the cardinality constraint directly into our value function from Subsection 4.2. Let $\tilde{V}_\gamma^{R,U}(k, y_1, y_2, c)$ be defined as in Subsection 4.2, with the extra condition that we offer at most c products from $L(k)$:

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

The cardinality constraint is handled in the same manner as the second constraint. Computing the optimal assortment increases the runtime in Theorem 3 by a factor of $O(n^2)$. Alternatively, it increases the runtime in Theorem 4 by a factor of $O(n)$ in Online Appendix B.

7. Numerical Study: Modeling Power of Features Tree Model

The features tree model is meant for products that follow the tree structure. Under this model, customers do not update their preferences for lower-level features if two products belong to different categories, which is represented by different branches of the tree. In Figure 2a, suppose the retailer introduces a pink, embroidered gown to the online store. The features tree does not allow customers to evaluate the colour of the new gown with respect to its online description when she sees the pink, pleated gown (product 1), because the highlight is a lower-level feature which is not shared by gowns of different styles. In this section, we measure how well the features tree model can capture customers' choice behaviour when customers follow a more complex choice model in reality. We use the generalized model in Figure 2c as the ground-truth model.

We test two aspects of our features tree model when customers make purchasing decisions according to the ground-truth model: its ability to approximate the true purchase probabilities and its ability to select an assortment that earns a large fraction of the true optimal expected revenue. To benchmark, we test against a simple, omnichannel choice model that does not incorporate products' features. We run our tests for the showroom setting to focus on online customers, and use set notation $\mathcal{S}$ to denote the display assortment. Under the features tree model, the probability that a customer chooses product i is $P_i^{\mathrm{T}}(\mathcal{S})$ and the expected revenue is $\Pi^{\mathrm{T}}(\mathcal{S})$.

7.1. Ground-Truth and Benchmark Models

The ground-truth model is a generalization of the list-of-features model and the features tree model (see Figure 2c). Suppose there are L feature classes and K feature values per class. A product is a combination of one feature value per feature class. We do not require that a product exists for every combination and allow $n < K^L$. Clearly, the ground-truth model subsumes the list-of-features model. The ground-truth model also subsumes the features tree model because we can define the L feature classes to be the levels of the tree, and the K feature values to be the features at each level regardless of their parents in the tree. We can remove all combinations of feature values that do not result in a product to recover the features tree model. In the ground-truth model, product i has initial preference weight w_i. Each 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 $\delta_{\ell,k}$. After updating preference weights, customers make their purchasing decisions from the full assortment $\mathcal{N}$ and the no-purchase option according to MNL. Given a display assortment $\mathcal{S}$, the probability that customers choose product i is denoted $P_i^{\mathrm{G}}(\mathcal{S})$ and the expected revenue is denoted $\Pi^{\mathrm{G}}(\mathcal{S})$.

The benchmark model is a special case of the list-of-features model and the features tree model. We consider a simple extension of MNL to the omnichannel retail setting by dropping the features dependence between products. Product i has preference weight v_i if it is displayed in-store and w_i otherwise. Customers choose from the full assortment $\mathcal{N}$ and the no-purchase option according to MNL by applying the appropriate preference weights. Given a display assortment $\mathcal{S}$, the probability that customers choose product i is denoted $P_i^{\mathrm{B}}(\mathcal{S})$ and the expected revenue is denoted $\Pi^{\mathrm{B}}(\mathcal{S})$. We defer the mathematical details of both models to Online Appendix C.

7.2. Test of Predictive Ability

First, we test the features tree model's ability to approximate the true purchase probabilities if customers make purchasing decisions according to the ground-truth model. We generate many instances of the ground-truth model and for each instance, we generate purchase data by randomly generating display assortments and the resulting purchase. We fit a features tree to the data of each instance, and use the KL-divergence to measure the performance of the fitted model.

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. A feature is denoted by its class-value pair (ℓ, k), and $A(i)$ denotes the set of features of product i. Feature (ℓ, k) is associated with utility $\mu_{\ell,k}$, such that $e^{\mu_{\ell,k}}$ is generated uniformly over $[0,2]$. A product's utility is the sum of its features' utilities. Following the structure of MNL, the preference weight of product i is $w_i = e^{\sum_{(\ell,k)\in A(i)} \mu_{\ell,k}}$. The multiplier $\delta_{\ell,k}$ 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 $\mathcal{N}$ on display will leave without a purchase 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 $\mathcal{S}_d$ is generated such that each product is displayed with probability 0.1. We generate the customer's purchasing decision $i_d \in \mathcal{N} \cup \{0\}$ according to the purchase probabilities under the ground-truth model.

The structure of the features tree depends on how we order the feature classes. We start with $L!$ features trees, where the levels of each tree corresponds to a permutation of the L feature classes. As described in Section 3.3, the value of $L = |A(i)|$ is expected to be small. The parameters of each features tree are then 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 largest log-likelihood. Similarly, we estimate the parameters of the benchmark model by finding a local optimal solution to its maximum likelihood estimator.

For $(L,K) \in \{2,3\} \times \{4,5,6\}$, we generate $M = 100$ instances of the ground-truth model. We fit a features tree model and a benchmark model according to the procedure above. For $m = 1, \ldots, M$, we measure the KL-divergence between the distributions under the ground-truth model and the fitted features tree model. Since the probability distribution depends on the assortment, we compute the average KL-divergence over all display assortments: $\mathrm{KL}^{\mathrm{T}}_m = \frac{1}{2^n} \sum_{S\subseteq\mathcal{N}} \sum_{i\in\mathcal{N}} P^{\mathrm{G}}_i(S) \cdot \log\left(P^{\mathrm{G}}_i(S)/P^{\mathrm{T}}_i(S)\right)$. Similarly, we compute the KL-divergence, $\mathrm{KL}^{\mathrm{B}}_m$, between the ground-truth model and the fitted benchmark model. This measure is always non-negative, and a lower value corresponds to a better approximation of the ground-truth model.

Finally, we want to measure the robustness of the fitted features tree model. We decide to use the features tree with the largest log-likelihood, but it is also reasonable to use one of the other $L! - 1$ features tree created by ordering the feature classes differently. Hence, for each instance m, we also report the median value of $\mathrm{KL}^{\mathrm{T}}_m$ over the $L!$ possible features tree.

		Performance									Robustness		
		Tree-KL_m^{T}				Benchmark-KL_m^{B}					Median KL_m^{T} of $L!$ trees		
L	K	Avg.	5th perc.	95th perc.	Time (sec.)	Avg.	5th perc.	95th perc.	Time (sec.)	Avg change in KL_m	Avg.	5th perc.	95th perc.
2	4	0.058	0.028	0.103	50.15	0.087	0.038	0.164	3.07	-28.9%	0.088	0.040	0.158
2	5	0.060	0.027	0.122	53.13	0.080	0.035	0.159	2.81	-18.1%	0.078	0.036	0.147
2	6	0.057	0.024	0.103	65.79	0.074	0.035	0.134	3.24	-17.2%	0.074	0.033	0.144
3	4	0.095	0.043	0.225	189.06	0.114	0.042	0.272	2.91	-10.1%	0.133	0.050	0.295
3	5	0.075	0.035	0.135	177.86	0.088	0.034	0.162	2.88	-8.8%	0.104	0.043	0.218
3	6	0.073	0.032	0.137	232.05	0.093	0.044	0.178	0.58	-16.2%	0.104	0.053	0.185

Table 1 Average KL-divergence between the purchase probabilities of the ground-truth and fitted models.

Results: In Table 1, we report the average, the 5th percentile, and the 95th percentile of $\text{KL}_m^{\text{T}}, \text{KL}_m^{\text{B}}$ over $M = 100$ instances for the fitted features tree and benchmark models. We also report the time used to estimate each model. In the last three columns, we report the median KL_m^{T} over the $L!$ possible fitted trees as a measure of robustness.

Our features tree model performs quite well in approximating the purchase probabilities of the ground-truth model. The KL-divergence is 8.8% to 28.9% smaller when we approximate the ground-truth model using the features tree with the largest log-likelihood instead of the benchmark model. This implies that our chosen features tree is a better approximation to the ground-truth model than the benchmark. The median KL-divergence is, on average, quite similar to the KL-divergence of the benchmark model. Hence, it is important to choose a good, but not necessarily the best, ordering of the feature classes when we build the features tree model. Next, we demonstrate that the improvements in the KL-divergence translate into significant revenue improvements even when the selected features tree is not necessarily the best one.

7.3. Test of Assortments Selected by Features Tree

The main advantage of the features tree model over the generalized ground-truth model is its computational tractability, so that we can compute an optimal assortment in polynomial runtime. We test the optimal assortment of the fitted features tree against the true optimal assortment.

Setup: For each instance of the ground-truth model in the previous subsection, we create $J = 250$ scenarios by generating a new price vector for each scenario. The price of each product is generated uniformly over $[1, 10]$. We denote the expected revenue function in scenario j as $\Pi_j^{\text{G}}(\cdot)$, 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 $\mathcal{S}_j^{\text{T}} = \arg\max_{\mathcal{S} \subseteq \mathcal{N}} \Pi_j^{\text{T}}(\mathcal{S})$ under the features tree model, using the features tree with the largest log-likelihood from Subsection 7.2. The true optimal solution $\mathcal{S}_j^* = \arg\max_{\mathcal{S} \subseteq \mathcal{N}} \Pi_j^{\text{G}}(\mathcal{S})$ is found by enumeration. For each instance, we measure the fraction of the true optimal expected revenue earned by assortment $\mathcal{S}_j^{\text{T}}$ in scenarios $j = 1, \dots J$, and then average over its J scenarios. We denote this measure by $\text{Earn}^{\text{T}} = \frac{1}{J} \sum_{j=1}^{J} \Pi_j^{\text{G}} \left(\mathcal{S}_j^{\text{T}}\right) / \Pi_j^{\text{G}} \left(\mathcal{S}_j^*\right)$. We perform the same test and compute Earn^{B} for the benchmark model.

		Performance									**Robustness**	
		Tree - $\mathrm{Earn}_m^{\mathrm{T}}$			**Benchmark -** $\mathrm{Earn}_m^{\mathrm{B}}$			**Improvement rel. to benchmark**			**Median** $\mathrm{Earn}_m^{\mathrm{T}}$ **of** $L!$ **trees**	
L	**K**	**Avg.**	**5th perc.**	**95th perc.**	**Avg.**	**5th perc.**	**95th perc.**	**Avg.**	**5th perc.**	**95th perc.**	**Avg.**	**Rel. to** $\mathrm{Earn}_m^{\mathrm{B}}$
2	4	0.9686	0.9449	0.9834	0.9208	0.8626	0.9697	5.74%	1.33%	13.31%	0.9540	3.68%
2	5	0.9675	0.9405	0.9839	0.9294	0.8705	0.9675	4.48%	0.96%	9.35%	0.9596	3.30%
2	6	0.9700	0.9442	0.9863	0.9342	0.8895	0.9669	4.16%	1.29%	8.31%	0.9619	3.00%
3	4	0.9422	0.8847	0.9738	0.8865	0.7904	0.9526	7.13%	1.53%	13.17%	0.9215	4.05%
3	5	0.9562	0.9256	0.9778	0.9109	0.8519	0.9580	5.53%	1.77%	11.46%	0.9407	3.32%
3	6	0.9518	0.9173	0.9796	0.9112	0.8495	0.9574	5.03%	1.01%	9.84%	0.9379	2.98%

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

We reuse the $M = 100$ instances from Subsection 7.2 to compute $\mathrm{Earn}_m^{\mathrm{T}}$ and $\mathrm{Earn}_m^{\mathrm{B}}$ for instance $m = 1, \ldots, M$. We also measure the robustness of the features tree model by considering expected revenue earned by the other $L! - 1$ possible features trees. For each instance m, we record the median value of $\mathrm{Earn}_m^{\mathrm{T}}$ over the $L!$ trees from the different orderings of the feature classes.

Results: In Table 2, we present the average of $\mathrm{Earn}_m^{\mathrm{T}}$ and $\mathrm{Earn}_m^{\mathrm{B}}$, as well as their 5th and the 95th percentile. We report the improvement in these measures when we use the features tree model rather than the benchmark model. In the last two columns, we report the median of $\mathrm{Earn}_m^{\mathrm{T}}$ over the $L!$ possible features tree, and compare these results to the benchmark model.

Our features tree model performs quite well and chooses display assortments that capture large fractions of the optimal expected revenue under the ground-truth model. On average, the features tree and the benchmark models capture 0.9594 and 0.9155 of the true optimal expected revenue respectively. On the low end of performance at the 5th percentile, the models capture 0.9262 and 0.8524 of the optimal expected revenue on average. Relative to the benchmark model, the expected revenue increases by 5.35% on average when we use the features tree model.

The features tree model performs well even if we order the feature classes differently. On average, the median value of $\mathrm{Earn}_m^{\mathrm{T}}$ over the $L!$ possible trees is still more than 0.9215 of the true optimal expected revenue. The median value of $\mathrm{Earn}_m^{\mathrm{T}}$ over the $L!$ possible trees is 3.39% larger than $\mathrm{Earn}_m^{\mathrm{B}}$ on average. Hence, even if we select another ordering of the feature classes for our features tree, we can still expect to outperform the benchmark model and achieve a large fraction of the optimal expected revenue. The ground-truth model generalizes the list-of-features model, which is already computationally difficult to optimize over. Our model is a good substitute because it is computationally tractable and achieves a reasonable fraction of the optimal expected revenue.

8. Numerical Study: Performance of FPTAS

We test our FPTAS to assess its practical performance compared to the its theoretical guarantee. We compare the expected revenue earned by our FPTAS against the upper-bound from Section 5.

Setup: We assume that offline and online customers have the same preference weight for product i if they both see it, so that $\hat{v}_i = w_i \cdot \prod_{k\in A(i)} \delta_k$ 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 revenue π_i is generated uniformly over $[1, 10]$. Each product's initial preference weight w_i is generated uniformly over $[1, 5]$. To set up our features tree, recall from Assumption 1 that we can convert a general features tree to a binary features tree by introducing auxiliary vertices such that $\delta_k = 1$. We use this idea to construct a general features tree, while representing it as a binary tree. First, we set up a balanced binary tree with 32 products as leaves. For a leaf $i \in \mathcal{N}$, 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 $\{0, 0.05, 0.1, 0.2\}$ and a larger β increases the probability that the features tree is a general tree. Finally, the preference weight of the no-purchase option is chosen so that a customer who sees the full assortment will choose the no-purchase option with probability $\eta \in \{0.05, 0.1\}$.

For each instance, we vary the fraction of online customers q. For $q \in \{0.2, 0.4, 0.6, 0.8\}$, 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 fraction of the upper-bound earned by x^q: $\text{EarnLP}^q = \Pi^q(x^q)/\bar{\Pi}^q$. We use the constrained longest path formulation in Online Appendix B to improve the runtime.

For each $(\eta, \beta) \in \{0.05, 0.1\} \times \{0, 0.05, 0.1, 0.2\}$, we generate $M = 50$ instances of the problem. For each instance m, we compute EarnLP^q_m for $q \in \{0.2, 0.4, 0.6, 0.8\}$. 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. For each (η, β), we also run our polynomial-time algorithm for the showroom setting and our FPTAS for the general setting with $n = 16, 32, 48, 64$ to measure the growth in the algorithms' runtimes. We compute the ratio of the runtimes for $n = 16, 48, 64$ to the runtime for $n = 32$. We also consider the ratio of the theoretical runtimes, as determined by Theorems 1 and 4, by looking at the leading polynomial term in n and ignoring the logarithmic terms. These tests were performed on a computer with 16 GB of RAM and a Intel i7-6700 processor.

Results: In Table 3, each row describes a combination of (η, β), followed by the average, minimum, and maximum of EarnLP^q_m. On average, our FPTAS achieves 0.984 of the upper-bound on optimal expected revenue, which is significantly better than its performance guarantee of 1/2. The upper-bound uses a relaxation of our OCA problem and is not tight. On average, our FPTAS achieves 0.984 of the upper-bound. The average runtime of our FPTAS is 155.5 seconds.

The ratios of the algorithms' runtimes are reported in Table 4. The second sub-columns report the ratio between the actual runtime for each n versus $n = 32$. The third sub-columns report the

		q= 0.2			**q= 0.4**			**q= 0.6**			**q= 0.8**			
η	β	**Avg.**	**Min.**	**Max.**	**Avg.**	**Min.**	**Max.**	**Avg.**	**Min.**	**Max.**	**Avg.**	**Min.**	**Max.**	**Time (sec.)**
0.05	0.00	0.971	0.932	0.990	0.982	0.959	0.995	0.989	0.971	0.996	0.996	0.988	0.998	158.9
0.05	0.05	0.970	0.935	0.992	0.982	0.964	0.995	0.989	0.976	0.996	0.996	0.990	0.998	191.5
0.05	0.10	0.968	0.932	0.993	0.981	0.959	0.995	0.991	0.973	0.996	0.996	0.988	0.998	186.0
0.05	0.20	0.966	0.932	0.990	0.979	0.958	0.993	0.989	0.972	0.996	0.995	0.988	0.998	173.6
0.10	0.00	0.967	0.928	0.992	0.981	0.959	0.994	0.989	0.968	0.996	0.995	0.963	0.998	140.2
0.10	0.05	0.972	0.941	0.992	0.983	0.959	0.995	0.989	0.971	0.997	0.996	0.976	0.999	113.3
0.10	0.10	0.965	0.928	0.988	0.980	0.953	0.993	0.989	0.974	0.996	0.995	0.973	0.998	149.7
0.10	0.20	0.970	0.932	0.990	0.983	0.960	0.994	0.990	0.959	0.996	0.995	0.958	0.999	131.0

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

	$\mathbf{n = 16}$			$\mathbf{n = 32}$			$\mathbf{n = 48}$			$\mathbf{n = 64}$		
	Avg. time (sec.)	**Actual rel. to** $n = 32$	**Theor. rel.** $n = 32$	**Avg. time (sec.)**	**Actual rel. to** $n = 32$	**Theor. rel.** $n = 32$	**Avg. time (sec.)**	**Actual rel. to** $n = 32$	**Theor. rel.** $n = 32$	**Avg. time (sec.)**	**Actual rel. to** $n = 32$	**Theor. rel.** $n = 32$
Show.	2.23E-5	0.49	0.13	4.60E-5	1.00	1.00	2.04E-4	4.44	3.38	3.25E-4	7.07	8.00
FPTAS	16.37	0.11	0.03	155.51	1.00	1.00	764.66	4.92	7.59	1,848.74	11.89	32.00

Table 4 Runtime of algorithms for showroom and general settings relative to $n = 32$, for $n = 16, 32, 48, 64$.

ratio between the theoretical runtimes. In the showroom setting, the ratios between the actual runtimes matches the ratio between the theoretical runtimes, except when $n = 16$. When $n = 16$, the ratio between the actual runtimes is four times greater than the theoretical ratio. One possible explanation is that the setup time (e.g. initializing arrays for dynamic program) is significant compared to the computation time when n is small. The same observation can be made for the FPTAS when $n = 16$. In contrast to the showroom setting, the ratio between the actual runtimes of the FPTAS is about half of the ratio between the theoretical runtimes when $n = 48$ and 64. In the practical implementation of the FPTAS, we can omit certain (R, U) from $\mathcal{K}_{\text{feas}} \subseteq \mathcal{K}_\epsilon$ without any computation. If Problem (4) is infeasible at $(0, R/(1+\epsilon), (1+\epsilon)U)$, then it must also be infeasible at $(0, R, U)$. Hence, the upper-bound for the theoretical runtime in Theorem 4 is not tight. Finally, the FPTAS takes 0.5 hours to terminate when $n = 64$, which is reasonable for practical implementation because computing the in-store assortment with the FPTAS is not an online exercise.

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.

One future research direction is to vary the fraction of customers that visit each store based on the assortments. In reality, customers choose between different retail channels based on the diversity of the assortments and the prices or possible discounts in each channel. The assortments not only influence customers' preferences for the products available in the other store, but it also influences the fraction of customers who purchase from each store. 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 may be other ways to explain how customers make purchasing decisions when they can access multiple channels. In our model, customers maximize their utilities under the structure of standard MNL, except that the mean utilities (and hence preference weights) are determined by the features tree as a function of the in-store assortment. We can incorporate the features tree into other choice models which use MNL as a building block. In the context of the mixture of MNL models, different customer segments would determine their preference weights according to different features trees. With the d-level nested logit model, customers would still choose among the options within a selected nest. However, the preference weights of the products would depend on the in-store assortment via the features tree, which affects the probability of selecting each nest by changing its total preference weight. The features tree describes the impact on customers' mean utilities when they see products in the physical store, whereas the underlying choice model describes the customers' choice process.

Finally, parameter estimation is an important aspect of assortment optimization because 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 to evaluate the usefulness of the features tree model in practice.

References

Adrienne Söndag (2019), 'A Guide To MAC Cosmetic Part 2 - Lipstick Formulas', http://www.thesundaygirl.com/2012/12/a-guide-to-mac-cosmetics-part-2.html. Accessed: 2019-05-17.

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.

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* .

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.

David's Bridal (2020), 'Dresses', https://www.davidsbridal.com. Accessed: 2020-07-08.

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* .

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.

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.

Goldstone, R. L. (1994), 'Similarity, interactive activation, and mapping.', *Journal of Experimental Psychology: Learning, Memory, and Cognition* **20**(1), 3.

Gregan-Paxton, J., Hoeffler, S. and Zhao, M. (2005), 'When categorization is ambiguous: Factors that facilitate the use of a multiple category inference strategy', *Journal of Consumer Psychology* **15**(2), 127–140.

Harsha, P., Subramanian, S. and Ettl, M. (2019), 'A practical price optimization approach for omnichannel retailing', *Informs Journal on Optimization* **1**(3), 241–264.

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.

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

M.A.C Cosmetics (2019*a*), 'Products: Lip Gloss', https://www.maccosmetics.com/products/13853/Products/Makeup/Lips/Lip-Gloss. Accessed: 2019-05-13.

M.A.C Cosmetics (2019*b*), 'Products: Lipstick', https://www.maccosmetics.com/products/13854/Products/Makeup/Lips/Lipstick. Accessed: 2019-05-13.

Markman, A. B. and Wisniewski, E. J. (1997), 'Similar and different: The differentiation of basic-level categories.', *Journal of Experimental Psychology: Learning, memory, and cognition* **23**(1), 54.

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

Moreau, C. P., Markman, A. B. and Lehmann, D. R. (2001), '"What is it?" Categorization flexibility and consumers' responses to really new products', *Journal of Consumer Research* **27**(4), 489–498.

Murphy, G. L. and Brownell, H. H. (1985), 'Category differentiation in object recognition: typicality constraints on the basic category advantage.', *Journal of Experimental Psychology: Learning, Memory, and Cognition* **11**(1), 70.

Murphy, G. L. and Ross, B. H. (2010), 'Uncertainty in category-based induction: When do people integrate across categories?', *Journal of Experimental Psychology: Learning, Memory, and Cognition* **36**(2), 263.

Noseworthy, T. J. and Goode, M. R. (2011), 'Contrasting rule-based and similarity-based category learning: The effects of mood and prior knowledge on ambiguous categorization', *Journal of Consumer Psychology* **21**(3), 362–371.

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

Rosch, E., Mervis, C. B., Gray, W. D., Johnson, D. M. and Boyes-Braem, P. (1976), 'Basic objects in natural categories', *Cognitive psychology* **8**(3), 382–439.

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.

Yamauchi, T. and Markman, A. B. (1998), 'Category learning by inference and classification', *Journal of Memory and language* **39**(1), 124–148.

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.