The d-Level Nested Logit Model: Assortment and Price Optimization Problems

Guang Li guangli@usc.edu

Paat Rusmevichientong rusmevic@marshall.usc.edu

Huseyin Topaloglu topaloglu@orie.cornell.edu

November 18, 2014 @ 4:04pm

Abstract

We consider assortment and price optimization problems under the d-level nested logit model. In the assortment optimization problem, the goal is to find the revenue-maximizing assortment of products to offer, when the prices of the products are fixed. Using a novel formulation of the d-level nested logit model as a tree of depth d, we provide an efficient algorithm to find the optimal assortment. For a d-level nested logit model with n products, the algorithm runs in $O(d\,n \log n)$ time. In the price optimization problem, the goal is to find the revenue-maximizing prices for the products, when the assortment of offered products is fixed. Although the expected revenue is not concave in the product prices, we develop an iterative algorithm that generates a sequence of prices converging to a stationary point. Numerical experiments show that our method converges faster than gradient-based methods, by many orders of magnitude. In addition to providing solutions for the assortment and price optimization problems, we give support for the d-level nested logit model by demonstrating that it is consistent with the random utility maximization principle and equivalent to the elimination by aspects model.

1. Introduction

In their seminal work, Talluri and van Ryzin (2004) demonstrated the importance of incorporating customer choice behavior when modeling demand in operations management problems. They observed that customers make a choice among the available products after comparing them in terms of price, quality, and possibly other features. This choice process creates interactions among the demand for different products, and it is important to capture these interactions. One of the models that is most commonly used to capture the choice process of customers among different products is the multinomial logit model, pioneered by Luce (1959) and McFadden (1974). In this paper, we consider the d-level nested logit model, allowing for an arbitrary number of levels d. In this model, each product is described by a list of d features. As an example, if the products are flight itineraries, then such features might include the departure time, fare class, airline, and the number of connections. We describe the products by using a tree with d levels, where each level in the tree corresponds to a particular feature. When choosing a product, a customer first chooses the desired value of the first feature, say the departure time, which gives rise to a subset of compatible flight itineraries. Then, she chooses the value of the second feature, which further narrows down the set of compatible flight itineraries. Each subsequent feature selection reduces the compatible set of flight itineraries further, until we are left with a single flight itinerary after d features have been selected. In this way, each product corresponds to a leaf node in the tree, whose unique path of length d from the root node describes the list of d features of the product.

We consider both assortment optimization and pricing problems under the d-level nested logit model. In the assortment optimization problem, the prices of the product are fixed. The goal is to find the revenue-maximizing assortment of products to offer. In the pricing problem, the assortment of offered products is fixed, but the utility that a customer derives from a product depends on the price of the product. The goal is to find the revenue-maximizing set of prices for the products. We make contributions in terms of formulation of the d-level nested logit model, solution to the assortment optimization problem and solution to the pricing problem. We proceed to describing our contributions in each one of these three domains.

We provide a novel formulation of the d-level nested logit model using a tree of depth d. Building on this formulation, we can describe the product selection probabilities through a succinct recursion over the nodes of the tree. Using the succinct description of the product selection probabilities, we formulate our assortment optimization and pricing problems. To our knowledge, this is the first paper to present a description of the d-level nested logit model that works for an arbitrary tree structure and an arbitrary number of levels.

We provide a complete solution to the assortment optimization problem. We give a recursive characterization of the optimal assortment, which, in turn, leads to an efficient assortment optimization algorithm for computing the optimal assortment. For a d-level nested logit model with n products, the running time of the algorithm is $O(d\,n\,\log n)$. To our knowledge, this is the first solution to the assortment optimization problem under the d-level nested logit model.

For the pricing problem, it is known that the expected revenue is not concave in the prices of the products even when the customers choose according to the two-level nested logit model. Thus, we cannot hope to find the optimal prices in general. Instead, we focus on finding stationary points, corresponding to the prices at which the gradient of the expected revenue function is zero. Using our recursive formulation of the d-level nested logit model, we give a simple expression for the gradient of the expected revenue. Although our gradient expression allows us to compute a stationary point by using a standard gradient ascent method, we find the gradient ascent method to be relatively slow and it requires a careful selection of step size. We develop a new iterative algorithm that generates a sequence of prices converging to a stationary point of the expected revenue function. Our algorithm is different from gradient ascent because the prices it generates do not follow the direction of the gradient. Furthermore, our algorithm completely avoids the problem of choosing a step size. Numerical experiments show that our algorithm converges to a stationary point of the expected revenue function much faster than gradient ascent.

2. Literature Review

We focus our literature review on papers that use variants of the multinomial and nested logit models in assortment and price optimization problems. We refer the reader to Kök et al. (2006) and Farias et al. (2013) for assortment optimization and pricing work under other choice models. The multinomial logit model dates back to the work of Luce (1959) and it is known to be consistent with random utility maximization. However, this model suffers from the independence of irrelevant alternatives property, which implies that if a new product is added to an assortment, then the demand for each existing product decreases by the same percentage. This property should not hold when different products cannibalize each other to different extents and the multinomial logit model can lead to biased estimates of the selection probabilities in such settings; see Train (2003). The nested logit model avoids the independence of irrelevant alternatives property, while remaining compatible with random utility maximization; see Börch-Supan (1990).

Talluri and van Ryzin (2004) and Gallego et al. (2004) consider assortment optimization problems under the multinomial logit model. They show that the optimal assortment is revenue ordered in the sense that it includes a certain number of products with the largest revenues. Liu and van Ryzin (2008) provide an alternative proof of the same result. Gallego et al. (2011) consider generalizations of the multinomial logit model, in which the attractiveness of the no purchase option increases when more restricted assortments are offered to customers. They show that the assortment problem under this choice model remains tractable and make generalizations to the network revenue management setting, in which customers arriving into the system observe the assortment of available fare classes and make a choice among them. Bront et al. (2009) and Rusmevichientong et al. (2013) consider assortment optimization problems under the mixture of logits model. They show that the problem is NP-hard, propose heuristic solution methods and investigate integer programming formulations. The papers by van Ryzin and Mahajan (1999) and Li (2007) consider models in which a firm needs to make joint assortment optimization and stocking quantity decisions. There is a certain revenue and a procurement cost associated with each product. Once the firm chooses the products to offer and their corresponding stocking quantities, a random number of customers arrive into the system and each customer chooses among the offered products according to the multinomial logit model. The goal is to choose the assortment of offered products and the stocking quantities to maximize the expected profit. Li (2007) assumes that the demand for each product is proportional to a random store traffic volume, whereas van Ryzin and Mahajan (1999) model the demand for different products with random variables whose coefficients of variation decrease with demand volume, capturing economies of scale. The paper by van Ryzin and Mahajan (1999) assumes that products have the same profit margin and shows that the optimal assortment includes a certain number of products with the largest attractiveness parameters in the choice model. If there are p products, then their result reduces the number of assortments to consider to p. For a nested logit model with m nests and p products in each nest, even if their ordering result holds in each nest, the number of assortments to consider is p^m.

Assortment optimization under the two-level nested logit model has been considered only recently. Kök and Xu (2010) study joint assortment optimization and pricing problems under the two-level nested logit model, where both the assortment of offered products and their prices are decision variables. They work with two nest structures. In the first nest structure, customers first choose a brand out of two brands, and then, a product type within the selected brand. In the second nest structure, customers first choose a product category, and then, a brand for the selected product category out of two brands. The authors show that the optimal assortment of product types within each brand is popularity ordered, in the sense that it includes a certain number of product types with the largest mean utilities. Thus, if there are p product types within each brand, then there are p possible assortments of product types to consider in a particular brand. Since there are two brands, the optimal assortment to offer is one of p^2 assortments. The optimal assortment can be found by checking the performance of all p^2 assortments. The problem becomes intractable when there is a large number of brands. If there are b brands, then the number of possible assortments to consider is p^b, which quickly gets large with b. We note that if we apply our assortment optimization algorithm in Section 5 to the two-level nested logit model with b brands and p product types within each brand, then the optimal assortment can be computed in $O(2pb\log(pb))$ operations because the total number of products is pb.

Davis et al. (2014) show how to compute the optimal assortment under the two-level nested logit model with an arbitrary number of nests. Assuming that there are m nests and p products within each nest (for a total of mp products), the authors show that there are only p possible assortments of products to consider in each nest. Furthermore, they formulate a linear program to find the best one of these assortment to choose from each nest. In this way, their approach avoids checking the performance of all p^m possible values for the optimal assortment. Gallego and Topaloglu (2012) extend this work to accommodate a variety of constraints on the offered assortment. Li and Rusmevichientong (2014) establish structural properties and use these properties to develop a greedy algorithm for computing the optimal assortment. All work so far focuses on the two-level nested logit model. To our knowledge, there is no assortment optimization work under the general d-level nested logit model. The linear program proposed by Davis et al. (2014) and the greedy algorithm developed by Li and Rusmevichientong (2014) do not generalize to the d-level nested logit model. We note that, to our knowledge, assortment optimization problems under other variants of logit models – the mixture of logits, the cross nested logits (Vovsha, 1997), paired combinatorial and generalized nested logit models (Wen and Koppelman, 2001) – are not tractable. In this paper, we show that the assortment optimization problem under the nested logit model is tractable, regardless of the number of levels and the structure of the tree.

For pricing problems under the multinomial logit model, Hanson and Martin (1996) point out that the expected revenue is not necessarily concave in prices. Song and Xue (2007) and Dong et al. (2009) make progress by noting that the expected revenue is concave in market shares. Chen and Hausman (2000) study the structural properties of the expected revenue function for a joint product selection and pricing problem. There is some recent work on price optimization under the two-level nested logit model. Li and Huh (2011) consider the case in which the products in the same nest share the same price sensitivity parameter, and show that the expected revenue function is concave in market shares. Gallego and Wang (2013) generalize this model to allow for arbitrary price sensitivity parameters, but the expected revenue function is no longer concave in market shares. Similar to assortment optimization, all of the earlier work on pricing under the nested logit model focuses on two levels. Subsequent to our work, Li and Huh (2013) consider pricing problems under the d-level nested logit model. They give a characterization of the optimal prices, but do not provide an algorithm to compute them and do not address the assortment problem.

The rest of the paper is organized as follows. After the literature review in Section 2, we describe the d-level nested logit model in Section 3 and formulate the assortment optimization problem. In Section, 4, we give a characterization of the optimal assortment. This characterization is translated into an algorithm for computing an optimal assortment in Section 5. In Section 6, we test the numerical performance of our assortment optimization algorithm. In Section 7, we formulate our pricing problem and give an algorithm to find a stationary point of the expected revenue function. In Section 8, we give numerical experiments that test the performance of our pricing algorithm. In Section 9, we provide practical motivation for the d-level nested logit model by showing that it is compatible with random utility maximization principle and equivalent to the elimination by aspects model. Also, we give practical situations where the assortment and price optimization problems studied in this paper become useful. We conclude in Section 10.

3. Assortment Optimization Problem

We have n products indexed by $\{1, 2, \ldots, n\}$, and the no-purchase option is denoted by product 0. The taxonomy of the products is described by a d-level tree, denoted by $\mathsf{T} = (\mathsf{V}, \mathsf{E})$ with vertices V and edges E. The tree has n leaf nodes at depth d, corresponding to the n products in $\{1, 2, \ldots, n\}$. Throughout the paper, we will use the terms depth and level interchangeably. A sample tree with $d = 3$ and $n = 9$ is given in Figure 1. The tree describes the process in which a customer chooses a product. Starting at the root node, denoted by root, the edges emanating from root represent the first criterion, or feature, that the customer uses in choosing her product. Each node in the first level corresponds to subsets of products that fit with a particular value of the first criterion. Product 0, corresponding to the no-purchase option, is labeled as node 0. It is located in level one and directly connected to root, with no children. The edges in the second level represent the second criterion that a customer uses to narrow down her choices, and each level-two node represents a subset of products that are compatible with the particular values of the first two criteria. The same interpretation applies to other levels. We use $\mathsf{Children}(j)$ to denote the set of child nodes of node j in the tree, and $\mathsf{Parent}(j)$ to denote the parent node of node j.

Associated with each node j is a set of products $N_j \subseteq \{0, 1, \ldots, n\}$, where N_j is the set of products, or leaf nodes, that are included in the subtree rooted at node j. In particular, if j is a leaf node, then $N_j = \{j\}$, which is just a singleton consisting of the product itself. On the other hand, if j is a non-leaf node, then $N_j = \dot{\bigcup}_{k \in \mathsf{Children}(j)} N_k$, which is the disjoint union of the sets of products at the child nodes of j. Note that if node j is in level h, then N_j corresponds to the subset of products that fit with the specific values of the first h criteria, as specified by the path from root to node j. We refer to a subset of products $S \subseteq \{1, 2, \ldots, n\}$ as an assortment. Each assortment S defines a collection of subsets $(S_j : j \in \mathsf{V})$ at each node of the tree with $S_j \subseteq N_j$, where if j is a leaf node other than node 0, then

$$S_j = \begin{cases} \{j\} & \text{if } j \in S \\ \varnothing & \text{if } j \notin S \ . \end{cases}$$

If j corresponds to node 0, then we set $S_0 = \{0\}$ by convention. If j is a non-leaf node, then $S_j = \dot{\bigcup}_{k \in \mathsf{Children}(j)} S_k$. Thus, S_j corresponds to the products in S that are included in the subtree rooted at node j; that is, $S_j = S \cap N_j$. Often times, we refer to an assortment S by its collection of subsets $(S_j : j \in \mathsf{V})$, and write $S = (S_j : j \in \mathsf{V})$. To illustrate the notation so far, we consider the tree shown in Figure 1. Including the no-purchase option, there are 10 products in this tree, and they are given by $\{0, 1, \ldots, 9\}$. The nodes of the tree are indexed by $\{0, 1, \ldots, 15\}$. We have, for example,

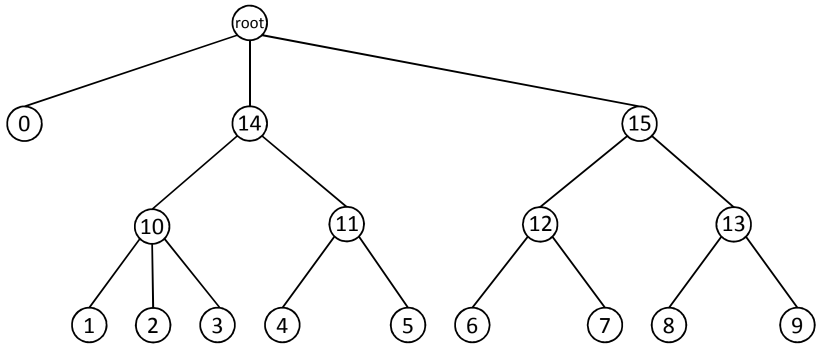


Figure 1: An illustration of the model formulation.

$\mathsf{Children}(10) = \{1,2,3\}$, $\mathsf{Parent}(3) = 10$, $N_2 = \{2\}$, $N_{10} = \{1,2,3\}$, and $N_{14} = \{1,2,3,4,5\}$. For an assortment $S = \{1,3,4,6\}$, we have $S_1 = \{1\}$, $S_2 = \varnothing$, $S_{10} = \{1,3\}$, $S_{11} = \{4\}$, $S_{13} = \varnothing$, $S_{14} = \{1,3,4\}$, and $S_{\mathsf{root}} = \{0,1,3,4,6\}$.

Associated with each leaf node j, we have the attractiveness parameter v_j, capturing the attractiveness of the product corresponding to this leaf node. Given an assortment $S = (S_j : j \in \mathsf{V})$, a customer associates the preference weight $V_j(S_j)$ with each node $j \in \mathsf{V}$, which is a function of the offered assortment $S = (S_j : j \in \mathsf{V})$ and the attractiveness parameters of the products $(v_1, \ldots, v_n)$. To make her choice, a customer starts from root in the tree and walks over the nodes of the tree in a probabilistic fashion until she reaches a leaf node. In particular, if the customer is at a non-leaf node j, then she follows node $k \in \mathsf{Children}(j)$ with probability $V_k(S_k)/\sum_{\ell \in \mathsf{Children}(j)} V_\ell(S_\ell)$. Thus, the customer is more likely to visit the nodes that have higher preference weights. Once the customer reaches a leaf node, if this leaf node corresponds to a product, then she purchases the product corresponding to this leaf node. If this leaf node corresponds to the no-purchase option, then she leaves without making a purchase. As a function of the offered assortment $S = (S_j : j \in \mathsf{V})$ and the attractiveness parameters of the products $(v_1, \ldots, v_n)$, the preference weight $V_j(S_j)$ for each node j in the tree is computed as follows. If j is a leaf node, then S_j is either $\{j\}$ or $\varnothing$. In this case, the preference weight of a leaf node j is defined as $V_j(\{j\}) = v_j$ and $V_j(\varnothing) = 0$. We assume that the no-purchase option, node 0, has a preference weight v_0 so that $V_0(S_0) = v_0$. For each non-leaf node j, the preference weight of this node is computed as

$$V_j(S_j) = \left(\sum_{k \in \mathsf{Children}(j)} V_k(S_k) \right)^{\eta_j}, \tag{1}$$

where $\eta_j \in (0,1]$ is a parameter of the nested logit model associated with node j. As we discuss in Section 9, the d-level nested logit model can be derived by appealing to the random utility maximization principle. Roughly speaking, $1 - \eta_j$ is a measure of correlation between the utilities of the products that are descendants of node j. If η_j is closer to zero, then the utilities of the products that are descendants of node j are more positively correlated. If $\eta_j = 1$, then node j can be omitted from the tree by connecting each child of node j directly to the parent of node j. If $\eta_j = 1$ for all $j \in \mathsf{V}$, then each leaf node can directly be connected to root and we recover the multinomial logit model, corresponding to the case in which the utilities of all products are independent; see McFadden (1978) and Koppelman and Sethi (2000). Since $V_j(S_j)$ determines the probability that a customer chooses node j in her choice process and the customer already starts at root without ever coming back to root, $V_{\mathsf{root}}(S_{\mathsf{root}})$ does not play a role at all in the choice process. Therefore, without loss of generality, we set $\eta_{\mathsf{root}} = 0$ so that we always have $V_{\mathsf{root}}(S_{\mathsf{root}}) = 1$.

The discussion in the paragraph above describes the choice process under the d-level nested logit model. Next, we proceed to formulating our assortment optimization problem. We use r_j to denote the revenue associated with product j. Given that we offer an assortment $S = (S_j : j \in \mathsf{V})$, we use $R_j(S_j)$ to denote the expected revenue obtained from a customer that is at node j in the tree during the course of her choice process. If the customer is at leaf node j and the product corresponding to this leaf node is offered, then the customer purchases the product and a revenue of r_j is obtained from this customer. If the customer is at leaf node j and the product corresponding this leaf node is not offered, then no revenue is obtained from this customer. Therefore, if j is a leaf node, then we can capture the expected revenue from a customer at this node by defining $R_j(\{j\}) = r_j$ and $R_j(\varnothing) = 0$. If a customer is at node 0 corresponding to the no-purchase option, then no revenue is obtained from the customer. Thus, we immediately have $R_0(S_0) = 0$. On the other hand, as mentioned in the paragraph above, if the customer is at a non-leaf node j, then she chooses node $k \in \mathsf{Children}(j)$ with probability $V_k(S_k)/\sum_{\ell \in \mathsf{Children}(j)} V_\ell(S_\ell)$. In this case, we can recursively write the expected revenue from a customer at a non-leaf node j as

$$R_j(S_j) = \sum_{k \in \mathsf{Children}(j)} \frac{V_k(S_k)}{\sum_{\ell \in \mathsf{Children}(j)} V_\ell(S_\ell)} \times R_k(S_k) \quad = \quad \frac{\sum_{k \in \mathsf{Children}(j)} V_k(S_k)\, R_k(S_k)}{\sum_{k \in \mathsf{Children}(j)} V_k(S_k)} \ . \tag{2}$$

Since each customer starts her choice process from root, if we offer the assortment $S = (S_j : j \in \mathsf{V})$, then the expected revenue obtained from a customer is $R_{\mathsf{root}}(S_{\mathsf{root}})$. We want to find an assortment that maximizes the expected revenue from a customer at root, yielding the assortment problem

$$Z^* = \max_{S \subseteq \{1,\ldots,n\}} \quad R_{\mathsf{root}}(S_{\mathsf{root}}) \ . \tag{3}$$

Throughout, we use $S^* = (S_j^* : j \in \mathsf{V})$ to denote an optimal solution to the assortment problem above. The objective function of this assortment problem is defined in a recursive fashion and involves nonlinearities, but it turns out that we can solve this problem in a tractable manner.

4. Properties of An Optimal Assortment

In this section, we give a characterization of the optimal assortment. We use this characterization in the next section to develop an algorithm for computing the optimal assortment. To give our characterization, we let $S^* = (S^*_j : j \in \mathsf{V})$ denote an optimal assortment and define the scalars $(\mathtt{t}^*_j : j \in \mathsf{V})$ for each node in the tree as

$$\mathtt{t}^*_j = \max\left\{ \mathtt{t}^*_{\mathsf{Parent}(j)}\ ,\ \eta_j\, \mathtt{t}^*_{\mathsf{Parent}(j)} + (1-\eta_j)\, R_j(S^*_j) \right\}\ ,$$

with the convention that $\mathtt{t}^*_{\mathsf{Parent(root)}} = 0$. Since $\eta_{\mathsf{root}} = 0$, we have $\mathtt{t}^*_{\mathsf{root}} = R_{\mathsf{root}}(S^*_{\mathsf{root}}) = Z^*$. If the optimal assortment is known, then we can compute $(V_j(S^*_j) : j \in \mathsf{V})$ by starting from the leaf nodes, enumerating the nodes of the tree in a breadth-first manner and using (1). Once we have $(V_j(S^*_j) : j \in \mathsf{V})$, we can compute $(R_j(S^*_j) : j \in \mathsf{V})$ in a similar fashion but by using (2) this time. Finally, once we have $(R_j(S^*_j) : j \in \mathsf{V})$, we can compute $(\mathtt{t}^*_j : j \in \mathsf{V})$ starting from root and enumerating the nodes of the tree in a breadth-first manner. To characterize the optimal assortment, for each node j, we consider the optimization problem

$$\max_{S_j \subseteq N_j} V_j(S_j) \left(R_j(S_j) - \mathtt{t}^*_{\mathsf{Parent}(j)} \right)\ . \tag{4}$$

Problem (4) only considers the products in N_j, which are the products included in the subtree rooted at node j. Thus, we refer to problem (4) as the *local problem at node j*. As mentioned in Section 3, we have $V_{\mathsf{root}}(S_{\mathsf{root}}) = 1$. Since $\mathtt{t}^*_{\mathsf{Parent(root)}} = 0$, comparison of problems (3) and (4) shows that the local problem at root is identical to the assortment optimization problem in (3).

Since the local problem at root is identical to the assortment problem in (3), an optimal solution to problem (3) also solves the local problem at root. In the following proposition, we generalize this observation by showing that if $S^* = (S^*_j : j \in \mathsf{V})$ is an optimal solution to the assortment optimization problem in (3), then for any node j, S^*_j solves the local problem at node j.

Proposition 4.1 (Recovering Optimal Assortment). *Let $S^* = (S^*_j : j \in \mathsf{V})$ be an optimal solution to problem* (3)*. Then, for all $j \in \mathsf{V}$, S^*_j is an optimal solution to the local problem at node j.*

The proof of this proposition is in Online Appendix A. We interpret $\mathtt{t}^*_{\mathsf{Parent}(j)}$ as the minimum expected revenue from a customer at node j to make it worthwhile to offer any of the products included in the subtree rooted at node j. If it is not possible to obtain an expected revenue of $\mathtt{t}^*_{\mathsf{Parent}(j)}$ or more from a customer at node j, then we are better off not offering any of the products included in the subtree rooted at node j. To see this interpretation, note that if $S^* = (S_j^* : j \in \mathsf{V})$ is an optimal assortment for problem (3), then by Proposition 4.1, S_j^* solves the local problem at node j. Also, the optimal objective value of the local problem at node j has to be non-negative, because the empty set trivially yields an objective value of zero for problem (4). Thus, if $S_j^* \neq \varnothing$ so that it is worthwhile to offer any of the products included in the subtree rooted at node j, then we must have $R_j(S_j^*) \geq \mathtt{t}^*_{\mathsf{Parent}(j)}$. Otherwise, S_j^* provides a negative objective value for the local problem at node j, contradicting the fact that S_j^* solves the local problem at node j.

If j is a leaf node, then the local problem at node j given in (4) can be solved efficiently because $N_j = \{j\}$ so that S_j is simply either $\{j\}$ or $\varnothing$. The crucial question is whether we can use a bottom-up approach and construct a solution for the local problem at any node j by using the solutions to the local problems at its child nodes. The following theorem affirmatively answers this question by showing that we can synthesize the solution of the local problem at node j by using the solutions of the local problems at each of its children.

Theorem 4.2 (Synthesis from Child Nodes)**.** *Consider an arbitrary non-leaf node $j \in \mathsf{V}$. If for all $k \in \mathsf{Children}(j)$, $\widehat{S}_k$ is an optimal solution to the local problem at node k, then $\bigcup_{k \in \mathsf{Children}(j)} \widehat{S}_k$ is an optimal solution to the local problem at node j.*

Thus, if we solve the local problem at all nodes $k \in \mathsf{Children}(j)$ and take the union of the optimal solutions, then we obtain an optimal solution to the local problem at node j. We give the proof of Theorem 4.2 in Online Appendix A. In Section 5, we use this theorem to develop an algorithm for computing the optimal assortment. The next corollary shows that the optimal assortment can be obtained by comparing the revenues of the products with the scalars $(\mathtt{t}_j^* : j \in \mathsf{V})$.

Corollary 4.3 (Revenue Comparison)**.** *Let $\widehat{S} = \left\{ \ell \in \{1, 2, \ldots, n\} : r_\ell \geq \mathtt{t}^*_{\mathsf{Parent}(\ell)} \right\}$. Then, $\widehat{S}$ is an optimal solution to problem* (3)*.*

Proof: Let $\widehat{S}_j \equiv \widehat{S} \cap N_j = \left\{ \ell \in N_j : r_\ell \geq \mathtt{t}^*_{\mathsf{Parent}(\ell)} \right\}$, in which case, $\widehat{S}_{\mathsf{root}} = \widehat{S}$. By induction on the depth of node j, we will show that $\widehat{S}_j$ is an optimal solution to the local problem at node j. Since the local problem at root is identical to problem (3), the desired result follows. If j is a leaf node, then $N_j = \{j\}$, $R_j(\{j\}) = r_j$, $V_j(\{j\}) = v_j$, and $R_j(\varnothing) = V_j(\varnothing) = 0$. So,

$$\begin{aligned} \arg\max_{S_j \subseteq N_j} V_j(S_j)\left(R_j(S_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) &= \arg\max \left\{ v_j \left(r_j - \mathtt{t}^*_{\mathsf{Parent}(j)} \right) \ , \ 0 \right\} \\ &= \begin{cases} \{j\} & \text{if } r_j \geq \mathtt{t}^*_{\mathsf{Parent}(j)} \\ \varnothing & \text{if } r_j < \mathtt{t}^*_{\mathsf{Parent}(j)} \end{cases} \quad = \quad \widehat{S}_j \ , \end{aligned}$$

establishing the result for a leaf node j. Now, suppose that the result is true for all nodes at depth h and consider node j at depth $h-1$. By the induction hypothesis, for all $k \in \mathsf{Children}(j)$, $\widehat{S}_k$ is an optimal solution to the local problem at node k. So, by Theorem 4.2, $\dot\bigcup_{k\in \mathsf{Children}(j)} \widehat{S}_k$ is an optimal solution to the local problem at node j. To complete the proof, note that

$$\dot{\bigcup_{k\in \mathsf{Children}(j)}} \widehat{S}_k = \dot{\bigcup_{k\in \mathsf{Children}(j)}} \left\{\ell \in N_k : r_\ell \geq \mathtt{t}^*_{\mathsf{Parent}(\ell)}\right\} = \left\{\ell \in N_j : r_\ell \geq \mathtt{t}^*_{\mathsf{Parent}(\ell)}\right\} = \widehat{S}_j \ ,$$

where the second equality uses the fact that $N_j = \dot\bigcup_{k\in \mathsf{Children}(j)} N_k$. □

By the above result, if product ℓ is in the optimal assortment, then any larger-revenue product sharing the same parent with product ℓ is also in the optimal assortment. As shown below, for a tree with one or two levels, this result recovers the optimality of revenue ordered assortments found in the existing literature for the multinomial and two-level nested logit models.

Example 4.4 (Multinomial Logit)**.** The multinomial logit model with n products, indexed by $\{1,2,\ldots,n\}$, corresponds to a one-level nested logit model, with $n+1$ leaf nodes connected to root. The first n of these nodes correspond to individual products and the last one corresponds to the no-purchase option. Since these is only one level in the tree, each leaf node has root as its parent. So, it follows from Corollary 4.3 that an optimal assortment is given by

$$S^* = \left\{\ell \in \{1,2,\ldots,n\} : r_\ell \geq \mathtt{t}^*_{\mathsf{Parent}(\ell)}\right\} = \{\ell \in \{1,2,\ldots,n\} : r_\ell \geq Z^*\} \ ,$$

where the second equality follows because $\mathtt{t}^*_{\mathsf{Parent}(\ell)} = \mathtt{t}^*_{\mathsf{root}} = Z^*$. The last expression above shows that if a particular product is in the optimal assortment, then any product with a larger revenue is in the optimal assortment as well.

Example 4.5 (Two-Level Nested Logit)**.** In the two-level nested logit model, the set of n products $\{1,2,\ldots,n\} = \dot\bigcup_{i=1}^m N_i$ is a disjoint union of m sets. The products in N_i are said to be in nest i. In this case, the associated tree has $m+1$ first-level nodes. The last one of these first-level nodes corresponds to the no-purchase option. For the other m first-level nodes, each first-level node i has $|N_i|$ children. Using Corollary 4.3, we have

$$S^* = \left\{\ell \in \{1,\ldots,n\} : r_\ell \geq \mathtt{t}^*_{\mathsf{Parent}(\ell)}\right\} \ = \ \dot{\bigcup_{i=1}^{m}} \left\{\ell \in N_i : r_\ell \geq \mathtt{t}^*_{\mathsf{Parent}(\ell)}\right\} \ = \ \dot{\bigcup_{i=1}^{m}} \left\{\ell \in N_i : r_\ell \geq \mathtt{t}^*_i\right\} \ ,$$

where the last equality uses the fact that the node corresponding to each product ℓ in nest i has the first-level node i as its parent. The last expression above shows that if a product in nest i is included in the optimal assortment, then any product in this nest with a larger revenue is included in the optimal assortment as well. This characterization of the optimal assortment for the two-level nested logit model was proved in Davis et al. (2014).

5. An Algorithm for Assortment Optimization

In this section, we use the characterization of an optimal assortment given in Section 4 to develop an algorithm that can be used to solve the assortment optimization problem in (3). For any set X, let 2^X denote the collection of all subsets of X. To find an optimal assortment, for each node j, we will construct a collection of subsets $\mathcal{A}_j \subseteq 2^{N_j}$ such that $\mathcal{A}_j$ includes an optimal solution to the local problem at node j. As pointed out in Section 4, the local problem at root is identical to the assortment optimization problem in (3). Thus, the collection $\mathcal{A}_{\mathsf{root}}$ includes an optimal solution to problem (3) and if the collection $\mathcal{A}_{\mathsf{root}}$ is reasonably small, then we can check the expected revenue of each subset in $\mathcal{A}_{\mathsf{root}}$ to find the optimal assortment.

To construct the collection $\mathcal{A}_j$ for each node j, we start from the leaf nodes and move up the tree by going over the nodes in a breadth-first manner. If ℓ is a leaf node, then $N_\ell = \{\ell\}$, so that the optimal solution to the local problem at node ℓ is either $\{\ell\}$ or $\varnothing$. Thus, if we let $\mathcal{A}_\ell = \{\{\ell\}, \varnothing\}$ for each leaf node ℓ, then the collection $\mathcal{A}_\ell$ includes an optimal solution to the local problem at node ℓ. Now, consider a non-leaf node j and assume that for each node $k \in \mathsf{Children}(j)$, we already have a collection $\mathcal{A}_k$ that includes an optimal solution to the local problem at node k. To construct a collection $\mathcal{A}_j$ that includes an optimal solution to the local problem at node j, for each $k \in \mathsf{Children}(j)$ and $u \in \mathbb{R}$, we let $\widehat{S}_k(u)$ be an optimal solution to the problem

$$\max_{S_k \in \mathcal{A}_k} V_k(S_k)\,(R_k(S_k) - u)\ . \tag{5}$$

Note that problem (5) considers only the subsets in the collection $\mathcal{A}_k$. We claim that the collection $\left\{\bigcup_{k\in \mathsf{Children}(j)} \widehat{S}_k(u) : u \in \mathbb{R}\right\}$ includes an optimal solution to the local problem at node j.

Claim 1: Let $\mathcal{A}_j = \left\{\bigcup_{k\in \mathsf{Children}(j)} \widehat{S}_k(u) : u \in \mathbb{R}\right\}$. Then, the collection of subsets $\mathcal{A}_j$ includes an optimal solution to the local problem at node j given in problem (4).

The objective function of problem (5) with $u = \mathtt{t}^*_{\mathsf{Parent}(k)}$ is identical to the objective function of the local problem at node k. Furthermore, by our hypothesis, $\mathcal{A}_k$ includes an optimal solution to the local problem at node k. Thus, if we solve problem (5) with $u = \mathtt{t}^*_{\mathsf{Parent}(k)}$, then the optimal solution $\widehat{S}_k(\mathtt{t}^*_{\mathsf{Parent}(k)})$ is an optimal solution to the local problem at node k. In this case, Theorem 4.2 implies that the assortment $\bigcup_{k\in \mathsf{Children}(j)} \widehat{S}_k(\mathtt{t}^*_{\mathsf{Parent}(k)})$ is an optimal solution to the local problem at node j. Since $\bigcup_{k\in \mathsf{Children}(j)} \widehat{S}_k(\mathtt{t}^*_{\mathsf{Parent}(k)}) \in \mathcal{A}_j$, the collection $\mathcal{A}_j$ includes an optimal solution to the local problem at node j, establishing the claim.

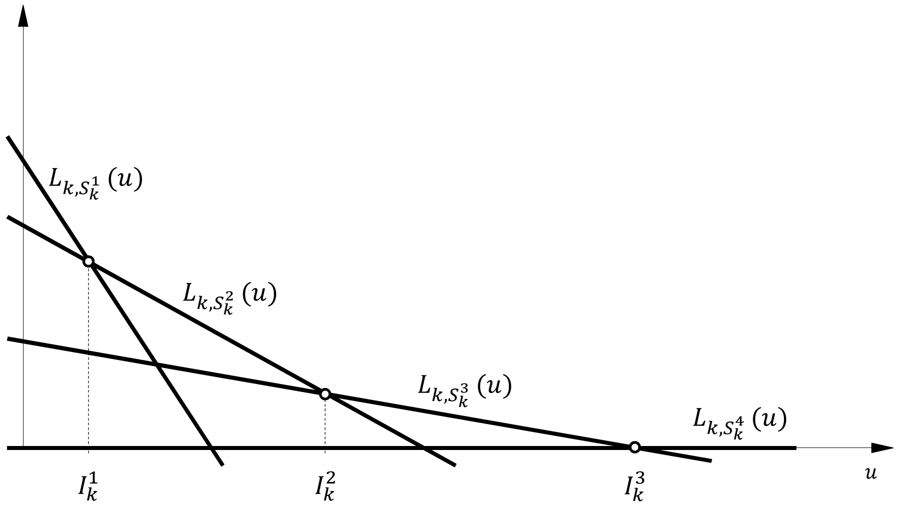


Figure 2: The lines $\{L_{k,S_k}(\cdot) : S_k \in \mathcal{A}_k\}$ and the points $\{I_k^q : q = 0, \ldots, Q_k\}$ for some collection $\mathcal{A}_k = \{S_k^1, S_k^2, S_k^3, S_k^4\}$.

Next, we show that all of the subsets in the collection $\mathcal{A}_j = \left\{ \bigcup_{k \in \mathsf{Children}(j)} \widehat{S}_k(u) : u \in \mathbb{R} \right\}$ can be constructed in a tractable fashion.

Claim 2: We have $|\mathcal{A}_j| \le \sum_{k \in \mathsf{Children}(j)} |\mathcal{A}_k|$, and all of the subsets in the collection $\mathcal{A}_j$ can be computed in $O\left(\sum_{k \in \mathsf{Children}(j)} |\mathcal{A}_k| \log |\mathcal{A}_k| \right)$ operations.

For each $S_k \in \mathcal{A}_k$, let $L_{k,S_k}(u) = V_k(S_k)\,(R_k(S_k) - u)$, in which case, problem (5) can be written as $\max_{S_k \in \mathcal{A}_k} L_{k,S_k}(u)$. Since $L_{k,S_k}(u)$ is a linear function of u, with slope $-V_k(S_k)$ and y-intercept $V_k(S_k) R_k(S_k)$, problem (5) reduces to finding the highest line at point u among the lines $\{L_{k,S_k}(\cdot) : S_k \in \mathcal{A}_k\}$. By checking the *pairwise intersection points* of the lines $\{L_{k,S_k}(\cdot) : S_k \in \mathcal{A}_k\}$, we can find points $-\infty = I_k^0 \le I_k^1 \le \ldots \le I_k^{Q_k - 1} \le I_k^{Q_k} = \infty$ with $Q_k = |\mathcal{A}_k|$ such that the highest one of the lines $\{L_{k,S_k}(\cdot) : S_k \in \mathcal{A}_k\}$ does not change as long as u takes values in one of the intervals $\{[I_k^{q-1}, I_k^q] : q = 1, \ldots, Q_k\}$. Figure 2 shows an example of the lines $\{L_{k,S_k}(\cdot) : S_k \in \mathcal{A}_k\}$ for some collection $\mathcal{A}_k = \{S_k^1, S_k^2, S_k^3, S_k^4\}$. The solid lines in this figure correspond to the lines in $\{L_{k,S_k}(\cdot) : S_k \in \mathcal{A}_k\}$ and the circles correspond to the points in $\{I_k^q : q = 0, \ldots, Q_k\}$, with $Q_k = 4$, $I_k^0 = -\infty$ and $I_k^4 = \infty$. Thus, the discussion in this paragraph shows that we can construct Q_k intervals $\{[I_k^{q-1}, I_k^q] : q = 1, \ldots, Q_k\}$ such that these intervals partition the real line and the optimal solution to problem (5) does not change when u takes values in one of these intervals. It is a standard result in analysis of algorithms that we can compute the intersection points $\{I_k^q : q = 0, \ldots, Q_k\}$ using $O(Q_k \log Q_k) = O(|\mathcal{A}_k| \log |\mathcal{A}_k|)$ operations; see Kleinberg and Tardos (2005).

Following the approach described in the above paragraph, we can construct the collection of points $\{I_k^q : q = 0, \ldots, Q_k\}$ for all $k \in \mathsf{Children}(j)$. Putting these collections together, we obtain the collection of points $\bigcup_{k\in \mathsf{Children}(j)}\{I_k^q : q = 0,\dots,Q_k\}$. Note that $I_k^0 = -\infty$ and $I_k^{Q_k} = \infty$ for all $k \in \mathsf{Children}(j)$. Ordering the points in this collection in increasing order and removing duplicates at $-\infty$ and ∞, we obtain the collection of points $\{I_j^q : q = 0,\dots,Q_j\}$, with $-\infty = I_j^0 \le I_j^1 \le \dots \le I_j^{Q_j-1} \le I_j^{Q_j} = \infty$. Note that $Q_j \le 1 + \sum_{k\in \mathsf{Children}(j)}(Q_k - 1) \le \sum_{k\in \mathsf{Children}(j)} Q_k$. The points $\{I_j^q : q = 0,\dots,Q_j\}$ will form the basis for constructing the collection of subsets $\mathcal{A}_j$.

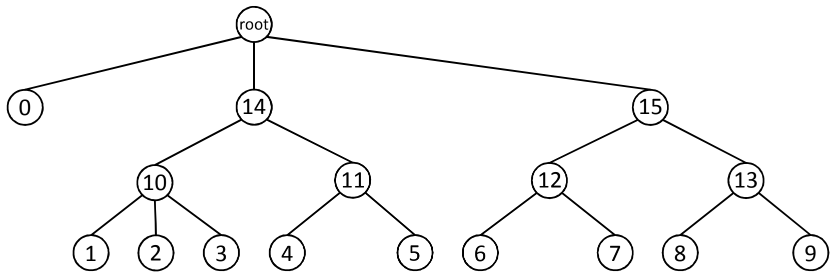
We make a crucial observation. Consider an arbitrary interval $[I_j^{q-1}, I_j^q]$, with $1 \le q \le Q_j$. By our construction of the points $\{I_j^q : q = 0,\dots,Q_j\}$ in the above paragraph, for all $k \in \mathsf{Children}(j)$, there exists some index $1 \le \zeta_k \le Q_k$ such that $[I_j^{q-1}, I_j^q] \subseteq [I_k^{\zeta_k-1}, I_k^{\zeta_k}]$. On the other hand, we recall that the optimal solution to problem (5) does not change as u takes values in one of the intervals $\{[I_k^{q-1}, I_k^q] : q = 1,\dots,Q_k\}$. Therefore, since $[I_j^{q-1}, I_j^q] \subseteq [I_k^{\zeta_k-1}, I_k^{\zeta_k}]$, the optimal solution to problem (5) does not change when u takes values in the interval $[I_j^{q-1}, I_j^q]$. In other words, the collection of subsets $\left\{\widehat{S}_k(u) \ : \ u \in [I_j^{q-1}, I_j^q]\right\}$ can be represented by $\underline{\textit{just a single subset}}$, which implies that the collection of subsets $\left\{\dot\bigcup_{k\in \mathsf{Children}(j)} \widehat{S}_k(u) \ : \ u \in [I_j^{q-1}, I_j^q]\right\}$ can also be represented by a single subset. Let $\widehat{S}_j^q$ denote the single subset that represents the collection of subsets $\left\{\dot\bigcup_{k\in \mathsf{Children}(j)} \widehat{S}_k(u) \ : \ u \in [I_j^{q-1}, I_j^q]\right\}$. In this case, we obtain

$$\begin{aligned}\mathcal{A}_j &= \left\{\dot\bigcup_{k\in \mathsf{Children}(j)} \widehat{S}_k(u) : u \in \mathbb{R}\right\} \\ &= \dot\bigcup_{q=1}^{Q_j} \left\{\dot\bigcup_{k\in \mathsf{Children}(j)} \widehat{S}_k(u) : u \in [I_j^{q-1}, I_j^q]\right\} = \left\{\widehat{S}_j^q : q = 1,2,\dots,Q_j\right\}. \qquad (6)\end{aligned}$$

Thus, the number of subsets in the collection $\mathcal{A}_j$ satisfies $|\mathcal{A}_j| = Q_j \le \sum_{k\in \mathsf{Children}(j)} Q_k = \sum_{k\in \mathsf{Children}(j)} |\mathcal{A}_k|$. Finally, the number of operations required to compute the collection $\mathcal{A}_j$ is equal to the running time for computing the intersection points $\{I_k^q : q = 0,\dots,Q_k\}$ for each $k \in \mathsf{Children}(j)$, giving the total running time of $O\left(\sum_{k\in \mathsf{Children}(j)} |\mathcal{A}_k| \log |\mathcal{A}_k|\right)$. The following lemma summarizes the key findings from the above discussion.

Lemma 5.1. *Consider an arbitrary non-leaf node $j \in \mathsf{V}$. For each $k \in \mathsf{Children}(j)$, suppose there exists a collection of subsets $\mathcal{A}_k \subseteq 2^{N_k}$ such that $\mathcal{A}_k$ contains an optimal solution to the local problem at node k. Then, we can construct a collection of subsets $\mathcal{A}_j \subseteq 2^{N_j}$ such that $\mathcal{A}_j$ includes an optimal solution to the local problem at node j with $|\mathcal{A}_j| \le \sum_{k\in \mathsf{Children}(j)} |\mathcal{A}_k|$. Moreover, this construction requires $O\left(\sum_{k\in \mathsf{Children}(j)} |\mathcal{A}_k| \log |\mathcal{A}_k|\right)$ operations.*

To obtain an optimal assortment, we iteratively apply Lemma 5.1 starting at the leaf nodes. For each leaf node ℓ, letting $\mathcal{A}_\ell = \{\{\ell\}, \varnothing\}$ suffices. Then, using the above construction and going over the nodes in a breadth-first manner, we can construct the collection $\mathcal{A}_j$ for every node j in the tree. To find an optimal assortment, we check the expected revenue of each subset in the collection



j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	root
v_j	17	4	5	10	6	9	4	7	10	13	·	·	·	·	·	·	·
r_j	0	14.8	9	5	13.4	7.5	15	8	18	6	·	·	·	·	·	·	·
η_j	·	·	·	·	·	·	·	·	·	·	0.93	0.87	0.89	0.67	0.75	0.9	0

Figure 3: A three-level problem instance and its parameters.

$\mathcal{A}_{\mathsf{root}}$ and choose the subset that provides the largest expected revenue. The following theorem shows that all of this work can be done in $O(dn\log n)$ operations.

Theorem 5.2. *For each node $j \in \mathsf{V}$, we can construct a collection $\mathcal{A}_j \subseteq 2^{N_j}$ such that $|\mathcal{A}_j| \leq 2n$ and $\mathcal{A}_j$ includes an optimal solution to the local problem at node j. Moreover, constructing $\mathcal{A}_j$ for all $j \in \mathsf{V}$ requires $O(dn\log n)$ operations.*

As a function of $\{|\mathcal{A}_k| : k \in \mathsf{Children}(j)\}$, Lemma 5.1 bounds the number of subsets in the collection $\mathcal{A}_j$ and gives the number of operations required to construct the collection $\mathcal{A}_j$. We can show Theorem 5.2 by using these relationships iteratively. The details are in Online Appendix B.

6. Numerical Illustrations for Assortment Optimization

In this section, we demonstrate our assortment optimization algorithm on one small problem instance and report its practical performance when applied on large problem instances. For the demonstration purpose, we consider a three-level nested logit model with nine products indexed by $1, 2, \ldots, 9$, along with a no-purchase option. We index the nodes of the tree by $\{0, 1, \ldots, 15\} \cup \{\mathsf{root}\}$, where the nodes $\{0, 1, \ldots, 9\}$ correspond to leaf nodes and the rest are non-leaf nodes. The top portion of Figure 3 shows the organization of the tree, whereas the bottom portion shows the model parameters. The parameters r_j and v_j are given for each leaf node j, whereas the parameter η_j is given for each non-leaf node j.

For this problem instance, if we let $\mathcal{A}_{10} = \{\{1, 2, 3\}, \{1, 2\}, \{1\}, \varnothing\}$, then one can verify that this collection includes an optimal solution to the local problem at node 10. Similarly, if we let $\mathcal{A}_{11} = \{\{4, 5\}, \{4\}, \varnothing\}$, then this collection includes an optimal solution to the local problem at node 11. By using the discussion in Section 5, we will construct a collection $\mathcal{A}_{14}$ such that this collection

S_{10}	$\{1,2,3\}$	$\{1,2\}$	$\{1\}$	$\varnothing$
$V_{10}(S_{10})$	15.46	7.72	3.63	0
$V_{10}(S_{10})\,R_{10}(S_{10})$	125.48	89.35	53.73	0
Optimal in Interval	$[-\infty, 4.67]$	$[4.67, 8.72]$	$[8.72, 14.80]$	$[14.80, \infty]$

S_{11}	$\{4,5\}$	$\{4\}$	$\varnothing$
$V_{11}(S_{11})$	10.55	4.75	0
$V_{11}(S_{11})\,R_{11}(S_{11})$	104.01	63.69	0
Optimal in Interval	$[-\infty, 6.96]$	$[6.96, 13.40]$	$[13.40, \infty]$

u	$[-\infty, 4.67]$	$[4.67, 6.96]$	$[6.96, 8.72]$	$[8.72, 13.40]$	$[13.40, 14.80]$	$[14.8, \infty]$
$\widehat{S}_{10}(u) \cup \widehat{S}_{11}(u)$	$\{1,2,3,4,5\}$	$\{1,2,4,5\}$	$\{1,2,4\}$	$\{1,4\}$	$\{1\}$	$\varnothing$

Table 1: Computation of $\mathcal{A}_{14}$. The top two portions show the solution to problem (5) for $k = 10$ and $k = 11$, respectively, for every value of $u \in \mathbb{R}$. The bottom portion lists each subset in $\mathcal{A}_{14}$.

includes an optimal solution to the local problem at node 14. To that end, for each $S_{10} \in \mathcal{A}_{10}$, we consider the line $L_{10,S_{10}}(u) = V_{10}(S_{10})\,(R_{10}(S_{10}) - u)$ with the slope $-V_{10}(S_{10})$ and y-intercept $V_{10}(S_{10}) \times R_{10}(S_{10})$. The second and third rows in the top portion of Table 1 list the slopes and intercepts for the lines corresponding to the four assortments in the collection $\mathcal{A}_{10}$. Finding the pairwise intersection points of these four lines, we determine that if $u \in [-\infty, 4.67]$, then the highest one of these lines is the one corresponding to the subset $\{1,2,3\}$. Similarly, if $u \in [4.67, 8.72]$, then the highest line is the one corresponding to the subset $\{1,2\}$. In other words, following the notation in Section 5, if $u \in [-\infty, 4.67]$, then the optimal solution to problem (5) for $k = 10$ is given by $\widehat{S}_{10}(u) = \{1,2,3\}$. Similarly, if $u \in [4.67, 8.72]$, then $\widehat{S}_{10}(u) = \{1,2\}$. The last row in the top portion of Table 1 lists the intervals such that if u belongs to one of these intervals, then $\widehat{S}_{10}(u)$ takes the value in the first row. Thus, the points $\{I_{10}^q : q = 0, 1, \ldots, 4\}$ are given by $\{-\infty, 4.67, 8.72, 14.80, \infty\}$. The middle portion of Table 1 focuses on the lines corresponding to the assortments in the collection $\mathcal{A}_{11}$. The formats of the top and middle portions of the table are the same. Thus, the points $\{I_{11}^q : q = 0, 1, \ldots, 3\}$ are given by $\{-\infty, 6.96, 13.40, \infty\}$. Taking the union of these two sets of points, we obtain $\{-\infty, 4.67, 6.96, 8.72, 13.40, 14.80, \infty\}$. In this case, we note that if $u \in [-\infty, 4.67]$, then the optimal solution to problem (5) for $k = 10$ is given by $\widehat{S}_{10}(u) = \{1,2,3\}$, whereas the optimal solution to problem (5) for $k = 11$ is given by $\widehat{S}_{11}(u) = \{4,5\}$. Thus, if $u \in [-\infty, 4.67]$, then $\bigcup_{k \in \mathsf{Children}(14)} \widehat{S}_k(u) = \{1,2,3,4,5\}$. Similarly, if $u \in [4.67, 6.96]$, then the optimal solution to problem (5) for $k = 10$ is given by $\widehat{S}_{10}(u) = \{1,2\}$, whereas the optimal solution to problem (5) for $k = 11$ is given by $\widehat{S}_{11}(u) = \{4,5\}$, which implies that if $u \in [4.67, 6.96]$, then $\bigcup_{k \in \mathsf{Children}(14)} \widehat{S}_k(u) = \{1,2,4,5\}$. Following this line of reasoning, the bottom portion of Table 1 lists $\bigcup_{k \in \mathsf{Children}(14)} \widehat{S}_k(u)$ for every value of $u \in \mathbb{R}$. Noting (6), we have $\mathcal{A}_{14} = \left\{ \bigcup_{k \in \mathsf{Children}(14)} \widehat{S}_k(u) : u \in \mathbb{R} \right\} = \{\{1,2,3,4,5\}, \{1,2,4,5\}, \{1,2,4\}, \{1,4\}, \{1\}, \varnothing\}$.

S_{12}	$\{6,7\}$	$\{6\}$	$\varnothing$
$V_{12}(S_{12})$	8.45	3.43	0
$V_{12}(S_{12})\,R_{12}(S_{12})$	89.11	51.51	0
Optimal in Interval	$[-\infty, 7.50]$	$[7.50, 15.00]$	$[15.00, \infty]$

S_{13}	$\{8,9\}$	$\{8\}$	$\varnothing$
$V_{13}(S_{13})$	8.17	4.68	0
$V_{13}(S_{13})\,R_{13}(S_{13})$	91.67	84.19	0
Optimal in Interval	$[-\infty, 2.14]$	$[2.14, 18.00]$	$[18.00, \infty]$

u	$[-\infty, 2.14]$	$[2.14, 7.50]$	$[7.50, 15.00]$	$[15.00, 18.00]$	$[18.00, \infty]$
$\widehat{S}_{12}(u) \cup \widehat{S}_{13}(u)$	$\{6,7,8,9\}$	$\{6,7,8\}$	$\{6,8\}$	$\{8\}$	$\varnothing$

Table 2: Computation of $\mathcal{A}_{15}$. The top two portions show the solution to problem (5) for $k = 12$ and $k = 13$, respectively, for every value of $u \in \mathbb{R}$. The bottom portion lists each subset in $\mathcal{A}_{15}$.

On the other hand, if we let $\mathcal{A}_{12} = \{\{6,7\}, \{6\}, \varnothing\}$, then one can verify that this collection includes an optimal solution to the local problem at node 12. Similarly, if we let $\mathcal{A}_{13} = \{\{8,9\}, \{8\}, \varnothing\}$, then this collection includes an optimal solution to the local problem at node 13. We use an argument similar to the one in the above paragraph to construct a collection $\mathcal{A}_{15}$ from the collections $\mathcal{A}_{12}$ and $\mathcal{A}_{13}$, such that the collection $\mathcal{A}_{15}$ includes an optimal solution to the local problem at node 15. Table 2 gives the details of our computations. The format of this table is identical to that of Table 1. The last row of the top portion shows that when u takes values in the intervals $[-\infty, 7.50]$, $[7.50, 15.00]$ and $[15.00, \infty]$, the optimal solution to problem (5) for $k = 12$ is, respectively, $\widehat{S}_{12}(u) = \{6,7\}$, $\widehat{S}_{12}(u) = \{6\}$ and $\widehat{S}_{12}(u) = \varnothing$. The last row of the middle portion shows that when u takes values in the intervals $[-\infty, 2.14]$, $[2.14, 18.00]$ and $[18.00, \infty]$, the optimal solution to problem (5) for $k = 13$ is, respectively, $\widehat{S}_{13}(u) = \{8,9\}$, $\widehat{S}_{13}(u) = \{8\}$ and $\widehat{S}_{13}(u) = \varnothing$. Collecting these results together, the last row of the bottom portion of Table 2, under the heading $\widehat{S}_{12}(u) \cup \widehat{S}_{13}(u)$, lists $\bigcup_{k \in \mathsf{Children}(15)} \widehat{S}_k(u)$ for every value of $u \in \mathbb{R}$. Thus, we have $\mathcal{A}_{15} = \{\{6,7,8,9\}, \{6,7,8\}, \{6,8\}, \{8\}, \varnothing\}$.

We have the collections $\mathcal{A}_{14}$ and $\mathcal{A}_{15}$ such that these collections include an optimal solution to the local problems at nodes 14 and 15, respectively. Finally, we use these collections to construct a collection $\mathcal{A}_{\mathsf{root}}$ such that this collection includes an optimal solution to the local problem at root. Our computations are given in Table 3 and they are similar to those in Tables 1 and 2. For example, the last row of the top portion shows that when u takes values in the intervals $[-\infty, 3.02]$ and $[3.02, 5.50]$, the optimal solution to problem (5) for $k = 14$ is, respectively, $\widehat{S}_{14}(u) = \{1,2,3,4,5\}$ and $\widehat{S}_{14}(u) = \{1,2,4,5\}$. The last row of the middle portion shows that when u takes values in the intervals $[-\infty, 1.05]$ and $[1.05, 6.69]$, the optimal solution to problem (5) for $k = 15$ is, respectively, $\widehat{S}_{15}(u) = \{6,7,8,9\}$ and $\widehat{S}_{15}(u) = \{6,7,8\}$. Collecting these results together, the second row of the bottom portion of Table 3, under the heading $\widehat{S}_{14}(u) \cup \widehat{S}_{15}(u)$, lists $\bigcup_{k \in \mathsf{Children}(\mathsf{root})} \widehat{S}_k(u)$ for every

S_{14}	$\{1,2,3,4,5\}$	$\{1,2,4,5\}$	$\{1,2,4\}$	$\{1,4\}$	$\{1\}$	$\varnothing$
$V_{14}(S_{14})$	11.52	8.84	6.64	4.93	2.63	0
$V_{14}(S_{14})\,R_{14}(S_{14})$	101.62	93.53	81.44	69.01	38.92	0
Highest in Interval	$[-\infty, 3.02]$	$[3.02, 5.50]$	$[5.50, 7.27]$	$[7.27, 13.10]$	$[13.10, 14.80]$	$[14.80, \infty]$

S_{15}	$\{6,7,8,9\}$	$\{6,7,8\}$	$\{6,8\}$	$\{8\}$	$\varnothing$
$V_{15}(S_{15})$	12.55	10.15	6.58	4.01	0
$V_{15}(S_{15})\,R_{15}(S_{15})$	136.48	133.96	110.08	72.16	0
Highest in Interval	$[-\infty, 1.05]$	$[1.05, 6.69]$	$[6.69, 14.75]$	$[14.75, 18.00]$	$[18.00, \infty]$

u	$[-\infty, 1.05]$	$[1.05, 3.02]$	$[3.02, 5.50]$	$[5.50, 6.69]$	$[6.69, 7.27]$
$\widehat{S}_{14}(u) \cup \widehat{S}_{15}(u)$	$\{1,2,3,4,5,6,7,8,9\}$	$\{1,2,3,4,5,6,7,8\}$	$\{1,2,4,5,6,7,8\}$	$\{1,2,4,6,7,8\}$	$\{1,2,4,6,8\}$
$R_{\mathsf{root}}(\widehat{S}_{14}(u) \cup \widehat{S}_{15}(u))$	5.80	6.09	6.32	**6.38**	6.34

u	$[7.27, 13.10]$	$[13.10, 14.75]$	$[14.75, 14.80]$	$[14.80, 18.00]$	$[18.00, \infty]$
$\widehat{S}_{14}(u) \cup \widehat{S}_{15}(u)$	$\{1,4,6,8\}$	$\{1,6,8\}$	$\{1,8\}$	$\{8\}$	$\varnothing$
$R_{\mathsf{root}}(\widehat{S}_{14}(u) \cup \widehat{S}_{15}(u))$	6.28	5.68	4.70	3.43	0.00

Table 3: Computation of $\mathcal{A}_{\mathsf{root}}$. The top two portions show the solution to problem (5) for $k = 14$ and $k = 15$, respectively, for every value of $u \in \mathbb{R}$. The bottom portion lists each subset in $\mathcal{A}_{\mathsf{root}}$.

value of $u \in \mathbb{R}$. These subsets form the collection $\mathcal{A}_{\mathsf{root}}$, which includes an optimal solution to the local problem at root. Since the local problem at root is equivalent to our assortment optimization problem, the collection $\mathcal{A}_{\mathsf{root}}$ includes an optimal assortment. The last row of the bottom portion of Table 3, under the heading $R_{\mathsf{root}}(\widehat{S}_{14}(u) \cup \widehat{S}(u)_{15})$, lists the expected revenue from each assortment in $\mathcal{A}_{\mathsf{root}}$ and the highest expected revenue is highlighted in bold. Therefore, the optimal assortment is $\{1, 2, 4, 6, 7, 8\}$ with an expected revenue of 6.38.

We have also tested the practical performance of our assortment optimization algorithm by applying it on larger test problems with two and three levels in the tree. To our knowledge, our algorithm is the only efficient approach for solving the assortment problem when there are three levels in the tree. For the test problems with three levels, we compare our assortment optimization algorithm with a full enumeration scheme that checks the expected revenue from every possible assortment. For the test problems with two levels, Davis et al. (2014) show that the optimal assortment can be obtained by solving a linear program. So, for two-level problems, we compare our assortment optimization algorithm with the full enumeration scheme and the linear programming approach in Davis et al. (2014). The largest test problems that we consider includes three levels in the tree and a total of 512 products. Our numerical experiments indicate that our approach provides the optimal assortment within about one to two seconds for these test problems. Details of our numerical experiments are given in Online Appendix C.

Finally, we have also considered a multi-period extension of the assortment optimization problem with capacity control. In this problem setting, a firm has limited availability of a resource. The firm offers sets of products to customers arriving over time and the sale of each product consumes the inventory of the resource. The goal of the firm is to decide which assortment of products to offer to customers over time to maximize the total expected revenue over a finite selling horizon. For this dynamic capacity control problem, we show that under the d-level nested logit model, a nested control policy is optimal. For more details, the reader is referred to Online Appendix D.

7. Price Optimization Problem

In this section, we study the problem of finding revenue-maximizing prices, assuming that the assortment of products is fixed at $\{1, 2, \ldots, n\}$. Our formulation of the price optimization problem is similar to that of the assortment optimization problem in Section 3. The taxonomy of the products is described by a d-level tree, denoted by $\mathsf{T} = (\mathsf{V}, \mathsf{E})$, which has n leaf nodes at depth d, corresponding to the n products in $\{1, 2, \ldots, n\}$. The no-purchase option is labeled as node 0, which is directly connected to root. Associated with each node j, we have a set of products $N_j \subseteq \{0, 1, \ldots, n\}$, which corresponds to the set of products included in the subtree that is rooted at node j. If j is a leaf node, then $N_j = \{j\}$ is a singleton, consisting of the product itself. On the other hand, if j is a non-leaf node, then $N_j = \dot{\bigcup}_{k \,\in\, \mathsf{Children}(j)} N_k$ is the disjoint union of the sets of products at the child nodes of j.

The decision variable is the price vector $\boldsymbol{p} = (p_1, \ldots, p_n) \in \mathbb{R}^n_+$, where p_ℓ denotes the price of product ℓ. By convention, the price of the no-purchase option is fixed at $p_0 = 0$. For each subset $X \subseteq \{0, 1, \ldots, n\}$, $\boldsymbol{p}_X = (p_\ell : \ell \in X)$ denotes the vector of prices of the products in X. The choice process of the customer is similar to the one in Section 3. Given the price vector $\boldsymbol{p} \in \mathbb{R}^n_+$, a customer associates the preference weight $V_j(\boldsymbol{p}_{N_j})$ with each node $j \in \mathsf{V}$. The preference weight of each node is computed as follows. If j is a leaf node, then we have $V_j(\boldsymbol{p}_{N_j}) = e^{\alpha_j - \beta_j\, p_j}$, where α_j and $\beta_j > 0$ are parameters of the d-level nested logit model associated with product j. Noting that the price of the no-purchase option is fixed at zero, the preference weight of the no-purchase option is $V_0(\boldsymbol{p}_{N_0}) = e^{\alpha_0}$. If, on the other hand, j is a non-leaf node, then we have

$$V_j\left(\boldsymbol{p}_{N_j}\right) = \left(\sum_{k \,\in\, \mathsf{Children}(j)} V_k(\boldsymbol{p}_{N_k}) \right)^{\eta_j} .$$

During the course of her choice process, if the customer is at node j, then she follows node $k \in \mathsf{Children}(j)$ with probability $V_k(\boldsymbol{p}_{N_k}) / \sum_{\ell \in \mathsf{Children}(j)} V_\ell(\boldsymbol{p}_{N_\ell})$.

We use $R_j(\boldsymbol{p}_{N_j})$ to denote the expected revenue obtained from a customer at node j. If j is a leaf node, then $R_j(\boldsymbol{p}_{N_j}) = p_j$. If, however, j is a non-leaf node, then we have

$$R_j\left(\boldsymbol{p}_{N_j}\right) = \frac{\sum_{k \,\in\, \mathsf{Children}(j)} V_k(\boldsymbol{p}_{N_k}) R_k(\boldsymbol{p}_{N_k})}{\sum_{k \,\in\, \mathsf{Children}(j)} V_k(\boldsymbol{p}_{N_k})} .$$

To facilitate our exposition, when it is clear from context, we will simply write $V_j(\boldsymbol{p})$ and $R_j(\boldsymbol{p})$ to denote $V_j(\boldsymbol{p}_{N_j})$ and $R_j(\boldsymbol{p}_{N_j})$, respectively. The expected revenue associated with the price vector $\boldsymbol{p}$ corresponds to the expected revenue at root, $R_{\mathsf{root}}(\boldsymbol{p})$. Thus, we are interested in the price optimization problem given by

$$\max_{\boldsymbol{p} \in \mathbb{R}^n_+} \quad R_{\mathsf{root}}(\boldsymbol{p}) \ .$$

For the two-level nested logit model, Gallego and Wang (2013) show that the expected revenue function $R_{\mathsf{root}}(\cdot)$ is not concave and can have multiple local maxima. Thus, we cannot hope to find the global maximum in general. Instead, we will focus on finding a stationary point of the expected revenue function $R_{\mathsf{root}}(\cdot)$; that is, the prices at which the gradient of the expected revenue function is zero. We begin by giving a succinct characterization of a stationary point. Next, we develop an iterative algorithm that generates a sequence of prices converging to a stationary point of the expected revenue function. Our algorithm is different from the traditional gradient ascent method because it moves in a direction that is different from the gradient. Furthermore, our algorithm avoids the problem of choosing a step size. As shown in our numerical experiments in Section 8, our algorithm converges to a stationary point much faster than gradient-based methods.

7.1 Characterization of a Stationary Point

For each $\boldsymbol{p} \in \mathbb{R}^n_+$, let a collection of scalars $(\mathtt{u}_j(\boldsymbol{p}) : j \in \mathsf{V})$ be defined as

$$\mathtt{u}_j(\boldsymbol{p}) = \eta_j \mathtt{u}_{\mathsf{Parent}(j)}(\boldsymbol{p}) + (1 - \eta_j) R_j(\boldsymbol{p}) \ ,$$

where we set $\mathtt{u}_{\mathsf{root}}(\boldsymbol{p}) = R_{\mathsf{root}}(\boldsymbol{p})$. Starting at the root node, we can compute the scalars $(\mathtt{u}_j(\boldsymbol{p}) : j \in \mathsf{V})$ by traversing the tree in a breadth-first manner. The following lemma shows how to compute the gradient of the expected revenue function using the scalars $(\mathtt{u}_j(\boldsymbol{p}) : j \in \mathsf{V})$. In this lemma and throughout the rest of the paper, we use $\theta_\ell(\boldsymbol{p})$ to denote the probability that a customer chooses product ℓ under the price vector $\boldsymbol{p}$. As mentioned in Section 7, if the customer is at node j, then she follows node $k \in \mathsf{Children}(j)$ with probability $V_k(\boldsymbol{p}) / \sum_{\ell \in \mathsf{Children}(j)} V_\ell(\boldsymbol{p})$. Thus, $\theta_\ell(\boldsymbol{p})$ is given by

$$\theta_\ell(\boldsymbol{p}) = \prod_{h=1}^{d} \frac{V_{\mathsf{An}(\ell,h)}(\boldsymbol{p})}{\sum_{k \in \mathsf{Sibling}(\mathsf{An}(\ell,h))} V_k(\boldsymbol{p})} \ ,$$

where $\mathsf{An}(\ell, h)$ denotes the ancestor of the leaf node ℓ in level h, with $\mathsf{An}(\ell, d) = \ell$ and $\mathsf{An}(\ell, 0) = \mathsf{root}$. Furthermore, for each node j other than root, $\mathsf{Sibling}(j) = \mathsf{Children}(\mathsf{Parent}(j))$ is the set of nodes that are siblings of j; that is, the set of nodes with the same parent as node j and this set includes node j itself.

Lemma 7.1 (Gradient of Expected Revenue)**.** *For each $\boldsymbol{p} \in \mathbb{R}^n_+$ and for every leaf node ℓ,*

$$\frac{\partial R_{\mathsf{root}}}{\partial p_\ell}(\boldsymbol{p}) = -\theta_\ell(\boldsymbol{p})\,\beta_\ell\left(p_\ell - \frac{1}{\beta_\ell} - \mathsf{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p})\right) ,$$

where $\theta_\ell(\boldsymbol{p})$ is the probability that a customer chooses product ℓ under the price vector $\boldsymbol{p}$.

We give the proof of this lemma in Online Appendix E. By using the above lemma, we can efficiently obtain the gradient of the expected revenue at any price vector $\boldsymbol{p} \in \mathbb{R}^n_+$. In particular, we can compute the preference weight $V_j(\boldsymbol{p})$ and the expected revenue $R_j(\boldsymbol{p})$ for all $j \in \mathsf{V}$, by starting with the leaf nodes and moving up the tree in a breadth-first manner. Once we compute the preference weights, it is straightforward to compute $\theta_\ell(\boldsymbol{p})$ for each leaf node ℓ. Similarly, by using $(R_j(\boldsymbol{p}) : j \in \mathsf{V})$, we can compute $(\mathsf{u}_j(\boldsymbol{p}) : j \in \mathsf{V})$ by starting from the root node and moving down the tree in a breadth-first manner. Once we know $\theta_\ell(\boldsymbol{p})$ and $\mathsf{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p})$ for each leaf node ℓ, the above lemma gives a closed-form expression for the gradient.

Lemma 7.1 allows us to apply the traditional gradient ascent method to compute a stationary point of the expected revenue function; see Sun and Yuan (2006). However, in our numerical experiments in Section 8, we observe that the traditional gradient ascent method can be relatively slow in practice and requires a careful selection of the step size. In the rest of this section, we develop a new algorithm for computing a stationary point of the expected revenue function. This algorithm is different from the gradient ascent method and completely avoids the problem of step size selection. When developing our algorithm, we make use of a succinct characterization of the stationary points of the expected revenue function, given by the following corollary. The proof of this corollary immediately follows from the above lemma and the fact that the probability of choosing each product is always positive.

Corollary 7.2 (A Necessary and Sufficient Condition for a Stationary Point)**.** *For each $\boldsymbol{p} \in \mathbb{R}^n_+$, $\boldsymbol{p}$ is a stationary point of $R_{\mathsf{root}}(\cdot)$ if and only if $p_\ell = \frac{1}{\beta_\ell} + \mathsf{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p})$ for all ℓ.*

7.2 A Pricing Algorithm without Gradients

By Corollary 7.2, if we can find a price vector $\boldsymbol{p} \in \mathbb{R}^n_+$ such that $p_\ell = \frac{1}{\beta_\ell} + \mathsf{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p})$ for all ℓ, then $\boldsymbol{p}$ corresponds to a stationary point of the expected revenue function. This observation yields the following iterative idea to find a stationary point of the expected revenue function. We start with a price vector $\boldsymbol{p}^0 \in \mathbb{R}^n_+$. In iteration s, we compute $(\mathsf{u}_j(\boldsymbol{p}^s) : j \in \mathsf{V})$. Using this collection of scalars, we compute the price vector $\boldsymbol{p}^{s+1}$ at the next iteration as $p^{s+1}_\ell = \frac{1}{\beta_\ell} + \mathsf{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p}^s)$ for each leaf node ℓ. It turns out that a modification of this idea generates a sequence of price vectors

$\{\boldsymbol{p}^s \in \mathbb{R}^n_+ : s = 0, 1, 2, \ldots\}$ that converges to a stationary point of the expected revenue function and it forms the basis of our pricing algorithm. We give our pricing algorithm below, which we call the Push-Up-then-Push-Down (PUPD) algorithm.

Push-Up-then-Push-Down (PUPD):

`Step 0.` Set the iteration counter $s = 0$ and $\boldsymbol{p}^0 = \boldsymbol{0}$.

`Step 1.` Compute the scalars $(\mathtt{t}^s_j : j \in \mathsf{V})$ recursively over the tree as follows. For root, we have $\mathtt{t}^s_{\mathsf{root}} = R_{\mathsf{root}}(\boldsymbol{p}^s)$. Using $R^s_j = R_j(\boldsymbol{p}^s)$ to denote the expected revenue at node j under the price vector $\boldsymbol{p}^s$, for the other nodes, we have

$$\mathtt{t}^s_j = \max\left\{\mathtt{t}^s_{\mathsf{Parent}(j)}\ ,\ \eta_j \mathtt{t}^s_{\mathsf{Parent}(j)} + (1-\eta_j) R^s_j\right\}\ .$$

`Step 2.` The price vector $\boldsymbol{p}^{s+1}$ at the next iteration is given by $p^{s+1}_\ell = \frac{1}{\beta_\ell} + \mathtt{t}^s_{\mathsf{Parent}(\ell)}$ for each leaf node ℓ. Increase s by one and go back to Step 1.

There are several stopping criteria to consider for the PUPD algorithm. For example, we can terminate the algorithm if $\|\boldsymbol{p}^{s+1} - \boldsymbol{p}^s\|_2$ is less than some pre-specified tolerance. In our numerical experiments in the next section that compare the PUPD algorithm with the gradient-based algorithm, we use a common criterion and stop the algorithms when the norm of the gradient of the expected revenue function $\|\nabla R_{\mathsf{root}}(\boldsymbol{p}^s)\|_2$ is less than some pre-specified tolerance.

Our algorithm is called Push-Up-then-Push-Down because given the price vector $\boldsymbol{p}^s$, we compute $R_{\mathsf{root}}(\boldsymbol{p}^s)$ by "pushing up," starting at the leaf nodes and computing the expected revenue at each node in the tree in a breadth-first manner, until we reach root. Once we obtain $(R_j(\boldsymbol{p}^s) : j \in \mathsf{V})$, we compute $(\mathtt{t}^s_j : j \in \mathsf{V})$ by "pushing down," starting from root and computing these scalars at each node in the tree in a breadth-first manner. Also, we note that our PUPD algorithm is different from the gradient ascent method because in each iteration s, the gradient of the expected revenue function at $\boldsymbol{p}^s$ is given by

$$\left.\frac{\partial R_{\mathsf{root}}}{\partial p_\ell}(\boldsymbol{p})\right|_{\boldsymbol{p}=\boldsymbol{p}^s} = -\theta_\ell(\boldsymbol{p}^s)\,\beta_\ell\left(p^s_\ell - \frac{1}{\beta_\ell} - \mathtt{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p}^s)\right)\ ,$$

whereas the change in the price vector in iteration s under our PUPD algorithm is given by $p^{s+1}_\ell - p^s_\ell = -(p^s_\ell - \frac{1}{\beta_\ell} - \mathtt{t}^s_{\mathsf{Parent}(\ell)})$. Thus, the direction of change $\boldsymbol{p}^{s+1} - \boldsymbol{p}^s$ under our PUPD algorithm is not necessarily parallel to the gradient of the expected revenue function at price vector $\boldsymbol{p}^s$. The following theorem gives our main result for the PUPD algorithm and it shows that the sequence of price vectors generated by our PUPD algorithm converges to a stationary point of the expected revenue function. The proof of this theorem is given in Online Appendix E.

Theorem 7.3 (Convergence to a Stationary Point)**.** *The sequence of prices under the* PUPD *algorithm converges to a stationary point of the expected revenue function* $R_{\mathsf{root}}(\cdot)$ *.*

7.3 Extension to Arbitrary Cost Functions

We can extend Lemma 7.1 to the case in which there is a cost associated with offering a product and this cost is a function of the market share of the product. In particular, we consider the problem

$$\max_{\boldsymbol{p}} \quad \Pi(\boldsymbol{p}) \equiv R_{\mathsf{root}}(\boldsymbol{p}) - \sum_{\ell=1}^{n} C_\ell(\theta_\ell(\boldsymbol{p})) \ , \tag{7}$$

where $C_\ell(\cdot)$ denotes the cost function associated with product ℓ and $\theta_\ell(\boldsymbol{p})$ denotes the probability that a customer chooses product ℓ under the price vector $\boldsymbol{p}$. The above model allows each product ℓ to involve a different cost $C_\ell(\cdot)$.

The main result of this section provides an expression for the gradient of the profit function $\Pi(\cdot)$. Before we proceed to the statement of the theorem, we introduce some notation. For any nodes i and j, let $\theta^i_j(\boldsymbol{p})$ denote the probability that we can get from node i to node j; that is,

$$\theta^i_j(\boldsymbol{p}) \quad = \quad \begin{cases} \displaystyle\prod_{h=1}^{m} \frac{V_{k_h}}{\sum_{u \in \mathsf{Sibling}(k_h)} V_u} & \text{if there is a (unique) path } (i = k_0, k_1, \ldots, k_m = j) \text{ from } i \text{ to } j \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\theta^{\mathsf{root}}_\ell(\boldsymbol{p})$ is exactly the same as $\theta_\ell(\boldsymbol{p})$ defined just before Lemma 7.1, and thus, we can write $\theta^{\mathsf{root}}_\ell(\boldsymbol{p})$ simply as $\theta_\ell(\boldsymbol{p})$, dropping the superscript root for convenience. For all $\boldsymbol{p} \in \mathbb{R}^n_+$ and for all leaf nodes i and ℓ, let $\mathsf{v}_{i,\ell}(\boldsymbol{p})$ be defined by:

$$\mathsf{v}_{i,\ell}(\boldsymbol{p}) \ \equiv \ \begin{cases} \sum_{s=0}^{\bar{s}_{i,\ell}} \frac{\prod_{h=s+1}^{d} \eta_{\mathsf{An}(\ell,h)}}{\theta^{\mathsf{root}}_{\mathsf{An}(\ell,s)}(\boldsymbol{p})} \left(1 - \eta_{\mathsf{An}(\ell,s)}\right) & \text{if } i \neq \ell \\ -\frac{1}{\theta^{\mathsf{root}}_\ell(\boldsymbol{p})} + \sum_{s=0}^{d-1} \frac{\prod_{h=s+1}^{d} \eta_{\mathsf{An}(\ell,h)}}{\theta^{\mathsf{root}}_{\mathsf{An}(\ell,s)}(\boldsymbol{p})} \left(1 - \eta_{\mathsf{An}(\ell,s)}\right) & \text{if } i = \ell, \end{cases}$$

where $\bar{s}_{i,\ell} = \max\{h : \mathsf{An}(i,h) = \mathsf{An}(\ell,h)\}$. Note that $\bar{s}_{i,\ell}$ is well-defined with $0 \leq \bar{s}_{i,\ell} \leq d$ because root at level 0 is a common ancestor of all nodes. Also, $\bar{s}_{i,\ell} = d$ if and only if $i = \ell$. Here is the main result of this section, whose proof is given in Online Appendix E.

Theorem 7.4 (Gradient for Arbitrary Cost Functions)**.** *For each $\boldsymbol{p} \in \mathbb{R}^n_+$ and for each leaf node ℓ,*

$$\frac{\partial \Pi}{\partial p_\ell}(\boldsymbol{p}) = -\theta^{\mathsf{root}}_\ell(\boldsymbol{p})\beta_\ell \left(p_\ell - \frac{1}{\beta_\ell} - \mathrm{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p}) + \sum_{i=1}^n C'_i(\theta^{\mathsf{root}}_i(\boldsymbol{p}))\, \theta^{\mathsf{root}}_i(\boldsymbol{p})\, \mathrm{v}_{i,\ell}(\boldsymbol{p}) \right) .$$

Theorem 7.4 holds for any d-level nested logit model with an arbitrary value for d. Since this theorem gives a tractable formula for the gradient of the expected profit with respect to the prices, we can use this formula to compute a stationary point of the profit function $\Pi(\cdot)$ through gradient-based methods. To our knowledge, this is the first such result. We observe that a stationary point $\boldsymbol{p}^*$ of the expected profit function $\Pi(\cdot)$ corresponds to a value of $\boldsymbol{p}$ that ensures that the value of the expression in the brackets in Theorem 7.4 is equal to zero. Thus, one can construct an analogue of the PUPD algorithm for the general expected profit function $\Pi(\cdot)$ as follows. Using $\boldsymbol{p}^s$ to denote the price vector at iteration s, we can compute $\mathrm{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p}^s)$, $\theta^{\mathsf{root}}_i(\boldsymbol{p}^s)$ and $\mathrm{v}_{i,\ell}(\boldsymbol{p}^s)$ at this price vector. In this case, the price vector $\boldsymbol{p}^{s+1}$ at the next iteration is given by $\frac{1}{\beta_\ell} + \mathrm{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p}^s) - \sum_{i=1}^n C'_i(\theta^{\mathsf{root}}_i(\boldsymbol{p}^s))\, \theta^{\mathsf{root}}_i(\boldsymbol{p}^s)\, \mathrm{v}_{i,\ell}(\boldsymbol{p}^s)$. Unfortunately, the dynamics of the price vector at successive iterations in this version of the PUPD algorithm are substantially more complicated than the dynamics of the PUPD algorithm given in Section 7.2. Thus, finding an analogue of the PUPD algorithm for the general expected profit function $\Pi(\cdot)$ is the subject of future research, but we note that the gradient expression in Theorem 7.4 does allow us to compute a stationary point of the expected profit function through gradient-based methods.

8. Numerical Results for Price Optimization

In this section, we give numerical experiments to test the performance of our PUPD algorithm. In particular, we compare our PUPD algorithm with the traditional gradient ascent method.

Numerical Setup: We use test problems with three levels in the tree. Letting m_h be the number of children of each node at depth h in the tree, we have a total of $m_0\, m_1\, m_2$ products. We vary $(m_0, m_1, m_2) \in \{2,4,6\} \times \{2,4,6\} \times \{2,4,6\}$. We refer to each (m_0, m_1, m_2) combination as a different problem class. We randomly generate 200 individual problem instances in each problem class. To generate each problem instance, we set $\alpha_0 = 0$ so that the preference weight of the no-purchase option is $e^{\alpha_0} = 1$. For each non-leaf node j, we sample the parameter η_j from the uniform distribution over $[0,1]$. The parameters α_j and β_j for each leaf node j are sampled from the uniform distributions over $[1,3]$ and $[2,3]$, respectively.

Throughout this section, we refer to our PUPD algorithm as PUPD and the gradient ascent method as GA. We implement PUPD precisely as described in Section 7.2. In our implementation of GA, we generate a sequence of price vectors $\{\boldsymbol{p}^s \in \mathbb{R}^n_+ : s = 0, 1, 2, \ldots\}$, where $\boldsymbol{p}^{s+1} = \boldsymbol{p}^s + t^s \nabla R_{\mathsf{root}}(\boldsymbol{p}^s)$, $\nabla R_{\mathsf{root}}(\boldsymbol{p}^s)$ is the gradient of the expected revenue function evaluated at $\boldsymbol{p}^s$ and t^s is a step size parameter. To choose the step size parameter t^s in iteration s, we find a local solution to the problem $\max_{t \geq 0} R_{\mathsf{root}}(\boldsymbol{p}^s + t \nabla R_{\mathsf{root}}(\boldsymbol{p}^s))$ by using golden section search; see Sun and Yuan (2006). We use the same stopping criterion for PUPD and GA, and we stop these algorithms when the norm of the gradient satisfies $\|\nabla R_{\mathsf{root}}(\boldsymbol{p}^s)\|_2 \leq 10^{-6}$.

Numerical Results: In Table 4, we compare the running times for PUPD and GA. The first three columns in Table 4 show the parameters of each problem class by giving (m_0, m_1, m_2). The fourth column shows the average number of iterations for PUPD to reach the stopping criterion, where the average is taken over all 200 problem instances in a problem class. Similarly, the fifth column shows the average number of iterations for GA to reach the stopping criterion. The next three columns give various statistics regarding the ratio between the numbers of iterations for GA and PUPD to reach the stopping criterion. To be specific, letting $\text{Itn}^{\text{GA}}(k)$ and $\text{Itn}^{\text{PUPD}}(k)$ respectively be the numbers of iterations for GA and PUPD to stop for problem instance k in a certain problem class, the sixth, seventh and eighth columns respectively give the average, minimum and maximum of the figures $\{\text{Itn}^{\text{GA}}(k)/\text{Itn}^{\text{PUPD}}(k) : k = 1, \ldots, 200\}$. Finally, the last three columns give various statistics regarding the ratio between the running times of GA and PUPD. In particular, letting $\text{Time}^{\text{GA}}(k)$ and $\text{Time}^{\text{PUPD}}(k)$ respectively be the running times for GA and PUPD to reach the stopping criterion for problem instance k in a certain problem class, the ninth, tenth and eleventh columns in the table respectively give the average, minimum and maximum of the figures $\{\text{Time}^{\text{GA}}(k)/\text{Time}^{\text{PUPD}}(k) : k = 1, \ldots, 200\}$.

The results in Table 4 indicate that PUPD takes drastically fewer iterations to reach the stopping criterion when compared with GA. The ratio between the number of iterations for GA and PUPD ranges between 2 to 514. This ratio never falls below one, showing that PUPD stops in fewer iterations than GA in all problem instances. We observe that as the problem size measured by the number of products $m_0\, m_1\, m_2$ increases, the number of iterations for GA to reach the stopping criterion tends to increase, whereas the number of iterations for PUPD remains stable and it even shows a slightly decreasing trend. These observations suggest the superiority of PUPD especially in larger problem instances. The last three columns in Table 4 show that, in addition to the number of iterations, the running times for PUPD are also substantially smaller than those of GA. GA takes at least twice as long as PUPD over all problem instances. There are problem instances, in which the running time for GA is 466 times larger than that of PUPD. The ratios of the running times never fall below one in any problem instance, indicating that PUPD always reaches the

Prb. Class			Avg. No. Itns		Ratio Between No. Itns.			Ratio Between Run. Times		
m_0	m_1	m_2	PUPD	GA	Avg.	Min	Max	Avg.	Min	Max
2	2	2	124	2,019	12	2	131	10	2	108
2	2	4	112	3,393	25	5	184	21	4	188
2	2	6	93	4,201	34	8	158	29	6	141
2	4	2	122	5,194	36	4	230	31	3	196
2	4	4	104	5,390	47	11	172	40	9	146
2	4	6	95	7,117	69	16	251	60	13	275
2	6	2	124	6,520	47	6	168	41	5	139
2	6	4	113	8,401	72	15	235	64	13	252
2	6	6	101	10,390	102	24	294	89	19	299
4	2	2	161	6,470	35	4	205	31	3	204
4	2	4	128	6,848	50	11	179	43	9	159
4	2	6	114	9,136	77	19	236	67	16	225
4	4	2	135	10,045	72	11	220	63	9	242
4	4	4	101	12,147	118	25	270	106	22	289
4	4	6	92	12,799	139	36	305	120	30	281
4	6	2	122	13,204	110	14	303	98	12	335
4	6	4	98	15,851	166	57	371	145	52	325
4	6	6	85	17,913	217	73	396	191	62	422
6	2	2	158	8,564	54	6	191	47	6	215
6	2	4	125	10,441	83	16	212	73	13	181
6	2	6	104	10,857	104	27	257	90	24	247
6	4	2	125	14,912	120	25	280	106	22	309
6	4	4	98	15,932	165	48	327	145	42	343
6	4	6	86	18,087	217	101	419	194	81	446
6	6	2	109	17,197	164	47	326	144	42	311
6	6	4	87	19,896	239	102	398	213	84	417
6	6	6	75	21,531	299	116	514	266	104	466

Table 4: Performance comparison between PUPD and GA.

stopping criterion faster than GA. This drastic difference between the performance of PUPD and GA can, at least partially, be explained by using the following intuitive argument. Noting Lemma 7.1, under GA, if the current price vector is $\boldsymbol{p}^s$, then the price of product ℓ changes in the direction $-\theta_\ell(\boldsymbol{p}^s)\,\beta_\ell\left(p_\ell^s - \frac{1}{\beta_\ell} - \mathsf{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p}^s)\right)$. If the price of product ℓ in price vector $\boldsymbol{p}^s$ is large, then the choice probability $\theta_\ell(\boldsymbol{p}^s)$ is expected to be small, which implies that the change in the price of product ℓ under GA is expected to be relatively small as well. However, if the price of a product is large, then its price has to change significantly to make a noticeable impact in the price of this product! In contrast, under PUPD, the choice probability $\theta_\ell(\boldsymbol{p}^s)$ does not come into play in a direct way when computing the change in the prices, except in an indirect way when computing the scalars $(\mathtt{t}_j^s : j \in \mathsf{V})$. We note that PUPD and GA can converge to different stationary points of the expected revenue function, but over all of our problem instances, the expected revenues at the stationary points obtained by PUPD and GA differ at most by 0.02%. We also tested the robustness of the PUPD algorithm by using different initial prices. The results are similar to those in Table 4. For details, the reader is referred to Online Appendix F.

Finally, we applied the PUPD algorithm on problem instances with two levels in the tree. When there are two levels in the tree, Gallego and Wang (2013) provide a set of implicit equations that must be satisfied by the optimal price vector. We use golden section search to numerically find a solution to this set of equations. Similar to the stopping criterion for PUPD, we stop the golden section search when we reach a price vector $\boldsymbol{p}^s$ that satisfies $\|\nabla R_{\mathsf{root}}(\boldsymbol{p}^s)\|_2 \leq 10^{-6}$. Our numerical results demonstrated that PUPD can run substantially faster than golden section search for problem instances with larger number of products. Furthermore, over all of our problem instances, the expected revenues at the stationary points obtained by PUPD and the golden search are essentially equal. For the details, the reader is referred to Online Appendix F.

9. The Choice Model and the Assortment Optimization Problem

In this section, we begin by providing practical motivation for the use of the d-level nested logit model. First, we show that this model is compatible with random utility maximization (RUM) principle. Second, we discuss that this model is equivalent to the elimination by aspects. Third, we provide practical situations in which the d-level nested logit model with three or more levels is used to capture the customer choice process. Finally, we demonstrate that the predictive accuracy of the d-level nested logit model increases as the number of levels in the tree increases. This discussion provides practical motivation for the d-level nested logit model. Following this discussion, we explain how the assortment and price optimization problems considered in this paper become useful when solving large-scale revenue management problems.

Random Utility Maximization: An attractive framework for describing the choice process of customers is to appeal to the RUM principle, where a customer associates random utilities with the products and the no-purchase option. The customer then chooses the option that provides the largest utility. As discussed in Section 3, if the customer is at a non-leaf node j, then she chooses node $k \in \mathsf{Children}(j)$ with probability $V_k(S_k)/\sum_{\ell \in \mathsf{Children}(j)} V_\ell(S_\ell)$. For the customer to choose the product at product ℓ, she needs to follow the path from root to the leaf node ℓ. Thus, given an assortment $S = (S_j : j \in \mathsf{V})$, the probability that a customer chooses a product $\ell \in S$ is

$$\theta_\ell(S) = \prod_{h=1}^{d} \frac{V_{\mathsf{An}(\ell,h)}\left(S_{\mathsf{An}(\ell,h)}\right)}{\sum_{k \in \mathsf{Sibling}(\mathsf{An}(\ell,h))} V_k(S_k)} \quad . \tag{8}$$

The following theorem shows that the choice probability above can be obtained by appealing to the RUM principle. The proof is given in Online Appendix G.

Theorem 9.1 (Consistency with Utility Maximization)**.** *The choice probability under the d-level nested logit model given in* (8) *is consistent with the RUM principle.*

Börch-Supan (1990) shows the compatibility of the two-level nested logit model with the RUM principle; see, also, McFadden (1974). The above theorem extends the result to the d-level setting. McFadden (1978) and Cardell (1997) discuss the consistency with the RUM principle for a general d-level setting, but they do not provide a formal proof.

Elimination by Aspects Model: Theorem 9.1 provides a behavioral justification for the d-level nested logit model by relating it to the RUM principle. Another approach to provide justification for the d-level nested logit model is to relate it to the elimination by aspects model proposed by Tversky (1972a,b) and Tversky and Sattath (1979). This model is widely accepted in the psychology and marketing literature for providing a reasonable heuristic for decision making. Roughly speaking, the elimination by aspects model works similar to a screening rule. The customer chooses an attribute, and eliminates the alternatives that do not meet the requirements based on this attribute. She continues in this manner until she is left with a single alternative. There is a large body of literature that explores the equivalence between the d-level nested logit and the elimination by aspects model. Manski and McFadden (1981) point out the equivalence between the two models in somewhat restricted settings. Tversky and Sattath (1979), Vovsha (1997) and Train (2003) also comment on the equivalence between the d-level nested logit and elimination by aspects models. Batley and Daly (2006) provide a formal proof that the d-level nested logit model can be derived by appealing to the elimination by aspects model.

Applications of Nested Logit Models with Three or More Levels: Psychologists and marketing researchers have conducted many experiments to validate the elimination by aspects model, and thus, providing justification for the d-level nested logit model in the process. Studies show that consumers indeed use multiple attributes to sequentially eliminate products from their considerations. For example, Andrews and Manrai (1998) consider scanner panel data for fabric softeners and they find that customers screen fabric softeners based on four attributes, brand name, size, product form and formulation. For advanced photo systems, Gilbride and Allenby (2004) show that customers consider over 10 attributes to determine the camera system that they will purchase. These attributes include body style, mid-roll change, operation feedback, zoom lens, view finder and price. Pihlens (2008) models the choice among different pasta sauce options by using 16 attributes, which include container size and type, sauce base, main ingredients and price. Waddell (1993) uses the three-level nested logit model to capture residential location choice. The three levels in the tree correspond to workplace, housing tenure, and residence choice. Coldren and Koppelman (2005) find that using three levels in the nested logit model provides an improvement over two levels when modeling demand for flight itineraries. Cervero and Duncan (2008) model rail travel adoption for daily commute by using the nested logit model with three levels. Carson et al. (2009) use the four-level nested logit model to represent fishing demand in Alaska. Gramlich (2009) models the choice among different vehicle options using trees with four levels, which correspond to product features such as vehicle segment, size, comfort level, country of origin and specific vehicle model.

Parameter Estimation and Predictive Accuracy as the Number of Levels d Increases: In Online Appendix H, we show that the log-likelihood function under the d-level nested logit model exhibits desirable concavity properties, enabling us to develop an effective estimation method. In Online Appendix H, we exploit these concavity properties to present numerical experiments, in which we fit the d-level nested logit model to customer choice data with progressively increasing number of levels d in the tree. Our experiments demonstrate that the predictive accuracy improves as the number of levels d increases. Comparing with the multinomial logit model ($d = 1$), two-level models ($d = 2$) reduce the misclassification errors by about 9%, while with three-level models ($d = 3$), the misclassification errors are reduced by 15%.

Assortment and Price Optimization Problems: Assortment and price optimization problems considered in this paper are useful when maximizing the immediate expected revenue from a customer but these problems also find applications when solving large-scale revenue management problems. Consider the case in which an airline operates k flights legs. We index the flight legs by $1, \ldots, k$. We let c_i be the capacity available on flight leg i. The airline offers n itineraries to its customers, where an itinerary is a combination of connecting flight legs and an associated price. We index the itineraries by $1, \ldots, n$. We use r_ℓ to denote the revenue associated with itinerary ℓ. We let $a_{i\ell} = 1$ if itinerary ℓ uses flight leg i, otherwise $a_{i\ell} = 0$. The problem takes place over a finite selling horizon of T periods. Time period T is the beginning of the selling horizon and the flight legs depart at time period 0. For simplicity, we assume that there is one customer arrival at each time period. The goal of the airline is to find a policy to decide which set of itineraries to make available to its customers to maximize the total expected revenue over the selling horizon.

We let x_i be the remaining capacity on flight leg i at the beginning of a time period and use $x = (x_1, \ldots, x_k)$ as the state variable. The feasible set of itineraries that can be offered at a time period is given by $\mathcal{F}(x) = \{S \subseteq \{1, \ldots, n\} : a_{i\ell}\, \mathbf{1}(\ell \in S) \leq x_i \ \forall\, i = 1, \ldots k,\ \ell = 1, \ldots, n\}$, where $\mathbf{1}(\cdot)$ is the indicator function. The definition of $\mathcal{F}(x)$ captures the fact that if we offer a set that includes itinerary ℓ and itinerary ℓ uses flight leg i, then there has to be capacity available on flight leg i. Thus, we can find the optimal policy by solving the dynamic program

$$\begin{aligned} J_t(x) &= \max_{S \subseteq \mathcal{F}(x)} \sum_{\ell \in S} \theta_\ell(S) \left(r_\ell + J_{t-1}(x - \textstyle\sum_{i=1}^k a_{i\ell}\, e_i) \right) + \theta_0(S) J_{t-1}(x) \\ &= \max_{S \subseteq \mathcal{F}(x)} \left\{ \sum_{\ell \in S} \theta_\ell(S) \left(r_\ell + J_{t-1}(x - \textstyle\sum_{i=1}^k a_{i\ell}\, e_i) - J_{t-1}(x) \right) \right\} + J_{t-1}(x) \ , \end{aligned} \tag{9}$$

where e_i is a unit vector in $\mathbb{R}^k$ with a one in the element corresponding to flight leg i and the choice probability $\theta_\ell(S)$ is driven by the d-level nested logit model. For practical airline networks with large number of flight legs, the dynamic program above involves a high-dimensional state variable. Thus, it is difficult to compute the value functions exactly. However, there exist a variety of methods to build approximations to the value functions; see Liu and van Ryzin (2008); Zhang and Adelman (2009). In this case, if we have an approximation $\tilde{J}_t(\cdot)$ to the value function $J_t(\cdot)$, then we can replace $J_t(\cdot)$ in the second maximization problem above with $\tilde{J}_t(\cdot)$ and solve this problem to find a set of itineraries to offer. To solve this problem, we observe that we can drop all itineraries for which we do not have enough capacity to offer. Once we drop these itineraries, this problem becomes an assortment optimization problem in which the revenue of product ℓ is given by $r_\ell + \tilde{J}_{t-1}(x - \sum_{i=1}^k a_{i\ell}\, e_i) - \tilde{J}_{t-1}(x)$ and we maximize the expected revenue from a customer. Thus, if we are given approximations to the value functions, then the problem of finding a set of itineraries to offer precisely corresponds to the assortment problem considered in this paper.

Another use of our assortment optimization problems occurs when we build a linear programming approximation to the revenue management problem described above. To formulate this linear program, we use the decision variable $h(S)$ to denote the number of time periods during which we offer the subset S of itineraries. In this case, we can use the linear program

$$\begin{aligned}
\max \quad & \sum_{S \subseteq \{1,\ldots,n\}} \sum_{\ell=1}^n r_\ell\, \theta_\ell(S)\, h(S) && (10)\\
\text{s.t.} \quad & \sum_{S \subseteq \{1,\ldots,n\}} \sum_{\ell=1}^n a_{i\ell}\, \theta_\ell(S)\, h(S) \le c_i && \forall\, i = 1,\ldots,k\\
& \sum_{S \subseteq \{1,\ldots,n\}} h(S) = T\\
& h(S) \ge 0 && \forall\, S \subseteq \{1,\ldots,n\},
\end{aligned}$$

to approximate the optimal expected revenue over the selling horizon; see Gallego et al. (2004); Liu and van Ryzin (2008). Noting that $\sum_{\ell=1}^n r_\ell\, \theta_\ell(S)$ is the expected revenue at a time period during which we offer the subset S of itineraries, the objective function in the problem above computes the total expected revenue over the selling horizon. The first set of constraints ensures that the total expected capacity consumption on each flight leg does not exceed the capacity availability on the flight leg, whereas the second constraint ensures that we offer a subset of itineraries at each time period. If there are n itineraries, then the number of decision variables in the linear program above is 2^n. Thus, it is customary to solve the linear program by using column generation. Using $\{\mu_i : i = 1, \ldots, k\}$ and σ to denote the dual variables associated with the two sets of constraints above, the reduced cost of the decision variable $h(S)$ is given by $\sum_{\ell=1}^n r_\ell\, \theta_\ell(S) - \sum_{i=1}^k \sum_{\ell=1}^n a_{i\ell}\, \theta_\ell(S)\, \mu_i - \sigma$. Thus, when generating columns, to find the decision variable with the largest reduced cost, we can solve the problem $\max_{S \subseteq \{1,\ldots,n\}} \sum_{\ell=1}^{n} r_\ell\, \theta_\ell(S) - \sum_{i=1}^{k} \sum_{\ell=1}^{n} a_{i\ell}\, \theta_\ell(S)\, \mu_i = \max_{S \subseteq \{1,\ldots,n\}} \sum_{\ell=1}^{n} \theta_\ell(S)\, (r_\ell - \sum_{i=1}^{k} a_{i\ell}\, \mu_i)$. We observe that the last problem precisely has the same form as the assortment optimization problem considered in this paper, where the revenue of product ℓ is $r_\ell - \sum_{i=1}^{k} a_{i\ell}\mu_i$ and we find an assortment of products to maximize the expected revenue from a customer. Thus, the assortment optimization problem studied in this paper becomes useful when generating columns for linear programming approximations of large-scale revenue management problems.

In the dynamic program in (9) and the linear program in (10), the prices of the itineraries are fixed and the airline decides which set of itineraries to make available. These models have their analogues in which the set of offered itineraries are fixed and the prices of the itineraries are decision variables; see Gallego and van Ryzin (1997). The pricing problem studied in this paper becomes useful when dealing with large-scale revenue management problems, in which the airline offers a fixed set of itineraries but dynamically adjusts the prices of the offered itineraries.

10. Conclusion

We considered assortment and price optimization problems under the d-level nested logit model. We showed how to obtain an optimal assortment, and developed an algorithm to obtain a set of prices corresponding to a stationary point of the expected revenue function. We investigated several motivations for the d-level nested logit model and for our assortment and pricing problems. For future research directions, one can consider extending the PUPD algorithm for joint assortment and price optimization, establishing connections between our assortment optimization algorithm and the linear programming formulation for the two-level model proposed by Davis et al. (2014), and developing assortment and pricing algorithms for the case where $\eta_j > 1$ for some nodes $j \in \mathsf{V}$.

Acknowledgement

The authors would like to thank the Area Editor, the Associate Editor, and the three anonymous referees for their detailed and thoughtful comments that improved the paper in many ways. We also want to thank Han Qiu for pointing out an error in the earlier version of our manuscript, which led to a new proof showing that the revenues under the PUPD algorithm are always bounded.

References

Andrews, R. L., and A. K. Manrai. 1998. Featured-based elimination: Model and emipirical comparison. *European Journal of Operational Research* 111 (2): 248–267.

Batley, R., and A. Daly. 2006. On the equivalence between elimination-by-aspects and generalised extreme value models of choice behaviour. *Journal of Mathematical Psychology* 50 (5): 456–467.

Börch-Supan, A. 1990. On the compatibility of nested logit models with utility maximization. *Journal of Econometrics* 43 (3): 373–388.

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

Cardell, N. S. 1997. Variance components structures for the extreme-value and logistic distributions with applications to models of heterogeneity. *Economic Theory* 13 (2): 185–213.

Carson, R. T., W. M. Hanemann, and T. C. Wegge. 2009. A nested logit model of recreational fishing demand in Alaska. *Marine Resource Economics* 24 (2): 101–129.

Cervero, R., and M. Duncan. 2008. Residential self selection and rail commuting: A nested logit analysis. Technical report, University of California, Berkeley.

Chen, K. D., and W. H. Hausman. 2000. Technical Note: Mathematical properties of the optimal product line selection problem using choice-based conjoint analysis. *Management Science* 46 (2): 327–332.

Coldren, G. M., and F. S. Koppelman. 2005. Modeling the competition among air-travel itinerary shares: GEV model development. *Transportation Research Part A* 39 (4): 345–365.

Davis, J. M., G. Gallego, and H. Topaloglu. 2014. Assortment optimization under variants of the nested logit model. *Operations Research* 62 (2): 250–273.

Dong, L., P. Kouvelis, and Z. Tian. 2009. Dynamic pricing and inventory control of substitute products. *Manufacturing & Service Operations Management* 11 (2): 317–339.

Farias, V. F., S. Jagabathula, and D. Shah. 2013. A non-parametric approach to modeling choice with limited data. *Management Science* 59 (2): 305–322.

Gallego, G., G. Iyengar, R. Phillips, and A. Dubey. 2004. Managing flexible products on a network. Working Paper, Columbia University.

Gallego, G., R. Ratliff, and S. Shebalov. 2011. A general attraction model and an efficient formulation for the network revenue management problem. Technical report, Columbia University, New York, NY.

Gallego, G., and H. Topaloglu. 2012. Constrained assortment optimization for the nested logit model. To appear in *Management Science*.

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

Gallego, G., and R. Wang. 2013. Multi-product price optimization and competition under the nested logit model with product-differentiated price sensitivities. *Operations Research* 62 (2): 450–461.

Gilbride, T. J., and G. M. Allenby. 2004. A choice model with conjunctive, disjunctive, and compensatory screening rules. *Marketing Science* 23 (3): 391–406.

Gramlich, J. P. 2009. *Gas prices and fuel efficiency in the US automobile industry: Policy implications of endogenous product choice*. Ph.D. Dissertation, Georgetown University.

Hanson, W., and K. Martin. 1996. Optimizing multinomial logit profit functions. *Management Science* 42 (7): 992–1003.

Kleinberg, J., and E. Tardos. 2005. *Algorithm Design*. New York: Addison Wesley.

Kök, A. G., M. Fisher, and R. Vaidyanathan. 2006. Assortment planning: Review of literature and industry practice. In *Retail Supply Chain Management*. New York: Springer.

Kök, A. G., and Y. Xu. 2010. Optimal and competitive assortments with endogenous pricing under hierarchical choice models. *Management Science* 57 (9): 1546–1563.

Koppelman, F. S., and V. Sethi. 2000. Closed-form discrete-choice models. In *Handbook of Transport Modeling*, ed. D. A. Hensher and K. J. Button.

Li, G., and P. Rusmevichientong. 2014. A greedy algorithm for the two-level nested logit model. *Operations Research Letters* 42 (5): 319–324.

Li, H., and T. Huh. 2013. Pricing under the nested attraction model with a multi-stage choice structure. INFORMS Annual Meeting, Minneapolis, MN.

Li, H., and W. T. Huh. 2011. Pricing multiple products with the multinomial logit and nested logit models: Concavity and implications. *Manufacturing & Service Operations Management* 13 (4): 549–563.

Li, Z. 2007. A single-period assortment optimization model. *Production and Operations Management* 16 (2): 369–380.

Liu, Q., and G. J. van Ryzin. 2008. On the choice-based linear programming model for network revenue management. *Manufacturing and Service Operations Management* 10 (2): 288–310.

Luce, R. D. 1959. *Individual choice behavior: A theoretical analysis.* New York, NY: Wiley.

Manski, C. F., and D. L. McFadden. 1981. *Structural analysis of discrete data and econometric applications.* The MIT Press.

McFadden, D. 1974. Conditional logit analysis of qualitative choice behavior. *Frontiers in Econometrics* 1 (2): 105–142.

McFadden, D. 1978. Modeling the choice of residential location. *Transportation Research Record* (672): 72–77.

Pihlens, D. A. 2008. *The multi-attribute elimination-by-aspects model.* Ph.D. Dissertation, University of Technology Sydney.

Rusmevichientong, P., D. B. Shmoys, C. Tong, and H. Topaloglu. 2013. Assortment optimization under the multinomial logit model with random choice parameters. To appear in *Production and Operations Management.*

Song, J.-S., and Z. Xue. 2007. Demand management and inventory control for substitutable products. Working Paper, Duke University.

Sun, W., and Y. Yuan. 2006. *Optimization Theory and Methods: Nonlinear Programming*, Volume 1. Springer.

Talluri, K., and G. J. van Ryzin. 2004. Revenue management under a general discrete choice model of consumer behavior. *Management Science* 50 (1): 15–33.

Train, K. 2003. *Discrete Choice Methods with Simulation.* New York, NY: Cambridge University Press.

Tversky, A. 1972a. Choice-by-elimination. *Journal of Mathematical Psychology* 9 (4): 341–367.

Tversky, A. 1972b. Elimination-by-aspect: A theory of choice. *Psychological Review* 79 (4): 281–299.

Tversky, A., and S. Sattath. 1979. Preference Trees. *Psychological Review* 86 (6): 542–573.

van Ryzin, G. J., and S. Mahajan. 1999. On the relationship between inventory costs and variety benefits in retail assortments. *Management Science* 45:1496–1509.

Vovsha, P. 1997. Application of cross-nested logit model to mode choice in Tel Aviv, Israel metropolitan area. *Transportation Research Record* 1607:6–15.

Waddell, P. 1993. Exogenous workplace choice in residential location models: Is the assumption valid? *Geographical Analysis* 25 (1): 65–82.

Wen, C.-H., and F. S. Koppelman. 2001. The generalized nested logit model. *Transportation Research Part B* 35:627–641.

Zhang, D., and D. Adelman. 2009, August. An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science* 43 (3): 381–394.

Online Appendix

A. Omitted Results in Section 4

The proofs of Proposition 4.1 and Theorem 4.2 use the next lemma, which shows the relationship between the local problem at node j given in (4) and the same problem at its children nodes.

Lemma A.1. *Consider an arbitrary non-leaf node j. Assume that for every $k \in \mathsf{Children}(j)$, there exists a set $\widehat{S}_k \subseteq N_k$ such that $V_k(\widehat{S}_k)\left(R_k(\widehat{S}_k) - \mathtt{t}^*_{\mathsf{Parent}(k)}\right) \geq 0$ and*

$$V_k(\widehat{S}_k)\left(R_k(\widehat{S}_k) - \mathtt{t}^*_{\mathsf{Parent}(k)}\right) \geq V_k(S^*_k)\left(R_k(S^*_k) - \mathtt{t}^*_{\mathsf{Parent}(k)}\right) \ . \tag{11}$$

Then, letting $\widehat{S}_j = \dot{\bigcup}_{k\in \mathsf{Children}(j)} \widehat{S}_k$, we have

$$V_j(\widehat{S}_j)\left(R_j(\widehat{S}_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) \geq V_j(S^*_j)\left(R_j(S^*_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) \ . \tag{12}$$

Moreover, if the inequality in (11) is strict for some $k \in \mathsf{Children}(j)$, so is the inequality in (12).

Proof: There are two cases to consider: $R_j(S^*_j) \leq \mathtt{t}^*_{\mathsf{Parent}(j)}$ and $R_j(S^*_j) > \mathtt{t}^*_{\mathsf{Parent}(j)}$. First, consider the case $R_j(S^*_j) \leq \mathtt{t}^*_{\mathsf{Parent}(j)}$. Then, it follows from the definition of $\mathtt{t}^*_j$ that $\mathtt{t}^*_j = \mathtt{t}^*_{\mathsf{Parent}(j)}$. By our hypothesis, $V_k(\widehat{S}_k)\left(R_k(\widehat{S}_k) - \mathtt{t}^*_j\right) \geq 0$ for all $k \in \mathsf{Children}(j)$, which implies that

$$V_j(\widehat{S}_j)^{1/\eta_j}\left(R_j(\widehat{S}_j) - \mathtt{t}^*_j\right) = \sum_{k\in \mathsf{Children}(j)} V_k(\widehat{S}_k)\left(R_k(\widehat{S}_k) - \mathtt{t}^*_j\right) \geq 0 \ ,$$

where the equality follows by noting that

$$V_j(\widehat{S}_j)^{1/\eta_j} = \sum_{k\in \mathsf{Children}(j)} V_k(\widehat{S}_k) \qquad \text{and} \qquad R_j(\widehat{S}_j) = \frac{\sum_{k\in \mathsf{Children}(j)} V_k(\widehat{S}_k)\, R_k(\widehat{S}_k)}{\sum_{k\in \mathsf{Children}(j)} V_k(\widehat{S}_k)} \ .$$

Thus, $V_j(\widehat{S}_j)\left(R_j(\widehat{S}_j) - \mathtt{t}^*_j\right) \geq 0 \geq V_j(S^*_j)\left(R_j(S^*_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right)$, completing the first case.

Now, assume that $R_j(S^*_j) > \mathtt{t}^*_{\mathsf{Parent}(j)}$. Since $V_k(\widehat{S}_k)\left(R_k(\widehat{S}_k) - \mathtt{t}^*_j\right) \geq V_k(S^*_k)\left(R_k(S^*_k) - \mathtt{t}^*_j\right)$ for all $k \in \mathsf{Children}(j)$, we obtain

$$\begin{aligned} V_j(\widehat{S}_j)^{1/\eta_j}\left(R_j(\widehat{S}_j) - \mathtt{t}^*_j\right) &= \sum_{k\in \mathsf{Children}(j)} V_k(\widehat{S}_k)\left(R_k(\widehat{S}_k) - \mathtt{t}^*_j\right) \\ &\geq \sum_{k\in \mathsf{Children}(j)} V_k(S^*_k)\left(R_k(S^*_k) - \mathtt{t}^*_j\right) \ = \ V_j(S^*_j)^{1/\eta_j}\left(R_j(S^*_j) - \mathtt{t}^*_j\right) \ , \end{aligned}$$

where the two equalities above follow from an argument similar to the one we used in the first case. The above expression implies that

$$\begin{aligned}
R_j(\widehat{S}_j) - \mathtt{t}^*_{\mathsf{Parent}(j)} &\geq \left[\frac{V_j(S_j^*)}{V_j(\widehat{S}_j)}\right]^{1/\eta_j} \left(R_j(S_j^*) - \mathtt{t}^*_j\right) + \mathtt{t}^*_j - \mathtt{t}^*_{\mathsf{Parent}(j)} \\
&= \eta_j \left[\frac{V_j(S_j^*)}{V_j(\widehat{S}_j)}\right]^{1/\eta_j} \left(R_j(S_j^*) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) + (1-\eta_j)\left(R_j(S_j^*) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) \\
&= \left(\eta_j \left[\frac{V_j(S_j^*)}{V_j(\widehat{S}_j)}\right]^{1/\eta_j} + 1 - \eta_j\right) \left(R_j(S_j^*) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) ,
\end{aligned}$$

where the first equality follows from the fact that $\mathtt{t}^*_j = \eta_j \mathtt{t}^*_{\mathsf{Parent}(j)} + (1-\eta_j) R_j(S_j^*)$ because $R_j(S_j^*) > \mathtt{t}^*_{\mathsf{Parent}(j)}$. The function $x \mapsto x^{1/\eta_j}$ is convex, whose derivative at $x = 1$ is $1/\eta_j$, and thus, for all $x \in \mathbb{R}_+$, we have $x^{1/\eta_j} \geq 1 + \frac{1}{\eta_j}(x-1)$. Writing this inequality as $\eta_j\, x^{1/\eta_j} + 1 - \eta_j \geq x$, and using this inequality with $x = V_j(S_j^*)/V_j(\widehat{S}_j)$ in the above expression, we obtain

$$R_j(\widehat{S}_j) - \mathtt{t}^*_{\mathsf{Parent}(j)} \geq \frac{V_j(S_j^*)}{V_j(\widehat{S}_j)} \left(R_j(S_j^*) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) ,$$

which yields $V_j(\widehat{S}_j)\left(R_j(\widehat{S}_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) \geq V_j(S_j^*)\left(R_j(S_j^*) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right)$ and completes the second case. Finally, going through the same reasoning in the proof, it is easy to verify that if the inequality in (11) is strict for some k, then the inequality in (12) is also strict. □

Here is the proof of Proposition 4.1.

Proof of Proposition 4.1: We will prove the result by induction on the depth of node j. The result is trivially true at root because the local problem at root is identical to the assortment problem in (3). Now, assume that the result is true for all nodes at depth h. We want to show that the result also holds for all nodes at depth $h+1$. Suppose, on the contrary, that there exists node i at depth $h+1$ such that

$$V_i(S_i^*)\left(R_i(S_i^*) - \mathtt{t}^*_{\mathsf{Parent}(i)}\right) \; < \; \max_{S_i \subseteq N_i} V_i(S_i)\left(R_i(S_i) - \mathtt{t}^*_{\mathsf{Parent}(i)}\right) ,$$

and let $\widetilde{S}_i \subseteq N_i$ be the set that achieves the maximum on the right side above. By our construction, we have that $V_i(\widetilde{S}_i)\left(R_i(\widetilde{S}_i) - \mathtt{t}^*_{\mathsf{Parent}(i)}\right) \geq 0$ because an empty set yields a value of zero for the optimization problem on the right side above. Also,

$$V_i(\widetilde{S}_i)\left(R_i(\widetilde{S}_i) - \mathtt{t}^*_{\mathsf{Parent}(i)}\right) \; > \; V_i(S_i^*)\left(R_i(S_i^*) - \mathtt{t}^*_{\mathsf{Parent}(i)}\right) .$$

We use j to denote the parent of node i. For each $k \in \mathsf{Children}(j)$, let $\widehat{S}_k$ be defined by:

$$\widehat{S}_k = \begin{cases} \widetilde{S}_i & \text{if } k = i \\ \arg\max_{S_k \subseteq N_k} V_i(S_k)\left(R_k(S_k) - \mathtt{t}^*_j\right) & \text{if } k \neq i \ . \end{cases}$$

By our construction, for every $k \in \mathsf{Children}(j)$,

$$V_k(\widehat{S}_k)\left(R_k(\widehat{S}_k) - \mathtt{t}^*_{\mathsf{Parent}(k)}\right) \quad \geq \quad V_k(S^*_k)\left(R_k(S^*_k) - \mathtt{t}^*_{\mathsf{Parent}(k)}\right) \ ,$$

and the inequality above is strict if $k = i$. Letting $\widehat{S}_j = \bigcup_{k \in \mathsf{Children}(j)} \widehat{S}_k$, it then follows from Lemma A.1 that

$$V_j(\widehat{S}_j)\left(R_j(\widehat{S}_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) \quad > \quad V_j(S^*_j)\left(R_j(S^*_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) \ .$$

Since node j is at depth h, by the inductive hypothesis, S^*_j solves the local problem at node j, yielding

$$V_j(S^*_j)(R_j(S^*_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}) = \max_{S_j \subseteq N_j} V_j(S_j)\left(R_j(S_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) \ .$$

The above equality contradicts the last inequality! Therefore, the result must also hold for all nodes at depth $h+1$, as desired. □

Below is the proof of Theorem 4.2.

Proof of Theorem 4.2: Because of the hypothesis of the theorem, since $\widehat{S}_k$ solves the local problem at node k, we must have $V_k(\widehat{S}_k)\left(R_k(\widehat{S}_k) - \mathtt{t}^*_{\mathsf{Parent}(k)}\right) \geq 0$ because the empty set gives an objective value of zero for the local problem at node k and $\widehat{S}_k$ is an optimal solution to the local problem at node k. Also, by definition of $\widehat{S}_k$, for every $k \in \mathsf{Children}(j)$,

$$V_k(\widehat{S}_k)\left(R_k(\widehat{S}_k) - \mathtt{t}^*_{\mathsf{Parent}(k)}\right) \geq V_k(S^*_k)\left(R_k(S^*_k) - \mathtt{t}^*_{\mathsf{Parent}(k)}\right) \ .$$

Let $\widehat{S}_j = \bigcup_{k \in \mathsf{Children}(j)} \widehat{S}_k$. It then follows from Lemma A.1 that

$$V_j(\widehat{S}_j)\left(R_j(\widehat{S}_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) \geq V_j(S^*_j)\left(R_j(S^*_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) = \max_{S_j \subseteq N_j} V_j(S_j)\left(R_j(S_j) - \mathtt{t}^*_{\mathsf{Parent}(j)}\right) \ ,$$

where the last equality follows from Proposition 4.1. This completes the proof. □

B. Omitted Results in Section 5

Below, we give the proof of Theorem 5.2.

Proof of Theorem 5.2: We will prove by induction that for each $h = 0, 1, 2, \ldots, d$,

$$\sum_{j \in \mathsf{V} \,:\, \mathsf{depth}(j) = h} |\mathcal{A}_j| \;\leq\; 2n \;,$$

where $\mathsf{depth}(j)$ denotes the depth of node j. The result is true for $h = d$ because for each leaf node j, $\mathcal{A}_j = \{\{j\}, \varnothing\}$. Suppose the result is true for $h+1$. Consider nodes at level h. It follows from Lemma 5.1 that

$$\sum_{j \in \mathsf{V} \,:\, \mathsf{depth}(j) = h} |\mathcal{A}_j| \;\leq\; \sum_{j \in \mathsf{V} \,:\, \mathsf{depth}(j) = h} \;\sum_{k \in \mathsf{Children}(j)} |\mathcal{A}_k| \;=\; \sum_{j \in \mathsf{V} \,:\, \mathsf{depth}(j) = h+1} |\mathcal{A}_j| \;\leq\; 2n \;,$$

where the last inequality follows from the inductive hypothesis. Therefore, the total running time of the algorithm is

$$\begin{aligned}
\sum_{h=0}^{d} \sum_{j \in \mathsf{V} \,:\, \mathsf{depth}(j) = h} |\mathcal{A}_j| \log\left(|\mathcal{A}_j|\right) \;&\leq\; \sum_{h=0}^{d} \left(\sum_{j \in \mathsf{V} \,:\, \mathsf{depth}(j) = h} |\mathcal{A}_j| \right) \times \log \left(\sum_{j \in \mathsf{V} \,:\, \mathsf{depth}(j) = h} |\mathcal{A}_j| \right) \\
&\leq\; \sum_{h=0}^{d} 2n \log(2n) \;=\; 2(d+1) n \log(2n) \;=\; O(dn \log n) \;,
\end{aligned}$$

which completes the proof. □

C. Practical Performance of Assortment Optimization Algorithm

We test the practical performance of our assortment optimization algorithm by applying it on test problems with two and three levels in the tree. To our knowledge, our algorithm is the only efficient approach for solving the assortment problem when there are three levels in the tree. For the test problems with three levels, we compare our assortment optimization algorithm with a full enumeration scheme that checks the expected revenue from every possible assortment. For the test problems with two levels, Davis et al. (2014) show that the optimal assortment can be obtained by solving a linear program. For the test problems with two levels, we compare our assortment optimization algorithm with the full enumeration scheme and the linear programming approach in Davis et al. (2014). We refer to our assortment optimization algorithm, the full enumeration scheme and the linear programming approach as AA, FE and LP, respectively.

For the test problems with three levels, we use m_h to denote the number of children of each node at depth h and we vary (m_0, m_1, m_2) over $\{2, 4, 8\} \times \{2, 4, 8\} \times \{2, 4, 8\}$. Therefore, smallest problem instances have $2^3 = 8$ products, whereas largest ones have $8^3 = 512$ products. We refer to each combination of (m_0, m_1, m_2) as a problem class. In each problem class, we randomly generate 200 problem instances. To generate each problem instance, we sample the parameters r_j and v_j for each leaf node from the uniform distribution over $[0, 5]$ and the parameter η_j for each non-leaf node from the uniform distribution over $[0, 1]$. We use the same approach to generate the test problems with two levels, but we vary (m_0, m_1) over $\{2, 4, 8\} \times \{2, 4, 8\}$.

Table 5 gives our results. In the left portion of the table, the first three columns show the parameters (m_0, m_1, m_2) for each problem class with three levels. The next two columns show the running times for AA and FE, averaged over 200 problem instances in a particular problem class. When the number of products exceeds 16, FE is not able to check the expected revenues from all 2^n assortments in a reasonable amount of time. For such problem instances, since the number of operations required to compute the expected revenue from each assortment is the same, we check the expected revenues from 1,000 randomly sampled assortments and linearly extrapolate the running time for checking the expected revenues from all 2^n assortments. The format of the right portion of Table 5 is similar to that of the left portion, but the right portion focuses on the test problems with two levels and gives the average running times for AA, FE and LP.

Considering the test problems with three levels in the left portion of Table 5, our results indicate that the running time for AA can be substantially faster than the running time for FE. For the problem class $(2, 2, 2)$, which involves the smallest number of products, the average running times for AA and FE are comparable. For all other problem classes, the average running time for AA is smaller than the average running time for FE by orders of magnitude. The running time of AA is reasonable. Even for the largest problem instances with 512 products, the average running time for AA is slightly above one second. Over all of our test problems, the number of assortments in the collection $\mathcal{A}_{\textsf{root}}$ constructed by AA is roughly equal to the number of products. Considering the test problems with two levels in the right portion of Table 5, our results indicate that the running time for LP is generally faster than the running time for AA. For a two-level tree with m_0 and m_1 child nodes in the two levels of the tree, there are $m_0\, m_1$ products and the linear program proposed by Davis et al. (2014) has $O(m_0)$ decision variables and $O(m_0 m_1)$ constraints. The number of operations required to solve a general linear program of this size is $O(m_0^4 m_1)$. In contrast, Theorem 5.2 shows that AA requires $O(m_0\, m_1 \log(m_0\, m_1))$ operations. So, the computational complexity for AA is better than the computational complexity for solving the linear program used by LP, but many

Prb. Class m_0	m_1	m_2	Avg. Secs. for AA	Avg. Secs. for FE
2	2	2	1.64×10^{-2}	9.50×10^{-3}
2	2	4	2.29×10^{-2}	2.49×10^{0}
2	2	8	3.17×10^{-2}	1.57×10^{5}
2	4	2	3.05×10^{-2}	2.47×10^{0}
2	4	4	5.23×10^{-2}	1.84×10^{5}
2	4	8	7.35×10^{-2}	7.07×10^{14}
2	8	2	7.94×10^{-2}	1.68×10^{5}
2	8	4	1.30×10^{-1}	7.24×10^{14}
2	8	8	2.21×10^{-1}	1.40×10^{34}
4	2	2	2.96×10^{-2}	2.50×10^{0}
4	2	4	4.17×10^{-2}	1.65×10^{5}
4	2	8	6.75×10^{-2}	7.13×10^{14}
4	4	2	5.89×10^{-2}	1.67×10^{5}
4	4	4	9.68×10^{-2}	7.33×10^{14}
4	4	8	1.60×10^{-1}	1.39×10^{34}
4	8	2	1.62×10^{-1}	7.64×10^{14}
4	8	4	2.70×10^{-1}	1.47×10^{34}
4	8	8	4.72×10^{-1}	5.36×10^{72}
8	2	2	6.83×10^{-2}	1.71×10^{5}
8	2	4	1.05×10^{-1}	7.41×10^{14}
8	2	8	1.71×10^{-1}	1.40×10^{34}
8	4	2	1.41×10^{-1}	7.84×10^{14}
8	4	4	2.38×10^{-1}	1.55×10^{34}
8	4	8	3.94×10^{-1}	5.37×10^{72}
8	8	2	3.86×10^{-1}	1.63×10^{34}
8	8	4	6.32×10^{-1}	5.91×10^{72}
8	8	8	1.10×10^{0}	7.76×10^{149}

Prb. Class m_0	m_1	Avg. Secs. for AA	Avg. Secs. for FE	Avg. Secs. for LP
2	2	8.41×10^{-3}	3.77×10^{-4}	2.94×10^{-3}
2	4	1.11×10^{-2}	5.10×10^{-3}	3.00×10^{-3}
2	8	1.14×10^{-2}	1.02×10^{0}	3.03×10^{-3}
4	2	1.32×10^{-2}	5.10×10^{-3}	3.03×10^{-3}
4	4	1.98×10^{-2}	1.15×10^{0}	3.17×10^{-3}
4	8	2.77×10^{-2}	7.59×10^{4}	3.25×10^{-3}
8	2	3.63×10^{-2}	1.19×10^{0}	3.26×10^{-3}
8	4	5.44×10^{-2}	7.36×10^{4}	3.18×10^{-3}
8	8	8.16×10^{-2}	3.16×10^{14}	3.42×10^{-3}

Table 5: Running times for AA, FE and LP for the test problems with two and three levels.

constraints of the linear program used by LP have only two non-zero entries and it appears that linear programming software can effectively exploit this sparse structure to obtain quite satisfactory running times for LP. For these problem instances, we observe that LP is faster than the greedy algorithm in Li and Rusmevichientong (2014). Thus, our results indicate that LP is a strong approach when the tree has two levels, but we emphasize that LP cannot deal with trees with more than two levels. For small test problems with four or eight products, the running times for FE can be competitive to the running times for AA, but the running times for FE quickly deteriorate as the number of products increases.

D. Extension to Multi-Period Capacity Allocation

In this section, we discuss the extension of our results to the multi-period capacity allocation model of Talluri and van Ryzin (2004). The model of Talluri and van Ryzin (2004) is one of the pioneering works demonstrating the importance of incorporating choice behavior into operational decisions. Let us briefly review the setup for this problem. We have an initial capacity of C seats on a flight leg that must be allocated over T periods. There are n products (fare classes) that can be offered to customers, indexed by $\{1, \dots, n\}$. If we sell one ticket for fare class ℓ, then we generate a revenue of r_ℓ. In each period, based on the remaining capacity, we must find an assortment of fare classes to offer to an arriving customer, who chooses a fare class from the assortment according to the d-level nested logit model. The goal is to determine the revenue-maximizing policy for offering assortments of fare classes over the selling horizon. For simplicity, we assume that there is exactly one customer arriving in each period, but all of our results extend to the case in which there is a positive probability that no customer shows up in a period.

We show that, under the d-level nested logit model, a nested control policy is optimal. In other words, we show that as we have less remaining capacity in a period, we offer a smaller assortment of fare classes. An important managerial implication of this result is that the optimal control policy can be implemented by using protection levels, which is the standard tool in traditional revenue management systems. That is, for each fare class ℓ, there exists a protection level $c^*_{\ell t}$ such that if the remaining capacity in period t is less than $c^*_{\ell t}$, then it is optimal not to make fare class ℓ available in this period. Due to the optimality of a protection level policy, the d-level nested logit model can be easily integrated with the existing revenue management controls.

To establish the optimality of a nested control policy, for each $x \in \{0, 1, \ldots, C\}$, let $J_t(x)$ denote the maximum expected revenue when we have x units of capacity and t periods remaining in the selling horizon. Under the d-level nested logit model, if we offer the assortment S of fare classes, then the probability that a customer chooses fare class ℓ is $\theta_\ell(S)$ as given in (8). In this case, $J_t(\cdot)$ satisfies the dynamic programming equation

$$\begin{aligned} J_t(x) &= \max_{S \subseteq \{1,\ldots,n\}} \sum_{\ell \in S} \theta_\ell(S)\,(r_\ell + J_{t-1}(x-1)) + \theta_0(S) J_{t-1}(x) \\ &= \max_{S \subseteq \{1,\ldots,n\}} \left\{ \sum_{\ell \in S} \theta_\ell(S)\,(r_\ell - \Delta J_{t-1}(x)) \right\} + J_{t-1}(x)\ , \end{aligned}$$

where $\Delta J_{t-1}(x) = J_{t-1}(x) - J_{t-1}(x-1)$ denotes the marginal value of capacity. The boundary conditions of the dynamic program are $J_0(x) = 0$ for all x and $J_t(0) = 0$ for all t. Let $S^*_t(x)$ denote the optimal solution in the last problem on the right side above, corresponding to the optimal assortment to offer when we have x units of remaining inventory and t periods to go. The main result of this section is stated in the following theorem.

Theorem D.1 (Nested Policy). *For the multi-period capacity allocation problem, there exists an optimal policy such that for each t, $S^*_t(x)$ is non-decreasing in x, satisfying $S^*_t(x-1) \subseteq S^*_t(x)$. Moreover, for each x, $S^*_t(x)$ is non-increasing in t, satisfying $S_t(x) \subseteq S_{t-1}(x)$.*

The first part of the above theorem shows that we offer a smaller assortment when we have less capacity in a period. In this case, letting $c^*_{\ell t}$ be the smallest x such that $\ell \in S^*_t(x)$, if the remaining capacity in period t is less than $c^*_{\ell t}$, then it is optimal not to offer fare class ℓ in this period. Therefore, we can implement the resulting optimal policy by using protection level policies. On the other hand, the second part of the above theorem shows that we offer a larger assortment when we have fewer periods remaining in the selling horizon.

We define some additional notation to prove Theorem D.1. For each $\delta \geq 0$ and each assortment S, let $f^\delta(S)$ denote the expected revenue under assortment $S = (S_j : j \in \mathsf{V})$ when the revenue of every product is increased by δ; that is,

$$f^\delta(S) \equiv \sum_{\ell \in S} (r_\ell + \delta)\theta_\ell(S) = \sum_{\ell \in S} r_\ell \, \theta_\ell(S) + \delta \sum_{\ell \in S} \theta_\ell(S) = f^0(S) + \delta \sum_{\ell \in S} \theta_\ell(S) \ ,$$

and let S^δ denote an assortment that maximizes $f^\delta(\cdot)$; that is, $S^\delta = \arg\max_{S \subseteq \{1,\ldots,n\}} f^\delta(S)$.

The proof of Theorem D.1 uses lemmas that establish properties of the collection $\mathcal{A}_j$ in the assortment optimization algorithm in Section 5. Throughout this section, we recall that $\widehat{S}_k(u)$ is an optimal solution to problem (5) and the collection of subsets $\mathcal{A}_j$ is constructed as in (6). In the next lemma, we show that the subsets in the collection $\mathcal{A}_j$ can be ordered such that one subset is included in another one, naturally with the exception of the largest subset. This lemma becomes useful in the proof of Lemma D.4.

Lemma D.2. *For any node j, consider the collection $\mathcal{A}_j = \{\widehat{S}^q_j : q = 1, \ldots, Q_j\}$ constructed as in (6). The subsets in this collection can be ordered such that $\widehat{S}^1_j \subseteq \widehat{S}^2_j \subseteq \ldots \subseteq \widehat{S}^{Q_j}_j$.*

Proof. We show the result by induction on the depth of node j. For leaf node j, we have $\mathcal{A}_j = \{\varnothing, \{j\}\}$. Since $\varnothing \subseteq \{j\}$, the result holds for leaf nodes. Assume that the result holds for all nodes at depth $h+1$ and consider node j at depth h. By the inductive hypothesis, for all $k \in \mathsf{Children}(j)$, the subsets in $\mathcal{A}_k$ can be ordered such that $\widehat{S}^1_k \subseteq \widehat{S}^2_k \subseteq \ldots \subseteq \widehat{S}^{Q_k}_k$. For all $k \in \mathsf{Children}(j)$, we claim that the optimal solution to problem (5) becomes a larger subset as u gets smaller.

Claim 1: Letting $\widehat{S}_k(u)$ be an optimal solution to problem (5), if $u > u'$, then $\widehat{S}_k(u) \subseteq \widehat{S}_k(u')$.

By the definition of $\widehat{S}_k(u)$, we have $V_k(\widehat{S}_k(u))\,(R_k(\widehat{S}_k(u) - u) \geq V_k(\widehat{S}_k(u'))\,(R_k(\widehat{S}_k(u') - u)$ and $V_k(\widehat{S}_k(u'))\,(R_k(\widehat{S}_k(u') - u') \geq V_k(\widehat{S}_k(u))\,(R_k(\widehat{S}_k(u) - u')$. Adding these two inequalities yields $V_k(\widehat{S}_k(u'))\,(u - u') \geq V_k(\widehat{S}_k(u))\,(u - u')$. Using $u > u'$, we get $V_k(\widehat{S}_k(u')) \geq V_k(\widehat{S}_k(u))$. Noting problem (5), $S_k(u)$ and $\widehat{S}_k(u')$ are both in $\mathcal{A}_k$ by definition and since the subsets in the collection $\mathcal{A}_k$ satisfy $\widehat{S}^1_k \subseteq \widehat{S}^2_k \subseteq \ldots \subseteq \widehat{S}^{Q_k}_k$ by the inductive hypothesis, we must have either $S_k(u) \subseteq \widehat{S}_k(u')$ or $S_k(u) \supseteq \widehat{S}_k(u')$. Since we have $V_k(\widehat{S}_k(u')) \geq V_k(\widehat{S}_k(u))$, the preference weight under the subset

$\widehat{S}_k(u')$ is larger. Thus, the subset $\widehat{S}_k(u')$ must be larger than $\widehat{S}_k(u)$, establishing the claim.

By the claim above, $\widehat{S}_k(u) \subseteq \widehat{S}_k(u')$ for all $k \in \mathsf{Children}(j)$, so that $\dot{\bigcup}_{k \in \mathsf{Children}(j)} \widehat{S}_k(u) \subseteq \dot{\bigcup}_{k \in \mathsf{Children}(j)} \widehat{S}_k(u')$. By (6), $\mathcal{A}_j$ is constructed as $\mathcal{A}_j = \left\{ \dot{\bigcup}_{k \in \mathsf{Children}(j)} \widehat{S}_k(u) : u \in \mathbb{R} \right\}$, so the subsets in the collection $\mathcal{A}_j$ can be ordered such that one subset includes the other one, as desired. $\square$

By Claim 1 in Section 5, the collection $\mathcal{A}_j$ includes an optimal solution to the local problem at node j. In the next lemma, we show that the collection $\mathcal{A}_j$ still includes an optimal solution to the local problem at node j even after increasing the revenues associated with the products by δ.

Lemma D.3. *For any node j, consider the collection $\mathcal{A}_j$ constructed as in* (6)*. This collection includes an optimal solution to the local problem at node j even when we increase the revenues associated with the products by any $\delta \geq 0$.*

Proof. The solution to the local problem at a leaf node j is always either $\{j\}$ or $\varnothing$. Thus, $\mathcal{A}_j = \{\{j\}, \varnothing\}$ always includes an optimal solution to the local problem at a leaf node j. Consider a non-leaf node j. For each $k \in \mathsf{Children}(j)$, we use $R^\delta_k(S_k)$ to denote the expected revenue obtained from a customer at node k given that we offer the assortment $S = (S_\ell : \ell \in \mathsf{V})$ and we increase the revenues of the products by δ. Since the expected revenue at each node is a convex combination of the product revenues, it follows that $R^\delta_k(S_k) = R_k(S_k) + \delta$. In this case, when we increase the revenues of the products by δ, the local problem at node k is given by

$$\max_{S_k \subseteq N_k} V_k(S_k) \left(R^\delta_k(S_k) - \mathtt{t}^\delta_{\mathsf{Parent}(k)} \right) \quad = \quad \max_{S_k \subseteq N_k} V_k(S_k) \left(R_k(S_k) - (\mathtt{t}^\delta_{\mathsf{Parent}(k)} - \delta) \right) \ .$$

In the expression above, letting $S^\delta = (S^\delta_\ell : \ell \in \mathsf{V})$ be an optimal solution to problem (3) when we increase the revenues of the products by δ, the scalars $(\mathtt{t}^\delta_j : j \in \mathsf{V})$ are given by

$$\mathtt{t}^\delta_j = \max \left\{ \mathtt{t}^\delta_{\mathsf{Parent}(j)} \ , \ \eta_j \, \mathtt{t}^\delta_{\mathsf{Parent}(j)} + (1 - \eta_j) \, R^\delta_j(S^\delta_j) \right\} \ ,$$

with the convention that $\mathtt{t}^\delta_{\mathsf{Parent(root)}} = 0$. Comparing the problem above with (5) and noting that we use $\widehat{S}_k(u)$ to denote an optimal solution to problem (5), it follows that $\widehat{S}_k(\mathtt{t}^\delta_{\mathsf{Parent}(k)} - \delta)$ is an optimal solution to the local problem at node k when we increase the revenues of the products by δ. In this case, by Theorem 4.2, $\dot{\bigcup}_{k \in \mathsf{Children}(j)} \widehat{S}_k(\mathtt{t}^\delta_{\mathsf{Parent}(k)} - \delta)$ is an optimal solution to the local problem at node j when we increase the revenues of the products by δ. By (6), $\mathcal{A}_j$ is constructed as $\mathcal{A}_j = \left\{ \dot{\bigcup}_{k \in \mathsf{Children}(j)} \widehat{S}_k(u) : u \in \mathbb{R} \right\}$, so $\dot{\bigcup}_{k \in \mathsf{Children}(j)} \widehat{S}_k(\mathtt{t}^\delta_{\mathsf{Parent}(k)} - \delta) \in \mathcal{A}_j$. Thus, it follows that the collection $\mathcal{A}_j$ includes an optimal solution to the local problem at node j even when we increase the revenues of the products by any δ. $\square$

The next lemma shows that when we increase the revenues of the products by a positive constant, the optimal assortment becomes larger.

Lemma D.4 (Larger Revenues Lead to Larger Assortments). *For all $\delta \geq 0$, $S^0 \subseteq S^\delta$.*

Proof. By Lemma D.3, the collection $\mathcal{A}_{\mathsf{root}}$ includes an optimal solution to the local problem at root when we increase the revenues of the products by any amount $\delta \geq 0$. Furthermore, the local problem at root is equivalent to the assortment problem that we want to solve. Thus, noting that S^0 is the optimal assortment and S^δ is the optimal assortment when we increase the revenues of the products by δ, both S^0 and S^δ are in $\mathcal{A}_{\mathsf{root}}$. On the other hand, by Lemma D.2, if S^0 and S^δ are in $\mathcal{A}_{\mathsf{root}}$, then we must have $S^0 \subseteq S^\delta$ or $S^\delta \subseteq S^0$. Given an assortment $S = (S_j : j \in \mathsf{V})$, we use $\theta_\ell(S)$ to denote the probability that a customer purchases product ℓ under the d-level nested logit model. By the definitions of S^0 and S^δ, we have $\sum_{\ell \in S^0} \theta_\ell(S^0)\, r_\ell \;\geq\; \sum_{\ell \in S^\delta} \theta_\ell(S^\delta)\, r_\ell$ and $\sum_{\ell \in S^\delta} \theta_\ell(S^\delta)\,(r_\ell + \delta) \geq \sum_{\ell \in S^0} \theta_\ell(S^0)\,(r_\ell + \delta)$. Adding these two inequalities yields $\sum_{\ell \in S^\delta} \theta_\ell(S^\delta) \geq \sum_{\ell \in S^0} \theta_\ell(S^0)$, indicating that when we offer the assortment S^δ, the probability that a customer purchases a product within this assortment is larger when compared with the case when we offer the assortment S^0. It is straightforward to check that under the d-level nested logit model, if we offer a larger assortment, then the probability that a customer makes a purchase within this assortment gets larger. At the beginning of the proof, we show that we have either $S^0 \subseteq S^\delta$ or $S^\delta \subseteq S^0$. Thus, noting that the probability that a customer makes a purchase is larger when we offer the assortment S^δ, it must the case that $S^0 \subseteq S^\delta$. □

Proof of Theorem D.1

Proof. Lemmas 4 and 5 in Talluri and van Ryzin (2004) show that the value functions are concave in the remaining capacity and the first differences of the value functions decrease as we approach the end of the selling horizon; that is, $\Delta J_t(x) \leq \Delta J_t(x-1)$ and $\Delta J_t(x) \geq \Delta J_{t-1}(x)$ for $t = 1, 2, \ldots, T$ and $x = 1, 2, \ldots, C$. To show that $S_t^*(x-1) \subseteq S_t^*(x)$, by definition, we have

$$\begin{aligned} S_t^*(x) &= \arg\max_{S \subseteq \{1,\ldots,n\}} \sum_{\ell \in S} \theta_\ell(S)\,(r_\ell - \Delta J_{t-1}(x-1) + \Delta J_{t-1}(x-1) - \Delta J_{t-1}(x)) \\ S_t^*(x-1) &= \arg\max_{S \subseteq \{1,\ldots,n\}} \sum_{\ell \in S} \theta_\ell(S)\,(r_\ell - \Delta J_{t-1}(x-1)) \ . \end{aligned}$$

Let $\delta = \Delta J_{t-1}(x-1) - \Delta J_{t-1}(x)$. Note that $\delta \geq 0$ since $\Delta J_t(x) \leq \Delta J_t(x-1)$. Thus, applying Lemma D.4, with $S^0 = S_t^*(x-1)$ and $S^\delta = S_t^*(x)$, yields $S_t^*(x-1) \subseteq S_t^*(x)$, as desired. To show that $S_t^*(x) \subseteq S_{t-1}^*(x)$, by definition, we have

$$\begin{aligned} S_t^*(x) &= \arg \max_{S \subseteq \{1,\ldots,n\}} \sum_{\ell \in S} \theta_\ell(S) \left(r_\ell - \Delta J_{t-1}(x)\right) \\ S_{t-1}^*(x) &= \arg \max_{S \subseteq \{1,\ldots,n\}} \sum_{\ell \in S} \theta_\ell(S) \left(r_\ell - \Delta J_{t-1}(x) + \Delta J_{t-1}(x) - \Delta J_{t-2}(x)\right) \ . \end{aligned}$$

Let $\delta = \Delta J_{t-1}(x) - \Delta J_{t-2}(x)$, which is non-negative since $\Delta J_{t-1}(x) \geq \Delta J_{t-2}(x)$. Applying Lemma D.4 with $S^0 = S_t^*(x)$ and $S^\delta = S_{t-1}^*(x)$ yields $S_t^*(x) \subseteq S_{t-1}^*(x)$, as desired. □

E. Omitted Results in Section 7

E.1 Proof of Lemma 7.1

Proof. For each node $j \in \mathsf{V}$, let $F_j(\cdot) : \mathbb{R}_+^n \times \mathbb{R}_+^n \to \mathbb{R}$ be defined by: for each $\boldsymbol{p} \in \mathbb{R}_+^n$ and $\boldsymbol{q} \in \mathbb{R}_+^n$,

$$F_j(\boldsymbol{p}, \boldsymbol{q}) = V_j(\boldsymbol{p}) \left(R_j(\boldsymbol{p}) - \mathsf{u}_{\mathsf{Parent}(j)}(\boldsymbol{q})\right) \ ,$$

with the convention that $\mathsf{u}_{\mathsf{Parent(root)}}(\cdot) = 0$. Since $\eta_{\mathsf{root}} = 0$, we have $V_{\mathsf{root}}(\boldsymbol{p}) = 1$ by noting the definition of $V_j(\boldsymbol{p})$, in which case, $F_{\mathsf{root}}(\boldsymbol{p}, \boldsymbol{q}) = R_{\mathsf{root}}(\boldsymbol{p})$ for all $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}_+^n$. At any price vector $\hat{\boldsymbol{p}}$, we will now show that for every non-leaf node j and product ℓ that is a descendant of j, we have

$$\left. \frac{\partial F_j}{\partial p_\ell} (\boldsymbol{p}, \boldsymbol{q}) \right|_{(\boldsymbol{p},\boldsymbol{q}) = (\hat{\boldsymbol{p}}, \hat{\boldsymbol{p}})} = \frac{V_j(\hat{\boldsymbol{p}})}{\sum_{k \in \mathsf{Children}(j)} V_k(\hat{\boldsymbol{p}})} \times \left. \frac{\partial F_i}{\partial p_\ell} (\boldsymbol{p}, \boldsymbol{q}) \right|_{(\boldsymbol{p},\boldsymbol{q}) = (\hat{\boldsymbol{p}}, \hat{\boldsymbol{p}})} \ ,$$

where i is the child of j that is the ancestor of ℓ. To see this, note that if we let $G_k(\boldsymbol{p}) = V_k(\boldsymbol{p}) R_k(\boldsymbol{p})$, then we have

$$F_j(\boldsymbol{p}, \boldsymbol{q}) = \left(\sum_{k \in \mathsf{Children}(j)} V_k(\boldsymbol{p}) \right)^{\eta_j} \left(\frac{\sum_{k \in \mathsf{Children}(j)} G_k(\boldsymbol{p})}{\sum_{k \in \mathsf{Children}(j)} V_k(\boldsymbol{p})} - \mathsf{u}_{\mathsf{Parent}(j)}(\boldsymbol{q}) \right) \ .$$

By definition, among $(V_k(\boldsymbol{p}) : k \in \mathsf{Children}(j))$, only $V_i(\boldsymbol{p})$ depends on p_ℓ. Similarly, among $(R_k(\boldsymbol{p}) : k \in \mathsf{Children}(j))$, only $R_i(\boldsymbol{p})$ depends on p_ℓ. Thus, differentiating the expression above with respect to $\boldsymbol{p}_\ell$, it follows that

$$\begin{aligned}
\frac{\partial F_j}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q}) &= \eta_j \left(\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})\right)^{\eta_j-1} \times \frac{\partial V_i}{\partial p_\ell}(\boldsymbol{p}) \times \left(\frac{\sum_{k\in \mathsf{Children}(j)} G_k(\boldsymbol{p})}{\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})} - \mathrm{u}_{\mathsf{Parent}(j)}(\boldsymbol{q})\right) \\
&\quad + \left(\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})\right)^{\eta_j} \times \left[\frac{\frac{\partial G_i}{\partial p_\ell}(\boldsymbol{p}) \sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p}) - \frac{\partial V_i}{\partial p_\ell}(\boldsymbol{p}) \sum_{k\in \mathsf{Children}(j)} G_k(\boldsymbol{p})}{\left(\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})\right)^2}\right] \\
&= \left(\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})\right)^{\eta_j-1} \times \frac{\partial V_i}{\partial p_\ell}(\boldsymbol{p}) \times \left(\eta_j R_j(\boldsymbol{p}) - \eta_j \mathrm{u}_{\mathsf{Parent}(j)}(\boldsymbol{q})\right) \\
&\quad + \left(\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})\right)^{\eta_j-1} \times \left[\frac{\frac{\partial G_i}{\partial p_\ell}(\boldsymbol{p}) \sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p}) - \frac{\partial V_i}{\partial p_\ell}(\boldsymbol{p}) \sum_{k\in \mathsf{Children}(j)} G_k(\boldsymbol{p})}{\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})}\right] ,
\end{aligned}$$

where the second equality uses the fact that $R_j(\boldsymbol{p}) = \sum_{k\in \mathsf{Children}(j)} G_k(\boldsymbol{p}) / \sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})$. Using this identity once more, we can continue the chain of equalities above as

$$\begin{aligned}
\frac{\partial F_j}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q}) &= \left(\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})\right)^{\eta_j-1} \times \frac{\partial V_i}{\partial p_\ell}(\boldsymbol{p}) \times \left(\eta_j R_j(\boldsymbol{p}) - \eta_j \mathrm{u}_{\mathsf{Parent}(j)}(\boldsymbol{q})\right) \\
&\qquad + \left(\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})\right)^{\eta_j-1} \times \left[\frac{\partial G_i}{\partial p_\ell}(\boldsymbol{p}) - \frac{\partial V_i}{\partial p_\ell}(\boldsymbol{p}) \times R_j(\boldsymbol{p})\right] \\
&= \left(\sum_{k\in \mathsf{Children}(j)} V_k(\boldsymbol{p})\right)^{\eta_j-1} \times \left\{\frac{\partial G_i}{\partial p_\ell}(\boldsymbol{p}) - \frac{\partial V_i}{\partial p_\ell}(\boldsymbol{p}) \left[\eta_j \mathrm{u}_{\mathsf{Parent}(j)}(\boldsymbol{q}) + (1-\eta_j) R_j(\boldsymbol{p})\right]\right\} .
\end{aligned}$$

Therefore, noting that $\eta_j \mathrm{u}_{\mathsf{Parent}(j)}(\hat{\boldsymbol{p}}) + (1-\eta_j) R_j(\hat{\boldsymbol{p}}) = \mathrm{u}_j(\hat{\boldsymbol{p}})$ by definition, the expression above yields

$$\left.\frac{\partial F_j}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\right|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})} = \left(\sum_{k\in \mathsf{Children}(j)} V_k(\hat{\boldsymbol{p}})\right)^{\eta_j-1} \times \left\{\left.\frac{\partial G_i}{\partial p_\ell}(\boldsymbol{p})\right|_{\boldsymbol{p}=\hat{\boldsymbol{p}}} - \mathrm{u}_j(\hat{\boldsymbol{p}}) \left.\frac{\partial V_i}{\partial p_\ell}(\boldsymbol{p})\right|_{\boldsymbol{p}=\hat{\boldsymbol{p}}}\right\} .$$

By definition, we have $F_i(\boldsymbol{p},\boldsymbol{q}) = G_i(\boldsymbol{p}) - \mathrm{u}_j(\boldsymbol{q}) V_i(\boldsymbol{p})$, which implies that $\frac{\partial F_i}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q}) = \frac{\partial G_i}{\partial p_\ell}(\boldsymbol{p}) - \mathrm{u}_j(\boldsymbol{q}) \frac{\partial V_i}{\partial p_\ell}(\boldsymbol{p})$, and thus,

$$\begin{aligned}
\left.\frac{\partial F_j}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\right|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})} &= \left(\sum_{k\in \mathsf{Children}(j)} V_k(\hat{\boldsymbol{p}})\right)^{\eta_j-1} \times \left.\frac{\partial F_i}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\right|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})} \\
&= \frac{V_j(\hat{\boldsymbol{p}})}{\sum_{k\in \mathsf{Children}(j)} V_k(\hat{\boldsymbol{p}})} \times \left.\frac{\partial F_i}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\right|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})} ,
\end{aligned}$$

which is the desired result. To complete the proof, we recursively apply the above result starting at the root node. Recalling that $F_{\mathsf{root}}(\boldsymbol{p},\boldsymbol{q}) = R_{\mathsf{root}}(\boldsymbol{p})$ for all $\boldsymbol{p},\boldsymbol{q} \in \mathbb{R}^n_+$, we obtain

$$\begin{aligned}\frac{\partial R_{\mathsf{root}}}{\partial p_\ell}(\boldsymbol{p})\bigg|_{\boldsymbol{p}=\hat{\boldsymbol{p}}} &= \frac{\partial F_{\mathsf{root}}}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\bigg|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})} = \frac{V_{\mathsf{root}}(\hat{\boldsymbol{p}})}{\sum_{k\in\,\mathsf{Children(root)}} V_k(\hat{\boldsymbol{p}})} \times \frac{\partial F_{\mathsf{An}(\ell,1)}}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\bigg|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})} \\ &= \frac{1}{\sum_{k\in\,\mathsf{Children(root)}} V_k(\hat{\boldsymbol{p}})} \times \frac{\partial F_{\mathsf{An}(\ell,1)}}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\bigg|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})},\end{aligned}$$

where the second equality follows by $\eta_{\mathsf{root}} = 0$ so that $V_{\mathsf{root}}(\hat{\boldsymbol{p}}) = 1$. We continue the chain of inequalities above by replacing the last partial derivative by its equivalent expression to get

$$\begin{aligned}\frac{\partial R_{\mathsf{root}}}{\partial p_\ell}(\boldsymbol{p})\bigg|_{\boldsymbol{p}=\hat{\boldsymbol{p}}} &= \frac{1}{\sum_{k\in\,\mathsf{Children(root)}} V_k(\hat{\boldsymbol{p}})} \times \frac{V_{\mathsf{An}(\ell,1)}(\hat{\boldsymbol{p}})}{\sum_{k\in\,\mathsf{Children(An}(\ell,1))} V_k(\hat{\boldsymbol{p}})} \times \frac{\partial F_{\mathsf{An}(\ell,2)}}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\bigg|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})} \\ &= \frac{V_{\mathsf{An}(\ell,1)}(\hat{\boldsymbol{p}})}{\sum_{k\in\mathsf{Sibling(An}(\ell,1))} V_k(\hat{\boldsymbol{p}})} \times \frac{1}{\sum_{k\in\mathsf{Sibling(An}(\ell,2)))} V_k(\hat{\boldsymbol{p}})} \times \frac{\partial F_{\mathsf{An}(\ell,2)}}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\bigg|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})}.\end{aligned}$$

Iteratively applying the same argument, we have that

$$\frac{\partial R_{\mathsf{root}}}{\partial p_\ell}(\boldsymbol{p})\bigg|_{\boldsymbol{p}=\hat{\boldsymbol{p}}} = \left(\prod_{h=1}^{d-1} \frac{V_{\mathsf{An}(\ell,h)}(\hat{\boldsymbol{p}})}{\sum_{k\in\mathsf{Sibling(An}(\ell,h))} V_k(\hat{\boldsymbol{p}})}\right) \times \frac{1}{\sum_{k\in\mathsf{Sibling}(\ell)} V_k(\hat{\boldsymbol{p}})} \times \frac{\partial F_\ell}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q})\bigg|_{(\boldsymbol{p},\boldsymbol{q})=(\hat{\boldsymbol{p}},\hat{\boldsymbol{p}})}.$$

Finally, note that $F_\ell(\boldsymbol{p},\boldsymbol{q}) = V_\ell(p_\ell)(p_\ell - \mathtt{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{q})) = e^{\alpha_\ell - \beta_\ell p_\ell}\left(p_\ell - \mathtt{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{q})\right)$, which yields

$$\frac{\partial F_\ell}{\partial p_\ell}(\boldsymbol{p},\boldsymbol{q}) = -\beta_\ell e^{\alpha_\ell-\beta_\ell p_\ell}\left(p_\ell - \mathtt{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{q})\right) + e^{\alpha_\ell-\beta_\ell p_\ell} = -\beta_\ell V_\ell(p_\ell)\left(p_\ell - \frac{1}{\beta_\ell} - \mathtt{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{q})\right).$$

Thus,

$$\frac{\partial R_{\mathsf{root}}}{\partial p_\ell}(\boldsymbol{p})\bigg|_{\boldsymbol{p}=\hat{\boldsymbol{p}}} = -\beta_\ell\left(\prod_{h=1}^{d} \frac{V_{\mathsf{An}(\ell,h)}(\hat{\boldsymbol{p}})}{\sum_{k\in\mathsf{Sibling(An}(\ell,h))} V_k(\hat{\boldsymbol{p}})}\right)\left(p_\ell - \frac{1}{\beta_\ell} - \mathtt{u}_{\mathsf{Parent}(\ell)}(\hat{\boldsymbol{p}})\right),$$

and the desired result follows by noting that the product on the right side above gives $\theta_\ell(\hat{\boldsymbol{p}})$. □

E.2 Proof of Theorem 7.3

In the rest of this section, we give a proof for Theorem 7.3 by using a series of lemmas. The following lemma gives an upper bound on $\mathtt{t}^s_j$ for each non-leaf node j in each iteration s of the PUPD algorithm.

Lemma E.1. *Let* $\bar{\mathsf{V}} = \mathsf{V} \setminus \{1,2,\ldots,n,\mathsf{root}\}$ *denote the set of non-leaf nodes, after excluding* root*,* *and let* $\eta_{\min} = \min_{j \in \bar{\mathsf{V}}} \eta_j$*. Then, for any non-leaf node* j *in level* h *and for all* s*,*

$$\mathtt{t}_j^s \leq \mathtt{t}_{\mathsf{root}}^s + \left(1 - (\eta_{\min})^h\right) Q^s \ ,$$

where $Q^s = \max\left\{\mathtt{t}_{\mathsf{root}}^s, \ \max_{j \in \bar{\mathsf{V}}} \mathtt{t}_j^s, \ \max_{j \in \bar{\mathsf{V}}} R_j^s\right\}$*.*

Proof: The result is trivially true for root. If j is a non-leaf node in level 1, then by definition,

$$\mathtt{t}_j^s = \max\left\{\mathtt{t}_{\mathsf{root}}^s, \ \eta_j \mathtt{t}_{\mathsf{root}}^s + (1-\eta_j) R_j^s\right\} = \eta_j \mathtt{t}_{\mathsf{root}}^s + (1-\eta_j) \max\left\{\mathtt{t}_{\mathsf{root}}^s, R_j^s\right\} \leq \mathtt{t}_{\mathsf{root}}^s + (1-\eta_{\min}) Q^s \ ,$$

where the last inequality follows from the definition of $\eta_{\min}$ and Q^s. Suppose the result is true for all non-leaf nodes in level $h-1$. For any non-leaf node j in level h,

$$\begin{aligned}
\mathtt{t}_j^s &= \max\left\{\mathtt{t}_{\mathsf{Parent}(j)}^s, \ \eta_j \mathtt{t}_{\mathsf{Parent}(j)}^s + (1-\eta_j) R_j^s\right\} \\
&= \eta_j \mathtt{t}_{\mathsf{Parent}(j)}^s + (1-\eta_j) \max\left\{\mathtt{t}_{\mathsf{Parent}(j)}^s, R_j^s\right\} \\
&\leq \eta_j \left(\mathtt{t}_{\mathsf{root}}^s + \left(1 - (\eta_{\min})^{h-1}\right) Q^s\right) + (1-\eta_j) Q^s \\
&= \eta_j \mathtt{t}_{\mathsf{root}}^s + \left(1 - \eta_j (\eta_{\min})^{h-1}\right) Q^s \\
&\leq \mathtt{t}_{\mathsf{root}}^s + \left(1 - (\eta_{\min})^h\right) Q^s \ ,
\end{aligned}$$

where the fist inequality follows from the inductive hypothesis and the definition of Q^s. This completes the proof. □

The following lemma establishes a relationship between the expected revenues in consecutive iterations of the PUPD algorithm. Its proof follows from an argument similar to the one in the proof of Lemma A.1.

Lemma E.2. *Assume that the price vectors* $\{\boldsymbol{p}^s : s = 0, 1, 2, \ldots\}$ *are generated by the PUPD algorithm. For each non-leaf node* j*, we have*

$$R_j^{s+1} \ \geq \ \frac{1}{\eta_j}\left(\eta_j + \frac{V_j^s}{V_j^{s+1}} - 1\right)\left(R_j^s - \mathtt{t}_j^s\right)^+ + \mathtt{t}_j^s \ ,$$

where $V_j^s = V_j(\boldsymbol{p}^s)$ *and* $V_j^{s+1} = V_j\left(\boldsymbol{p}^{s+1}\right)$*.*

Proof: We will prove the result by induction on the depth of node j, starting with nodes at depth $d-1$. If node j is at depth $d-1$, then it is the parent of a leaf node. Note the inequality in the lemma is trivially true if $R_j^s \leq \mathtt{t}_j^s$. In particular, by our construction of the PUPD algorithm, for each $\ell \in \mathsf{Children}(j)$, $p_\ell^{s+1} \geq \mathtt{t}^s_{\mathsf{Parent}(\ell)} = \mathtt{t}_j^s$, in which case, since R_j^{s+1} is a convex combination of $(p_\ell^{s+1} : \ell \in N_j)$, we have $R_j^{s+1} \geq \mathtt{t}_j^s$ and the lemma follows. So, assume that $R_j^s > \mathtt{t}_j^s$. By definition of the PUPD algorithm, for each $\ell \in \mathsf{Children}(j)$, $p_\ell^{s+1} = \frac{1}{\beta_\ell} + \mathtt{t}_j^s = \arg\max_{x \geq 0} V_\ell(x)\left(x - \mathtt{t}_j^s\right)$, so for each $\ell \in \mathsf{Children}(j)$,

$$V_\ell^{s+1}\left(R_\ell^{s+1} - \mathtt{t}_j^s\right) \;\geq\; V_\ell^s\left(R_\ell^s - \mathtt{t}_j^s\right) \;.$$

Adding this inequality above over all $\ell \in \mathsf{Children}(j)$, we get

$$\begin{aligned}
\left(\sum_{\ell \in \mathsf{Children}(j)} V_\ell^{s+1}\right)\left(R_j^{s+1} - \mathtt{t}_j^s\right) &= \sum_{\ell \in \mathsf{Children}(j)} V_\ell^{s+1}\left(R_\ell^{s+1} - \mathtt{t}_j^s\right) \\
&\geq \sum_{\ell \in \mathsf{Children}(j)} V_\ell^s\left(R_\ell^s - \mathtt{t}_j^s\right) = \left(\sum_{\ell \in \mathsf{Children}(j)} V_\ell^s\right)\left(R_j^s - \mathtt{t}_j^s\right) \;,
\end{aligned}$$

where the two equalities above use the fact that $R_j^{s+1} = \sum_{k \in \mathsf{Children}(j)} V_k^{s+1} R_k^{s+1} / \sum_{k \in \mathsf{Children}(j)} V_k^{s+1}$ and $R_j^s = \sum_{k \in \mathsf{Children}(j)} V_k^s R_k^s / \sum_{k \in \mathsf{Children}(j)} V_k^s$. Noting that $V_j^{s+1} = \left(\sum_{k \in \mathsf{Children}(j)} V_j^{s+1}\right)^{\eta_j}$ and $V_j^s = \left(\sum_{k \in \mathsf{Children}(j)} V_j^s\right)^{\eta_j}$, we write the above inequality as

$$\left(V_j^{s+1}\right)^{1/\eta_j}\left(R_j^{s+1} - \mathtt{t}_j^s\right) \;\geq\; \left(V_j^s\right)^{1/\eta_j}\left(R_j^s - \mathtt{t}_j^s\right) = \left(V_j^s\right)^{1/\eta_j}\left(R_j^s - \mathtt{t}_j^s\right)^+ \;,$$

which implies that

$$R_j^{s+1} \;\geq\; \left(\frac{V_j^s}{V_j^{s+1}}\right)^{1/\eta_j}\left(R_j^s - \mathtt{t}_j^s\right)^+ + \mathtt{t}_j^s \;\geq\; \frac{1}{\eta_j}\left(\eta_j + \frac{V_j^s}{V_j^{s+1}} - 1\right)\left(R_j^s - \mathtt{t}_j^s\right)^+ + \mathtt{t}_j^s \;,$$

where the last inequality follows from the fact that the function $q \mapsto q^{1/\eta_j}$ is convex whose derivative at 1 is $1/\eta_j$, and thus, $q^{1/\eta_j} \geq 1 + \frac{1}{\eta_j}(q-1) = \frac{1}{\eta_j}(\eta_j + q - 1)$ for all $q \in \mathbb{R}_+$. This completes the base case.

Now, suppose the result holds for every node at depth $h+1$. Consider a node j at depth h. We want to show that

$$R_j^{s+1} \;\geq\; \frac{1}{\eta_j}\left(\eta_j + \frac{V_j^s}{V_j^{s+1}} - 1\right)\left(R_j^s - \mathtt{t}_j^s\right)^+ + \mathtt{t}_j^s \;.$$

The inequality is trivially true if $R_j^s \leq \mathtt{t}_j^s$. In particular, due to the way we compute the scalars $(\mathtt{t}_j^s : j \in \mathsf{V})$ in the PUPD algorithm, for each node ℓ that is a descendent of node j, we have $\mathtt{t}^s_{\mathsf{Parent}(\ell)} \;\geq\; \mathtt{t}_j^s$. Thus, for each leaf node (or product) ℓ that is a descendent of node j, our computation of the prices in the PUPD algorithm yields $p_\ell^{s+1} \;\geq\; \mathtt{t}^s_{\mathsf{Parent}(\ell)} \;\geq\; \mathtt{t}_j^s$. Since R_j^{s+1} is a convex combination of the prices p_ℓ^{s+1} at the leaf nodes $\ell \in N_j$ that are descendants of node j, we get $R_j^{s+1} \geq \min_{\ell \in N_j} p_\ell^{s+1} \geq \mathtt{t}_j^s$ and the inequality above immediately follows. So, in the rest of the proof, we assume that $R_j^s > \mathtt{t}_j^s$. We now claim that for every node $k \in \mathsf{Children}(j)$,

$$V_k^{s+1}\left(R_k^{s+1} - \mathtt{t}_j^s\right) \;\geq\; V_k^s\left(R_k^s - \mathtt{t}_j^s\right) \;.$$

The claim is trivially true if $R_k^s \leq \mathtt{t}_j^s$ because by our construction of the PUPD algorithm we have $R_k^{s+1} \geq \mathtt{t}_k^s \geq \mathtt{t}_{\mathsf{Parent}(k)}^s = \mathtt{t}_j^s$, which can be shown by using the same argument that we just used above. So, consider the case in which $R_k^s > \mathtt{t}_j^s$. By definition of $\mathtt{t}_k^s$, we have that

$$\mathtt{t}_k^s = \max\left\{\, \mathtt{t}_j^s \;,\; \eta_k \mathtt{t}_j^s + (1-\eta_k) R_k^s \,\right\} = \eta_k \mathtt{t}_j^s + (1-\eta_k) R_k^s \;,$$

which implies that $R_k^s - \mathtt{t}_k^s = \eta_k\left(R_k^s - \mathtt{t}_j^s\right)$ and $\mathtt{t}_k^s - \mathtt{t}_j^s = (1-\eta_k)\left(R_k^s - \mathtt{t}_j^s\right)$. Since node $k \in \mathsf{Children}(j)$ is at depth $h+1$, it follows from the inductive hypothesis that

$$\begin{aligned} R_k^{s+1} - \mathtt{t}_j^s \;&\geq\; \frac{1}{\eta_k}\left(\eta_k + \frac{V_k^s}{V_k^{s+1}} - 1\right)\left(R_k^s - \mathtt{t}_k^s\right)^+ + \left(\mathtt{t}_k^s - \mathtt{t}_j^s\right) \\ &=\; \left(\eta_k + \frac{V_k^s}{V_k^{s+1}} - 1\right)\left(R_k^s - \mathtt{t}_j^s\right) + (1-\eta_k)\left(R_k^s - \mathtt{t}_j^s\right) \;=\; \frac{V_k^s}{V_k^{s+1}}\left(R_k^s - \mathtt{t}_j^s\right) \;, \end{aligned}$$

which establishes the claim.

To finish the proof, since the claim we established holds for every $k \in \mathsf{Children}(j)$, adding this inequality over all $\ell \in \mathsf{Children}(j)$, we have that

$$\begin{aligned} \left(\sum_{k \in \mathsf{Children}(j)} V_k^{s+1}\right)\left(R_j^{s+1} - \mathtt{t}_j^s\right) \;&=\; \sum_{k \in \mathsf{Children}(j)} V_k^{s+1}\left(R_k^{s+1} - \mathtt{t}_j^s\right) \\ &\geq\; \sum_{k \in \mathsf{Children}(j)} V_k^s\left(R_k^s - \mathtt{t}_j^s\right) = \left(\sum_{k \in \mathsf{Children}(j)} V_k^s\right)\left(R_j^s - \mathtt{t}_j^s\right) \;, \end{aligned}$$

or equivalently,

$$\left(V_j^{s+1}\right)^{1/\eta_j}\left(R_j^{s+1} - \mathtt{t}_j^s\right) \;\geq\; \left(V_j^s\right)^{1/\eta_j}\left(R_j^s - \mathtt{t}_j^s\right) = \left(V_j^s\right)^{1/\eta_j}\left(R_j^s - \mathtt{t}_j^s\right)^+ \quad .$$

Therefore,

$$R_j^{s+1} \;\geq\; \left(\frac{V_j^s}{V_j^{s+1}}\right)^{1/\eta_j}\left(R_j^s - \mathtt{t}_j^s\right)^+ + \mathtt{t}_j^s \;\geq\; \frac{1}{\eta_j}\left(\eta_j + \frac{V_j^s}{V_j^{s+1}} - 1\right)\left(R_j^s - \mathtt{t}_j^s\right)^+ + \mathtt{t}_j^s \;,$$

where the final inequality follows because the function $q \mapsto q^{1/\eta_j}$ is convex whose derivative at 1 is $1/\eta_j$, and thus, $q^{1/\eta_j} \geq 1 + \frac{1}{\eta_j}(q-1) = \frac{1}{\eta_j}(\eta_j + q - 1)$ for all $q \in \mathbb{R}_+$. This completes the induction and the result follows. $\square$

The following lemma builds on the previous one to show the monotonicity of the expected revenues in consecutive iterations of the PUPD algorithm.

Lemma E.3. *Assume that the price vectors $\{\boldsymbol{p}^s : s = 0, 1, 2, \ldots\}$ are generated by the PUPD algorithm. If $p_\ell^{s+1} \geq p_\ell^s$ for every leaf node ℓ, then for each non-leaf node j, $R_j^{s+1} \geq R_j^s$.*

Proof: It follows from Lemma E.2 that

$$R_j^{s+1} - R_j^s \quad \geq \quad \frac{1}{\eta_j}\left(\eta_j + \frac{V_j^s}{V_j^{s+1}} - 1\right)\left(R_j^s - \mathtt{t}_j^s\right)^+ - \left(R_j^s - \mathtt{t}_j^s\right) \ . \tag{13}$$

There are two cases to consider: $R_j^s \leq \mathtt{t}_j^s$ and $R_j^s > \mathtt{t}_j^s$. When $R_j^s \leq \mathtt{t}_j^s$, it follows immediately from the above inequality that $R_j^{s+1} \geq R_j^s$. So, consider the case where $R_j^s > \mathtt{t}_j^s$. In this case, it follows from the definition of $\mathtt{t}_j^s$ in the PUPD algorithm that $R_j^s > \mathtt{t}_j^s \geq \mathtt{t}_{\mathsf{Parent}(j)}^s$, which implies that

$$\mathtt{t}_j^s = \max\left\{\mathtt{t}_{\mathsf{Parent}(j)}^s \ , \ \eta_j \mathtt{t}_{\mathsf{Parent}(j)}^s + (1-\eta_j)R_j^s\right\} = \eta_j \mathtt{t}_{\mathsf{Parent}(j)}^s + (1-\eta_j)R_j^s \ ,$$

and thus, $R_j^s - \mathtt{t}_j^s = \eta_j\left(R_j^s - \mathtt{t}_{\mathsf{Parent}(j)}^s\right)$. Therefore, it follows from (13) that

$$\begin{aligned} R_j^{s+1} - R_j^s \quad &\geq \quad \left(\eta_j + \frac{V_j^s}{V_j^{s+1}} - 1\right)\left(R_j^s - \mathtt{t}_{\mathsf{Parent}(j)}^s\right) - \eta_j\left(R_j^s - \mathtt{t}_{\mathsf{Parent}(j)}^s\right) \\ &= \quad \left(\frac{V_j^s}{V_j^{s+1}} - 1\right)\left(R_j^s - \mathtt{t}_{\mathsf{Parent}(j)}^s\right) \ . \end{aligned}$$

By our hypothesis, we have that $p_\ell^{s+1} \geq p_\ell^s$ for every product ℓ. As prices of the products increase, the corresponding preference weights decrease. Thus, we have $V_j^s \geq V_j^{s+1}$, so that $\frac{V_j^s}{V_j^{s+1}} - 1 \geq 0$. Since $R_j^s > \mathtt{t}_{\mathsf{Parent}(j)}^s$, the chain of inequalities above yields $R_j^{s+1} \geq R_j^s$. $\square$

Before we turn to the proof for Theorem 7.3, we give the following corollary to the last lemma showing that the scalars $(\mathtt{t}_j^s : j \in \mathsf{V})$ are monotonically increasing in consecutive iterations.

Corollary E.4. *Assume that the price vectors $\{\boldsymbol{p}^s : s = 0, 1, 2, \ldots\}$ are generated by the PUPD algorithm. If $\mathtt{t}_j^s \geq \mathtt{t}_j^{s-1}$ for every non-leaf node j, then $\mathtt{t}_j^{s+1} \geq \mathtt{t}_j^s$ for every non-leaf node j.*

Proof: By definition, we have $p_\ell^s = \frac{1}{\beta_\ell} + \mathtt{t}^{s-1}_{\mathsf{Parent}(\ell)}$ and $p_\ell^{s+1} = \frac{1}{\beta_\ell} + \mathtt{t}^{s}_{\mathsf{Parent}(\ell)}$ for every leaf node ℓ. Thus, our hypothesis implies that $p_\ell^{s+1} \geq p_\ell^s$ for every leaf node ℓ. Then, it follows from Lemma E.3 that $R_j^{s+1} \geq R_j^s$ for every non-leaf node j. So, $\mathtt{t}^{s+1}_{\mathsf{root}} = R^{s+1}_{\mathsf{root}} \geq R^{s}_{\mathsf{root}} = \mathtt{t}^{s}_{\mathsf{root}}$. In this case, for every node $j \in \mathsf{Children}(\mathsf{root})$, we get

$$\mathtt{t}_j^{s+1} \;=\; \max\left\{\mathtt{t}^{s+1}_{\mathsf{root}}\ ,\ \eta_j \mathtt{t}^{s+1}_{\mathsf{root}} + (1-\eta_j) R_j^{s+1}\right\} \geq \max\left\{\mathtt{t}^{s}_{\mathsf{root}}\ ,\ \eta_j \mathtt{t}^{s}_{\mathsf{root}} + (1-\eta_j) R_j^{s}\right\} \;=\; \mathtt{t}_j^s\ ,$$

establishing the desired monotonicity result for the first-level nodes. Having establish monotonicity for the first-level nodes, using exactly the same argument as above, we can show that $\mathtt{t}_j^{s+1} \geq \mathtt{t}_j^s$ for all nodes j in level two. By induction, the result then holds for all non-leaf nodes. □

The next lemma shows that the revenues at each node generated under the PUPD algorithm are always bounded.

Lemma E.5. *Let* $\bar{\mathsf{V}} = \mathsf{V} \setminus \{1, 2, \ldots, n, \mathsf{root}\}$ *denote the set of non-leaf nodes, after excluding* root*, and let* $Q^s = \max\left\{\mathtt{t}^s_{\mathsf{root}},\ \max_{j\in\bar{\mathsf{V}}} \mathtt{t}_j^s,\ \max_{j\in\bar{\mathsf{V}}} R_j^s\right\}$*. Then, the sequence* $\{Q^s : s = 0, 1, 2, \ldots\}$ *is bounded.*

Proof: Since $\boldsymbol{p}^0 = \mathbf{0}$, $R_j(\boldsymbol{p}^0) = \mathtt{t}_j^0 = 0$ for all j. Thus, $\mathtt{t}_j^1 \geq \mathtt{t}_j^0$ for every non-leaf node j. By Corollary E.4, the scalars $(\mathtt{t}_j^s : j \in \mathsf{V})$ are always monotonically increasing in consecutive iterations of the PUPD algorithm. In particular, $\{\mathtt{t}^s_{\mathsf{root}} : s = 0, 1, \ldots\}$ is an increasing sequence. Since $\mathtt{t}^s_{\mathsf{root}} = R_{\mathsf{root}}(\boldsymbol{p}^s) = \frac{\sum_{j\in \mathsf{Children}(\mathsf{root})} R_j^s V_j^s}{v_0 + \sum_{j\in \mathsf{Children}(\mathsf{root})} V_j^s}$, the sequence $\{\mathtt{t}^s_{\mathsf{root}} : s = 0, 1, \ldots\}$ is also bounded because the no-purchase option 0 prevents the revenue from becoming infinity. The boundedness implies that $\lim_{s\to\infty} \mathtt{t}^s_{\mathsf{root}} = \bar{\mathtt{t}}_{\mathsf{root}}$ for some $\bar{\mathtt{t}}_{\mathsf{root}} \in \mathbb{R}_+$.

Let $\beta_{\min} = \min_{\ell=1,\ldots,n} \beta_\ell$, $\eta_{\min} = \min_{j\in\bar{\mathsf{V}}} \eta_j$, $A = \frac{1}{\beta_{\min}} + \bar{\mathtt{t}}_{\mathsf{root}}$, and $\alpha = 1 - (\eta_{\min})^{d-1}$. Note that A is finite because $\beta_{\min} > 0$, and $0 \leq \alpha < 1$ because $0 < \eta_{\min} \leq 1$.

Claim: For all s, $Q^{s+1} \leq A + \alpha Q^s$.

To establish the claim, first consider any non-leaf node j in level $d-1$, so j is a parent of some leaf nodes. Since R_j^{s+1} is the convex combination of the prices at children nodes of j in iteration $s+1$,

$$\begin{aligned} R_j^{s+1} \;&\leq\; \max_{\ell\in \mathsf{Children}(j)} p_\ell^{s+1} \;=\; \max_{\ell\in \mathsf{Children}(j)} \left\{\frac{1}{\beta_\ell} + \mathtt{t}_j^s\right\} \;\leq\; \frac{1}{\beta_{\min}} + \mathtt{t}_j^s \\ &\leq\; \frac{1}{\beta_{\min}} + \mathtt{t}^s_{\mathsf{root}} + \left(1 - (\eta_{\min})^{d-1}\right) Q^s \;\leq\; A + \alpha Q^s\ , \end{aligned}$$

where the equality follows from the definition of the PUPD algorithm, and the next-to-last inequality follows from Lemma E.1. The final inequality follows from the fact that $\mathtt{t}^s_{\mathsf{root}} \leq \bar{\mathtt{t}}_{\mathsf{root}}$ for all s. Since the above upper bound holds for every node in level $d-1$, and the revenue at each node is a convex combination of the revenue at children nodes, it follows that $R_j^{s+1} \leq A + \alpha Q^s$ for all non-leaf nodes, including root.

We will now show that $\mathtt{t}_j^{s+1} \leq A + \alpha Q^s$ for all non-leaf node j. This is true at root because $\mathtt{t}_{\mathsf{root}}^{s+1} = R_{\mathsf{root}}^{s+1}$. Consider any node j in level 1. Then, by definition,

$$\mathtt{t}_j^{s+1} = \max\left\{\mathtt{t}_{\mathsf{root}}^{s+1},\ \eta_j \mathtt{t}_{\mathsf{root}}^{s+1} + (1-\eta_j) R_j^{s+1}\right\} \leq A + \alpha Q^s \ ,$$

where the inequality follows from the fact that both $\mathtt{t}_{\mathsf{root}}^{s+1}$ and R_j^{s+1} are bounded above by $A + \alpha Q^s$. Suppose the result is true for all non-leaf nodes in level $h-1$. For any non-leaf node j in level h,

$$\mathtt{t}_j^{s+1} = \max\left\{\mathtt{t}_{\mathsf{Parent}(j)}^{s+1},\ \eta_j \mathtt{t}_{\mathsf{Parent}(j)}^{s+1} + (1-\eta_j) R_j^{s+1}\right\} \leq A + \alpha Q^s \ ,$$

where the inequality follows from the inductive hypothesis and the fact that $R_j^{s+1} \leq A + \alpha Q^s$ for all non-leaf nodes j. This completes the induction.

Since R_j^{s+1} and $\mathtt{t}_j^{s+1}$ are bounded above by $A + \alpha Q^s$ for all non-leaf nodes j, it follows that $Q^{s+1} \leq A + \alpha Q^s$, establishing the claim. Repeated application of the claim implies that for all s,

$$Q^s \leq A\left(1 + \alpha + \alpha^2 + \cdots + \alpha^{s-1}\right) + \alpha^s Q^0 \leq \frac{A}{1-\alpha} \ ,$$

where the last inequality follows because $Q^0 = 0$, since $\boldsymbol{p}^0 = \mathbf{0}$. This is the desired result. □

Proof of Theorem 7.3

Proof: Since $\boldsymbol{p}^0 = \mathbf{0}$, $R_j(\boldsymbol{p}^0) = \mathtt{t}_j^0 = 0$ for all j. Thus, $\mathtt{t}_j^1 \geq \mathtt{t}_j^0$ for every non-leaf node j. It then follows from Corollary E.4 that the scalars $(\mathtt{t}_j^s : j \in \mathsf{V})$ are always monotonically increasing in consecutive iterations of the PUPD algorithm. By Lemma E.5, the scalars $(\mathtt{t}_j^s : j \in \mathsf{V})$ are also bounded. Thus, the sequence $\{(\mathtt{t}_j^s : j \in \mathsf{V}) : s = 0, 1, 2, \ldots\}$ must converge. Therefore, for every non-leaf node j, $\lim_{s\to\infty} \mathtt{t}_j^s = \bar{\mathtt{t}}_j$, for some $\bar{\mathtt{t}}_j$. Since the prices in the PUPD algorithm are computed as $p_\ell^s = \frac{1}{\beta_\ell} + \mathtt{t}_{\mathsf{Parent}(\ell)}^{s-1}$, the prices must also converge. It remains to show that the limit point of the prices is a stationary point of the expected revenue function $R_{\mathsf{root}}(\cdot)$.

The way we compute the scalars $(\mathtt{t}_j^s : j \in \mathsf{V})$ in the PUPD algorithm implies that $\mathtt{t}_j^s \geq \mathtt{t}_{\mathsf{Parent}(j)}^s$ so that $\mathtt{t}_j^s \geq \mathtt{t}_{\mathsf{root}}^s$ for every non-leaf node j and in every iteration s. Thus, the prices in the PUPD algorithm satisfy $p_\ell^s = \frac{1}{\beta_\ell} + \mathtt{t}_{\mathsf{Parent}(\ell)}^{s-1} \geq \frac{1}{\beta_{\max}} + \mathtt{t}_{\mathsf{root}}^{s-1}$ for each leaf node ℓ, where $\beta_{\max} = \max_{\ell=1,\ldots,n} \beta_\ell$. Since the expected revenue at each node in $\mathsf{V} \backslash \{\mathsf{root}, 0\}$ can be written as a convex combination of the prices, we get $R_j^s \geq \frac{1}{\beta_{\max}} + \mathtt{t}_{\mathsf{root}}^{s-1}$ for every $j \in \mathsf{V} \backslash \{\mathsf{root}, 0\}$. By definition, for each $j \in \mathsf{Children}(\mathsf{root})$,

$$\mathtt{t}_j^s - \mathtt{t}_{\mathsf{root}}^s \;=\; \max\{0\ ,\ (1-\eta_j)(R_j^s - \mathtt{t}_{\mathsf{root}}^s)\} \;\geq\; (1-\eta_j)\left(\frac{1}{\beta_{\max}} + \mathtt{t}_{\mathsf{root}}^{s-1} - \mathtt{t}_{\mathsf{root}}^s\right)^+ ,$$

where the last inequality follows from the fact that $R_j^s \geq \frac{1}{\beta_{\max}} + \mathtt{t}_{\mathsf{root}}^{s-1}$. Since $\mathtt{t}_{\mathsf{root}}^s$ is convergent, $\lim_{s\to\infty}\left(\mathtt{t}_{\mathsf{root}}^{s-1} - \mathtt{t}_{\mathsf{root}}^s\right) = 0$. Therefore, there exists T_0 such that for all $s \geq T_0$, $\left|\mathtt{t}_{\mathsf{root}}^{s-1} - \mathtt{t}_{\mathsf{root}}^s\right| \leq \frac{1}{2\beta_{\max}}$, which implies that $\mathtt{t}_j^s - \mathtt{t}_{\mathsf{root}}^s \geq \frac{1-\eta_j}{2\beta_{\max}} > 0$. Since $j \in \mathsf{Children}(\mathsf{root})$ is arbitrary, this means that for all $s \geq T_0$ and for every child j of root, $\mathtt{t}_j^s > \mathtt{t}_{\mathsf{root}}^s$, and thus,

$$\mathtt{t}_j^s = \max\left\{\mathtt{t}_{\mathsf{root}}^s\ ,\ \eta_j \mathtt{t}_{\mathsf{root}}^s + (1-\eta_j)R_j^s\right\} \;=\; \eta_j \mathtt{t}_{\mathsf{root}}^s + (1-\eta_j)R_j^s\ .$$

Similarly, consider an arbitrary node j in the second level. Then,

$$\mathtt{t}_j^s - \mathtt{t}_{\mathsf{Parent}(j)}^s \;=\; \max\{0\ ,\ (1-\eta_j)(\ R_j^s - \mathtt{t}_{\mathsf{Parent}(j)}^s\ )\} \;\geq\; (1-\eta_j)\left(\frac{1}{\beta_{\max}} + \mathtt{t}_{\mathsf{Parent}(j)}^{s-1} - \mathtt{t}_{\mathsf{Parent}(j)}^s\right)^+ ,$$

where the inequality follows, once again, from the fact that $R_j^s \geq \frac{1}{\beta_{\max}} + \mathtt{t}_{\mathsf{Parent}(j)}^{s-1}$. Since $\mathtt{t}_{\mathsf{Parent}(j)}^s$ is convergent, $\lim_{s\to\infty}\left(\mathtt{t}_{\mathsf{Parent}(j)}^{s-1} - \mathtt{t}_{\mathsf{Parent}(j)}^s\right) = 0$. Then, using exactly the same argument as the previous paragraph, there exists R_0 such that for all $s \geq R_0$ and for all nodes j in the second level, $\mathtt{t}_j^s > \mathtt{t}_{\mathsf{Parent}(j)}^s$, and thus,

$$\mathtt{t}_j^s = \eta_j \mathtt{t}_{\mathsf{Parent}(j)}^s + (1-\eta_j)R_j^s\ .$$

By iteratively applying exactly the same argument, we conclude that there exists Q_0 such that for all $s \geq Q_0$ and for every non-leaf node j,

$$\mathtt{t}_j^s = \eta_j \mathtt{t}_{\mathsf{Parent}(j)}^s + (1-\eta_j)R_j^s\ .$$

However, noting the definition of $\mathtt{u}_j(\boldsymbol{p})$ at the beginning of Section 7.1, this means that for all $s \geq Q_0$ and for every non-leaf node j, $\mathtt{t}_j^s = \mathtt{u}_j(\boldsymbol{p}^s)$. Therefore, for all $s \geq Q_0$, the sequence of prices at the leaf nodes is defined by: for $\ell = 1, \ldots, n$,

$$p_\ell^{s+1} = \frac{1}{\beta_\ell} + \mathtt{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p}^s)\ .$$

So, since the prices converge, the limit point $\bar{\boldsymbol{p}} = (\bar{p}_1, \ldots, \bar{p}_n)$ satisfies the following system of equations: for $\ell = 1, \ldots, n$,

$$\bar{p}_\ell = \frac{1}{\beta_\ell} + \mathtt{u}_{\mathsf{Parent}(\ell)}(\bar{\boldsymbol{p}})\ ,$$

and it follows from Corollary 7.2 that $\bar{\boldsymbol{p}}$ is a stationary point of the expected revenue function. □

E.3 Proof of Theorem 7.4

Previously, we have shown that for each leaf node ℓ

$$\frac{\partial R_{\mathsf{root}}}{\partial p_\ell}(\boldsymbol{p}) = -\theta_\ell^{\mathsf{root}}(\boldsymbol{p})\beta_\ell\left(p_\ell - \frac{1}{\beta_\ell} - \mathsf{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p})\right) .$$

Thus, it suffices to derive the partial derivative of the cost function. The following lemma gives the partial derivative of $\frac{\partial \theta_i^{\mathsf{root}}(\boldsymbol{p})}{\partial p_\ell}$.

Lemma E.6 (Partial Derivative of Probabilities)**.** *For each leaf node ℓ,*

$$\begin{aligned}&\frac{\partial \theta_i^{\mathsf{root}}(\boldsymbol{p})}{\partial p_\ell}\\ &= \theta_i^{\mathsf{root}}(\boldsymbol{p})\left\{\sum_{s=1}^{\bar{s}_{i,\ell}}\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}\left(\frac{1}{V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})} - \frac{1}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(\ell,s))}V_k(\boldsymbol{p})}\right) - \frac{\frac{\partial V_{\mathsf{An}(\ell,\bar{s}_{i,\ell}+1)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(\ell,\bar{s}_{i,\ell}+1))}V_k(\boldsymbol{p})}\right\},\end{aligned}$$

where $\bar{s}_{i,\ell} = \max\{h : \mathsf{An}(i,h) = \mathsf{An}(\ell,h)\}$*. Also, we define* $V_{\mathsf{An}(\ell,d+1)}(\cdot) = 0$*.*

Proof. Fix ℓ. Consider an arbitrary i. By definition,

$$\begin{aligned}&\frac{\partial \theta_i^{\mathsf{root}}(\boldsymbol{p})}{\partial p_\ell}\\ &= \sum_{s=1}^{d}\left(\prod_{h:1\le h\le d, h\neq s}\frac{V_{\mathsf{An}(i,h)}(\boldsymbol{p})}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,h))}V_k(\boldsymbol{p})}\right)\times\frac{\partial}{\partial p_\ell}\frac{V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))}V_k(\boldsymbol{p})}\\ &= \sum_{s=1}^{d}\prod_{h:1\le h\le d, h\neq s}\left\{\frac{V_{\mathsf{An}(i,h)}(\boldsymbol{p})}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,h))}V_k(\boldsymbol{p})}\right.\\ &\qquad\left.\times\frac{\left(\frac{\partial V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\partial p_\ell}\right)\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))}V_k(\boldsymbol{p}) - V_{\mathsf{An}(i,s)}(\boldsymbol{p})\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))}\frac{\partial V_k(\boldsymbol{p})}{\partial p_\ell}}{\left(\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))}V_k(\boldsymbol{p})\right)^2}\right\}.\end{aligned}$$

To compute the expression on the righthand side, there are 3 cases to consider.

- **Case 1:** $\mathsf{An}(\ell, s) = \mathsf{An}(i, s)$. In this case,

$$\frac{\left(\frac{\partial V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\partial p_\ell}\right) \sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p}) - V_{\mathsf{An}(i,s)}(\boldsymbol{p}) \sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} \frac{\partial V_k(\boldsymbol{p})}{\partial p_\ell}}{\left(\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,h))} V_k(\boldsymbol{p})\right)^2}$$

$$= \frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell} \left(\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p}) - V_{\mathsf{An}(i,s)}(\boldsymbol{p})\right)}{\left(\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})\right)^2}$$

$$= \frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} \times \left(1 - \frac{V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})}\right)$$

$$= \frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} - \frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} \times \frac{V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} .$$

- **Case 2:** $\mathsf{An}(\ell, s) \neq \mathsf{An}(i, s)$, but $\mathsf{An}(\ell, s) \in \mathsf{Sibling}\,(\mathsf{An}(i, s))$. In this case,

$$\frac{\left(\frac{\partial V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\partial p_\ell}\right) \sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p}) - V_{\mathsf{An}(i,s)}(\boldsymbol{p}) \sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} \frac{\partial V_k(\boldsymbol{p})}{\partial p_\ell}}{\left(\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})\right)^2}$$

$$= -\frac{V_{\mathsf{An}(i,s)}(\boldsymbol{p}) \frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{\left(\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})\right)^2}$$

$$= -\frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} \times \frac{V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} .$$

- **Case 3:** $\mathsf{An}(\ell, s) \neq \mathsf{An}(i, s)$, and $\mathsf{An}(\ell, s) \notin \mathsf{Sibling}\,(\mathsf{An}(i, s))$. In this case,

$$\frac{\left(\frac{\partial V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\partial p_\ell}\right) \sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p}) - V_{\mathsf{An}(i,s)}(\boldsymbol{p}) \sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} \frac{\partial V_k(\boldsymbol{p})}{\partial p_\ell}}{\left(\sum_{k\in\mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})\right)^2} = 0 .$$

Let $\bar{s}_{i,\ell} = \max\{h : \mathsf{An}(i, h) = \mathsf{An}(\ell, h)\}$. Note that $\bar{s}_{i,\ell}$ is always well-defined and $0 \le \bar{s}_{i,\ell} \le d$ because root is a common ancestor for both i and ℓ. Note that if $h \le \bar{s}_{i,\ell}$, then $\mathsf{An}(i, h) = \mathsf{An}(\ell, h)$. Also, we have that $\mathsf{An}(\ell, \bar{s}_{i,\ell} + 1) \in \mathsf{Sibling}(\mathsf{An}(i, \bar{s}_{i,\ell} + 1))$. Also, for all $h > \bar{s}_{i,\ell} + 1$, $\mathsf{An}(\ell, h) \notin$

$\mathsf{Sibling}\,(\mathsf{An}(i,h))$. This implies that

$$
\begin{aligned}
&\frac{\partial \theta_i^{\mathsf{root}}(\boldsymbol{p})}{\partial p_\ell} \\
&= \sum_{s=1}^{\bar{s}_{i,\ell}} \prod_{h:1\le h\le d, h\neq s} \frac{V_{\mathsf{An}(i,h)}(\boldsymbol{p})}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,h))} V_k(\boldsymbol{p})} \times \frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} \times \left(1 - \frac{V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})}\right) \\
&\quad - \prod_{h:1\le h\le d, h\neq \bar{s}_{i,\ell}+1} \frac{V_{\mathsf{An}(i,h)}(\boldsymbol{p})}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,h))} V_k(\boldsymbol{p})} \times \frac{\frac{\partial V_{\mathsf{An}(\ell,\bar{s}_{i,\ell}+1)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,\bar{s}_{i,\ell}+1))} V_k(\boldsymbol{p})} \times \frac{V_{\mathsf{An}(i,\bar{s}_{i,\ell}+1)}(\boldsymbol{p})}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,\bar{s}_{i,\ell}+1))} V_k(\boldsymbol{p})} \\
&= \sum_{s=1}^{\bar{s}_{i,\ell}} \prod_{h:1\le h\le d, h\neq s} \frac{V_{\mathsf{An}(i,h)}(\boldsymbol{p})}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,h))} V_k(\boldsymbol{p})} \times \frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} \\
&\quad - \sum_{s=1}^{\bar{s}_{i,\ell}} \prod_{h:1\le h\le d, h\neq s} \frac{V_{\mathsf{An}(i,h)}(\boldsymbol{p})}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,h))} V_k(\boldsymbol{p})} \times \frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} \times \frac{V_{\mathsf{An}(i,s)}(\boldsymbol{p})}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})} \\
&\quad - \prod_{h:1\le h\le d, h\neq \bar{s}_{i,\ell}+1} \frac{V_{\mathsf{An}(i,h)}(\boldsymbol{p})}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,h))} V_k(\boldsymbol{p})} \times \frac{\frac{\partial V_{\mathsf{An}(\ell,\bar{s}_{i,\ell}+1)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,\bar{s}_{i,\ell}+1))} V_k(\boldsymbol{p})} \times \frac{V_{\mathsf{An}(i,\bar{s}_{i,\ell}+1)}(\boldsymbol{p})}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,\bar{s}_{i,\ell}+1))} V_k(\boldsymbol{p})} \\
&= \sum_{s=1}^{\bar{s}_{i,\ell}} \frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}\theta_i(\boldsymbol{p}) - \sum_{s=1}^{\bar{s}_{i,\ell}} \frac{\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,s))} V_k(\boldsymbol{p})}\theta_i(\boldsymbol{p}) - \frac{\frac{\partial V_{\mathsf{An}(\ell,\bar{s}_{i,\ell}+1)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(i,\bar{s}_{i,\ell}+1))} V_k(\boldsymbol{p})}\theta_i(\boldsymbol{p}) \\
&= \theta_i^{\mathsf{root}}(\boldsymbol{p}) \left\{ \sum_{s=1}^{\bar{s}_{i,\ell}} \frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell} \left(\frac{1}{V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})} - \frac{1}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(\ell,s))} V_k(\boldsymbol{p})} \right) - \frac{\frac{\partial V_{\mathsf{An}(\ell,\bar{s}_{i,\ell}+1)}(\boldsymbol{p})}{\partial p_\ell}}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(\ell,\bar{s}_{i,\ell}+1))} V_k(\boldsymbol{p})} \right\} .
\end{aligned}
$$

□

The following lemma gives an explicit formula for the partial derivative $\frac{\partial V_{\mathsf{An}(\ell,s)}(\boldsymbol{p})}{\partial p_\ell}$.

Lemma E.7 (Derivative of Preference Weight)**.** *For any $1 \le s \le d$ and product ℓ,*

$$
\begin{aligned}
&\frac{\partial V_{\mathsf{An}(\ell,s)}}{\partial p_\ell} \\
&= \; -\beta_\ell V_\ell \cdot \prod_{h=s}^{d-1} \left(\eta_{\mathsf{An}(\ell,h)} V_{\mathsf{An}(\ell,h)}^{1-\frac{1}{\eta_{\mathsf{An}(\ell,h)}}} \right) \; = \; -\beta_\ell V_\ell \cdot \prod_{h=s}^{d-1} \left(\eta_{\mathsf{An}(\ell,h)} \frac{V_{\mathsf{An}(\ell,h)}}{\sum_{k\in\,\mathsf{Children}(\mathsf{An}(\ell,h))} V_k} \right) \\
&= \; -\beta_\ell \cdot \left\{ \prod_{h=s+1}^{d} \left(\eta_{\mathsf{An}(\ell,h)} \frac{V_{\mathsf{An}(\ell,h)}}{\sum_{k\in \mathsf{Sibling}(\mathsf{An}(\ell,h))} V_k} \right) \right\} \times \eta_{\mathsf{An}(\ell,s)} V_{\mathsf{An}(\ell,s)} \\
&= \; -\beta_\ell \cdot \left\{ \prod_{h=s+1}^{d} \eta_{\mathsf{An}(\ell,h)} \right\} \times \theta_\ell^{\mathsf{An}(\ell,s)}(\boldsymbol{p}) \times \eta_{\mathsf{An}(\ell,s)} \times V_{\mathsf{An}(\ell,s)} \; = \; -\beta_\ell \cdot \left\{ \prod_{h=s}^{d} \eta_{\mathsf{An}(\ell,h)} \right\} \times \theta_\ell^{\mathsf{An}(\ell,s)}(\boldsymbol{p}) \times V_{\mathsf{An}(\ell,s)} \; .
\end{aligned}
$$

Proof. We prove the above proposition by induction. When $n = d$,

$$LHS = \frac{\partial V_{\mathsf{An}(\ell,d)}}{\partial p_\ell} = \frac{\partial V_\ell}{\partial p_\ell} = \frac{\partial \exp(\alpha_\ell - \beta_\ell p_\ell)}{\partial p_\ell} = -\beta_\ell V_\ell = RHS \ ,$$

as $\prod_{j=d}^{d-1} \left(\eta_{\mathsf{An}(\ell,j)} V_{\mathsf{An}(\ell,j)}\right)^{1-\frac{1}{\eta_{\mathsf{An}(\ell,j)}}} = 1$. Thus, P_d is true.

Now suppose the proposition is true for $n = k+1 \in [2, d]$, i.e.,

$$\frac{\partial V_{\mathsf{An}(\ell,k+1)}}{\partial p_\ell} = -\beta_\ell V_\ell \cdot \prod_{j=k+1}^{d-1} \left(\eta_{\mathsf{An}(\ell,j)} V_{\mathsf{An}(\ell,j)}^{1-\frac{1}{\eta_{\mathsf{An}(\ell,j)}}} \right) \ .$$

Consider P_k. By definition, $V_{\mathsf{An}(\ell,k)} = \left(\sum_{s\in \mathsf{Children}(\mathsf{An}(\ell,k))} V_s\right)^{\eta_{\mathsf{An}(\ell,k)}}$. Thus,

$$\begin{aligned}
\frac{\partial V_{\mathsf{An}(\ell,k)}}{\partial p_\ell} &= \eta_{\mathsf{An}(\ell,k)} \cdot \left(\sum_{s\in \mathsf{Children}(\mathsf{An}(\ell,k))} V_s \right)^{\eta_{\mathsf{An}(\ell,k)}-1} \cdot \left(\sum_{s\in \mathsf{Children}(\mathsf{An}(\ell,k))} \frac{\partial V_s}{\partial p_\ell} \right) \\
&= \eta_{\mathsf{An}(\ell,k)} \cdot \left(\sum_{s\in \mathsf{Children}(\mathsf{An}(\ell,k))} V_s \right)^{\eta_{\mathsf{An}(\ell,k)}-1} \cdot \frac{\partial V_{\mathsf{An}(\ell,k+1)}}{\partial p_\ell} \\
&= \eta_{\mathsf{An}(\ell,k)} \cdot V_{\mathsf{An}(\ell,k)}^{\frac{\eta_{\mathsf{An}(\ell,k)}-1}{\eta_{\mathsf{An}(\ell,k)}}} \cdot \left(-\beta_\ell V_\ell \cdot \prod_{j=k+1}^{d-1} \left(\eta_{\mathsf{An}(\ell,j)} V_{\mathsf{An}(\ell,j)}^{1-\frac{1}{\eta_{\mathsf{An}(\ell,j)}}} \right) \right) \\
&= \eta_{\mathsf{An}(\ell,k)} \cdot V_{\mathsf{An}(\ell,k)}^{1-\frac{1}{\eta_{\mathsf{An}(\ell,k)}}} \cdot \left(-\beta_\ell V_\ell \cdot \prod_{j=k+1}^{d-1} \left(\eta_{\mathsf{An}(\ell,j)} V_{\mathsf{An}(\ell,j)}^{1-\frac{1}{\eta_{\mathsf{An}(\ell,j)}}} \right) \right) \\
&= -\beta_\ell V_\ell \cdot \prod_{j=k}^{d-1} \left(\eta_{\mathsf{An}(\ell,j)} V_{\mathsf{An}(\ell,j)}^{1-\frac{1}{\eta_{\mathsf{An}(\ell,j)}}} \right),
\end{aligned}$$

where the second equality follows because $\frac{\partial V_r}{\partial p_\ell} = 0$ if r is not an ancestor node of leaf node ℓ; the third equality follows from $V_{\mathsf{An}(\ell,k)} = \left(\sum_{s\in \mathsf{Children}(\mathsf{An}(\ell,k))} V_s\right)^{\eta_{\mathsf{An}(\ell,k)}}$ and P_{k+1}. Thus, P_k is true. By induction, P_n is true for all $n \in [1, d]$. □

Proof of Theorem 7.4:

Proof. The above lemma implies the following expression for the partial derivative:

Case 1: $i \neq \ell$. In this case, we have that $\bar{s}_{i,\ell} \leq d-1$, and

$$\begin{aligned}
&\frac{\partial \theta_i^{\mathsf{root}}(\boldsymbol{p})}{\partial p_\ell} \\
&= -\beta_\ell \theta_i^{\mathsf{root}}(\boldsymbol{p}) \left\{ -\left(\prod_{h=\bar{s}_{i,\ell}+1}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \theta_\ell^{\mathsf{An}(\ell,\bar{s}_{i,\ell})}(\boldsymbol{p}) + \sum_{s=1}^{\bar{s}_{i,\ell}} \left(\prod_{h=s}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \left(\theta_\ell^{\mathsf{An}(\ell,s)}(\boldsymbol{p}) - \theta_\ell^{\mathsf{An}(\ell,s-1)}(\boldsymbol{p}) \right) \right\} \\
&= \beta_\ell \theta_i^{\mathsf{root}}(\boldsymbol{p}) \left\{ \left(\prod_{h=\bar{s}_{i,\ell}+1}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \theta_\ell^{\mathsf{An}(\ell,\bar{s}_{i,\ell})}(\boldsymbol{p}) + \sum_{s=1}^{\bar{s}_{i,\ell}} \left(\prod_{h=s}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \left(\theta_\ell^{\mathsf{An}(\ell,s-1)}(\boldsymbol{p}) - \theta_\ell^{\mathsf{An}(\ell,s)}(\boldsymbol{p}) \right) \right\} \\
&= \beta_\ell \theta_i^{\mathsf{root}}(\boldsymbol{p}) \left\{ \sum_{s=1}^{\bar{s}_{i,\ell}} \theta_\ell^{\mathsf{An}(\ell,s)}(\boldsymbol{p}) \left(\prod_{h=s+1}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \left(1 - \eta_{\mathsf{An}(\ell,s)} \right) + \left(\prod_{h=1}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \theta_\ell^{\mathsf{root}}(\boldsymbol{p}) \right\} \\
&= \beta_\ell \theta_i^{\mathsf{root}}(\boldsymbol{p}) \sum_{s=0}^{\bar{s}_{i,\ell}} \theta_\ell^{\mathsf{An}(\ell,s)}(\boldsymbol{p}) \left(\prod_{h=s+1}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \left(1 - \eta_{\mathsf{An}(\ell,s)} \right) \\
&= \beta_\ell \theta_\ell^{\mathsf{root}}(\boldsymbol{p}) \theta_i^{\mathsf{root}}(\boldsymbol{p}) \sum_{s=0}^{\bar{s}_{i,\ell}} \frac{\prod_{h=s+1}^{d} \eta_{\mathsf{An}(\ell,h)}}{\theta_{\mathsf{An}(\ell,s)}^{\mathsf{root}}(\boldsymbol{p})} \left(1 - \eta_{\mathsf{An}(\ell,s)} \right) \;=\; \beta_\ell \theta_\ell^{\mathsf{root}}(\boldsymbol{p}) \theta_i^{\mathsf{root}}(\boldsymbol{p}) \mathtt{v}_{i,\ell}(\boldsymbol{p}) \ .
\end{aligned}$$

Case 2: $i = \ell$. In this case, we have that $\bar{s}_{i,\ell} = d$, and

$$\begin{aligned}
&\frac{\partial \theta_i^{\mathsf{root}}(\boldsymbol{p})}{\partial p_\ell} \\
&= -\beta_\ell \theta_i^{\mathsf{root}}(\boldsymbol{p}) \sum_{s=1}^{d} \left(\prod_{h=s}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \left(\theta_\ell^{\mathsf{An}(\ell,s)}(\boldsymbol{p}) - \theta_\ell^{\mathsf{An}(\ell,s-1)}(\boldsymbol{p}) \right) \\
&= \beta_\ell \theta_i^{\mathsf{root}}(\boldsymbol{p}) \sum_{s=1}^{d} \left(\prod_{h=s}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \left(\theta_\ell^{\mathsf{An}(\ell,s-1)}(\boldsymbol{p}) - \theta_\ell^{\mathsf{An}(\ell,s)}(\boldsymbol{p}) \right) \\
&= \beta_\ell \theta_i^{\mathsf{root}}(\boldsymbol{p}) \left\{ -\eta_{\mathsf{An}(\ell,d)} \theta_\ell^{\mathsf{An}(\ell,d)} + \sum_{s=1}^{d-1} \theta_\ell^{\mathsf{An}(\ell,s)}(\boldsymbol{p}) \left(\prod_{h=s+1}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \left(1 - \eta_{\mathsf{An}(\ell,s)} \right) + \left(\prod_{h=1}^{d} \eta_{\mathsf{An}(\ell,h)} \right) \theta_\ell^{\mathsf{root}}(\boldsymbol{p}) \right\} \\
&= \beta_\ell \theta_\ell^{\mathsf{root}}(\boldsymbol{p}) \theta_i^{\mathsf{root}}(\boldsymbol{p}) \left\{ -\frac{1}{\theta_\ell^{\mathsf{root}}(\boldsymbol{p})} \;+\; \sum_{s=0}^{d-1} \frac{\prod_{h=s+1}^{d} \eta_{\mathsf{An}(\ell,h)}}{\theta_{\mathsf{An}(\ell,s)}^{\mathsf{root}}(\boldsymbol{p})} \left(1 - \eta_{\mathsf{An}(\ell,s)} \right) \right\} \;=\; \beta_\ell \theta_\ell^{\mathsf{root}}(\boldsymbol{p}) \theta_i^{\mathsf{root}}(\boldsymbol{p}) \mathtt{v}_{i,\ell}(\boldsymbol{p}) \ .
\end{aligned}$$

So, we have that

$$\begin{aligned}
\frac{\partial \Pi}{\partial p_\ell}(\boldsymbol{p}) &= -\theta_\ell^{\mathsf{root}}(\boldsymbol{p}) \beta_\ell \left(p_\ell - \frac{1}{\beta_\ell} - \mathtt{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p}) \right) - \sum_{i=1}^{n} C_i'(\theta_i^{\mathsf{root}}(\boldsymbol{p})) \frac{\partial \theta_i(\boldsymbol{p})}{\partial p_\ell} \\
&= -\theta_\ell^{\mathsf{root}}(\boldsymbol{p}) \beta_\ell \left(p_\ell - \frac{1}{\beta_\ell} - \mathtt{u}_{\mathsf{Parent}(\ell)}(\boldsymbol{p}) + \sum_{i=1}^{n} C_i'(\theta_i^{\mathsf{root}}(\boldsymbol{p})) \theta_i^{\mathsf{root}}(\boldsymbol{p}) \mathtt{v}_{i,\ell}(\boldsymbol{p}) \right) \ ,
\end{aligned}$$

which is the desired result. $\square$

F. Additional Numerical Results for Price Optimization

In this section, we provide additional numerical results for price optimization problems. In our first set of results, we focus on the test problems with three levels in the tree and test the performance of PUPD and GA with different initial starting points. Following these numerical results, we focus on test problems with two levels in the tree. When there are two levels in the tree, Gallego and Wang (2013) give a set of conditions that need to be satisfied by the optimal price vector. Therefore, for a tree with two levels, we can numerically look for a price vector that satisfies the optimality conditions in Gallego and Wang (2013).

Problem Instances with Three Levels in the Tree: Our results in Section 8 in the paper show that PUPD performs significantly better than GA, but gradient ascent methods can provide more satisfactory performance when the initial vector of prices is close to a stationary point of the expected revenue function. To check the performance of GA with such informed initial prices, for each problem instance, we compute a stationary point of the expected revenue function by using GA. Then, we perturb these prices by 2.5% to obtain prices that are close to a stationary point of the expected revenue function and we use these prices as a vector of informed initial prices. In Table 6, we show the performance of GA and PUPD when we start these algorithms by using the informed initial prices. The format of this table is identical to that of Table 4. The results in Table 6 indicate that PUPD continues to perform significantly better than GA, both in terms of the number of iterations and the running time to reach the stopping criterion. The ratios between the iterations and the running times never fall below one, indicating that PUPD provides improvements over GA for all of the problem instances.

We note that other alternative implementations of the gradient ascent method are not likely to be more effective than PUPD. In particular, noting Lemma 7.1, the computational work per iteration for PUPD is no larger than the computational work required to compute the gradient of the expected revenue function at a particular price vector. Thus, given that a gradient ascent method needs to compute the gradient of the expected revenue function in each iteration, PUPD and GA will have comparable computational work per iteration even if we can compute the optimal step size for GA instantaneously. Since PUPD reaches the stopping criterion in substantially fewer iterations than GA, it is likely to maintain its performance advantage over GA for any alternative implementations of the gradient ascent method.

Problem Instances with Two Levels in the Tree: Our experimental setup for the test problems with two levels is similar to the one for the test problems with three levels. Using m_h

Prb. Class			Avg. No. Itns		Ratio Between No. Itns.			Ratio Between Run. Times		
m_0	m_1	m_2	PUPD	GA	Avg.	Min	Max	Avg.	Min	Max
2	2	2	83	1,797	15	2	195	12	2	153
2	2	4	72	3,059	33	5	265	28	3	295
2	2	6	57	3,882	50	9	285	41	6	317
2	4	2	82	4,702	52	4	346	40	4	274
2	4	4	66	4,628	63	11	265	53	9	234
2	4	6	58	5,878	93	19	383	78	13	423
2	6	2	83	5,445	60	5	248	49	4	257
2	6	4	72	6,925	95	18	361	80	12	414
2	6	6	62	8,429	138	29	517	112	25	383
4	2	2	107	5,722	46	3	302	39	3	267
4	2	4	81	6,126	69	11	276	59	8	269
4	2	6	70	8,132	110	19	362	93	17	281
4	4	2	89	8,580	94	12	334	81	11	300
4	4	4	67	9,855	145	36	362	120	22	298
4	4	6	55	10,407	189	41	393	145	26	427
4	6	2	80	10,878	141	18	407	120	15	345
4	6	4	61	12,940	222	68	490	192	47	543
4	6	6	51	14,542	298	90	607	255	82	642
6	2	2	106	7,651	72	7	320	61	5	220
6	2	4	79	9,371	118	16	515	99	12	364
6	2	6	64	9,314	146	27	493	120	21	364
6	4	2	82	12,590	156	24	345	135	19	389
6	4	4	61	13,263	226	58	477	190	49	492
6	4	6	51	14,717	298	128	699	262	106	634
6	6	2	70	14,098	212	72	459	186	59	444
6	6	4	53	16,136	321	126	709	278	98	628
6	6	6	45	17,124	411	148	762	362	113	837

Table 6: Performance comparison between PUPD and GA with informed initial prices.

to denote the number of children of each node at depth h, we vary $(m_0, m_1) \in \{2,4,8\} \times \{2,4,8\}$, yielding nine problem classes. In each problem class, we randomly generate 200 individual problem instances by using the same sampling strategy that we use for the test problems with three levels. For the two-level nested logit model, Gallego and Wang (2013) give a set of conditions that need to be satisfied by the optimal price vector. These conditions come in the form of an implicit set of equations. We use golden section search to numerically find a solution to this set of conditions. We use OC to refer to the solution strategy that finds a solution to the optimality conditions of Gallego and Wang (2013). We stop the golden section search for OC when we reach a price vector $\boldsymbol{p}^s$ at which the gradient of the expected revenue function satisfies $\|\nabla R_{\mathsf{root}}(\boldsymbol{p}^s)\|_2 \leq 10^{-6}$. We use the same stopping criterion for PUPD.

We summarize our results in Table 7. The first two columns in this table show the parameters (m_0, m_1) of each problem class. The third, fourth and fifth columns give various statistics regarding the ratio between the running times of OC and PUPD. In particular, these three columns give the average, minimum and maximum of the ratios of the running times of OC and PUPD, where the average, minimum and maximum are computed over the 200 problem instances in a particular problem class. The results in Table 7 indicate that average running times for PUPD are

Prb. Class		Ratio Between Run. Times		
m_0	m_1	Avg.	Min	Max
2	2	9.6	0.8	42.8
2	4	9.2	1.0	55
2	8	11.4	1.2	72.8
4	2	10.7	1.2	49.6
4	4	14.1	1.1	65.3
4	8	19.7	2.5	93.5
8	2	15.1	2.9	68.6
8	4	23.7	4.2	196.1
8	8	34.8	5.1	163.5

Table 7: Performance comparison between PUPD and OC.

substantially smaller than those of OC. There are problem instances where the ratio of the running times is below one, indicating that OC runs faster than PUPD for these problem instances, but such problem instances correspond to the cases where the number of products is small. In larger problem instances, PUPD can be more than 160 times faster than OC. Although we do not report this statistic in Table 7, considering the total of $1,800$ problem instances we work with, in more than 99.4% of them, the running time of PUPD is faster than the running time of OC. The ratios of the running times have an increasing trend as the problem size measured by the number of products $m_0\, m_1$ increases, which indicates that the running time advantage of PUPD becomes more pronounced as the number of products increases. PUPD and OC can converge to different stationary points of the expected revenue function and this expectation was confirmed in our numerical experiments. Nevertheless, over all of our problem instances, the expected revenues at the stationary points obtained by PUPD and OC are essentially equal. Overall, our experiments demonstrate that PUPD performs remarkably well, when compared to both GA and OC.

G. Proof of Theorem 9.1

It suffices to consider the case when we offer the full assortment of products $S = \{1, 2, \ldots, n\}$. Without loss of generality, we can ignore the no-purchase option and treat one of the products as the no-purchase option. For each non-leaf node j, define the scalar τ_j as $\tau_j = \tau_{\mathsf{Parent}(j)} \times \eta_j$ with the convention that $\tau_{\mathsf{root}} = 1$, and $\tau_j = \tau_{\mathsf{Parent}(j)}$ for each leaf node j. Let a function $G(\cdot) : \mathbb{R}^n_+ \to \mathbb{R}_+$ be defined by: for each $\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n_+$,

$$G(\boldsymbol{y}) = \sum_{j \,\in\, \mathsf{Children}(\mathsf{root})} Y_j(\boldsymbol{y}) \ ,$$

where for each non-leaf node j, $Y_j(\boldsymbol{y}) = \left(\sum_{k \,\in\, \mathsf{Children}(j)} Y_k(\boldsymbol{y})^{1/\tau_j} \right)^{\tau_j}$, and if j is a leaf node, then $Y_j(\boldsymbol{y}) = y_j$. Note that for each non-leaf node j, $Y_j(\boldsymbol{y})$ is a function of y_ℓ for all leaf nodes ℓ that are descendants of j. We also have that $G(\boldsymbol{y}) = Y_{\mathsf{root}}(\boldsymbol{y})$ because $\tau_{\mathsf{root}} = 1$. We will show that $G(\cdot)$ satisfies the following properties:

(a) $G(\boldsymbol{y}) \geq 0$ for all $\boldsymbol{y} \in \mathbb{R}^n_+$.

(b) $G(\cdot)$ is homogenous of degree 1; that is, $G(\rho\, \boldsymbol{y}) = \rho\, G(\boldsymbol{y})$ for all $\rho \geq 0$.

(c) The function $G(\cdot)$ is positive whenever its arguments are positive.

(d) For each distinct $\{i_1, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$, $\frac{\partial^k G}{\partial y_{i_1} \partial y_{i_2} \cdots \partial y_{i_k}}(\boldsymbol{y})$ is non-negative if k is odd, and non-positive if k is even.

Then, it follows from Theorem 1 in McFadden (1978) that for each product ℓ, if the selection probability of this product is given by

$$\frac{y_\ell}{G(\boldsymbol{y})} \cdot \frac{\partial G}{\partial y_\ell}(\boldsymbol{y}) \;\Bigg|_{\boldsymbol{y} \,=\, (e^{\mu_1}, \ldots, e^{\mu_n})} ,$$

then we obtain a choice model that is consistent with utility maximization, where the random utility of each product ℓ is given by $\mathsf{Utility}_\ell = \mu_\ell + \varepsilon_\ell$, and $(\varepsilon_1, \ldots, \varepsilon_n)$ has a multivariate extreme value distribution. To complete the proof, in the rest of this section, we will show that the selection probability of product ℓ is indeed equal to the above expression for appropriately chosen $(\mu_1, \ldots, \mu_n)$; in particular, we will show that

$$\prod_{h=1}^{d} \frac{V_{\mathsf{An}(\ell,h)}\left(S_{\mathsf{An}(\ell,h)}\right)}{\sum_{k \in \mathsf{Sibling}(\mathsf{An}(\ell,h))} V_k(S_k)} \quad = \quad \frac{y_\ell}{G(\boldsymbol{y})} \cdot \frac{\partial G}{\partial y_\ell}(\boldsymbol{y}) \;\Bigg|_{\boldsymbol{y} \,=\, (v_1^{\tau_1}, \ldots, v_n^{\tau_n})} .$$

Recall that $\tau_j = \tau_{\mathsf{Parent}(j)} \times \eta_j$ for each non-leaf node j with the convention that $\tau_{\mathsf{root}} = 1$ and $\tau_j = \tau_{\mathsf{Parent}(j)}$ for each leaf node j. We need two lemmas before completing the proof of Theorem 9.1. The following lemma gives an expression for the derivative of $Y_j(\boldsymbol{y})$ with respect to y_ℓ.

Lemma G.1. *Assume that j is a non-leaf node and ℓ is a leaf node that is a descendant of node j. Then, we have*

$$\frac{\partial Y_j}{\partial y_\ell}(\boldsymbol{y}) = Y_j(\boldsymbol{y})^{(\tau_j - 1)/\tau_j} \left[\prod_{v \in \mathsf{path}(j,\ell)} Y_v(\boldsymbol{y})^{(\eta_v - 1)/\tau_v} \right] y_\ell^{(1-\tau_\ell)/\tau_\ell} ,$$

where $\mathsf{path}(j,\ell)$ is the set of nodes on the path from node j to ℓ, excluding the end nodes j and ℓ.

Proof: We show the result by using induction on the depth of node j. If node j is at depth $d-1$, then we have $Y_j(\boldsymbol{y}) = \left(\sum_{k\in \mathsf{Children}(j)} y_k^{1/\tau_j}\right)^{\tau_j}$, and differentiating with respect to y_ℓ, we get

$$\frac{\partial Y_j}{\partial y_\ell}(\boldsymbol{y}) = \tau_j \left(\sum_{k\in \mathsf{Children}(j)} y_k^{1/\tau_j}\right)^{\tau_j - 1} \frac{1}{\tau_j}\, y_\ell^{(1-\tau_j)/\tau_j} = Y_j(\boldsymbol{y})^{(\tau_j-1)/\tau_j} y_\ell^{(1-\tau_\ell)/\tau_\ell} \ ,$$

where the second equality holds because $\tau_\ell = \tau_{\mathsf{Parent}(\ell)} = \tau_j$ for the leaf node ℓ. This completes the base case. Now, assume that the result holds at any node at depth $h+1$ and focus on node j that is at depth h. Consider the path in the tree from node j to leaf node ℓ and denote the nodes on this path by $j = j_h, j_{h+1}, \ldots, j_{d-1}, j_d = \ell$, where j_g corresponds to the node at depth g. For each $k \in \mathsf{Children}(j)$, observe that $Y_k(\boldsymbol{y})$ depends on y_ℓ only if node ℓ is included in the subtree rooted at node k. Thus, among $(Y_k(\boldsymbol{y}) : k \in \mathsf{Children}(j))$, only $Y_{j_{h+1}}(\boldsymbol{y})$ depends on y_ℓ. In this case, recalling $Y_j(\boldsymbol{y}) = \left(\sum_{k\in \mathsf{Children}(j)} Y_k(\boldsymbol{y})^{1/\tau_j}\right)^{\tau_j}$, we get

$$\begin{aligned}
\frac{\partial Y_j}{\partial y_\ell}(\boldsymbol{y}) &= \tau_j \left(\sum_{k\in \mathsf{Children}(j)} Y_k(\boldsymbol{y})^{1/\tau_j}\right)^{\tau_j-1} \frac{1}{\tau_j}\, Y_{j_{h+1}}(\boldsymbol{y})^{(1-\tau_j)/\tau_j} \frac{\partial Y_{j_{h+1}}}{\partial y_\ell}(\boldsymbol{y}) \\
&= Y_j(\boldsymbol{y})^{(\tau_j-1)/\tau_j}\, Y_{j_{h+1}}(\boldsymbol{y})^{(1-\tau_j)/\tau_j} \\
&\qquad \times Y_{j_{h+1}}(\boldsymbol{y})^{(\tau_{j_{h+1}}-1)/\tau_{j_{h+1}}} \left[\prod_{v\in \mathsf{path}(j_{h+1},\ell)} Y_v(\boldsymbol{y})^{(\eta_v-1)/\tau_v}\right] y_\ell^{(1-\tau_\ell)/\tau_\ell} \\
&= Y_j(\boldsymbol{y})^{(\tau_j-1)/\tau_j}\, Y_{j_{h+1}}(\boldsymbol{y})^{(\eta_{j_{h+1}}-1)/\tau_{j_{h+1}}} \left[\prod_{v\in \mathsf{path}(j_{h+1},\ell)} Y_v(\boldsymbol{y})^{(\eta_v-1)/\tau_v}\right] y_\ell^{(1-\tau_\ell)/\tau_\ell} \\
&= Y_j(\boldsymbol{y})^{(\tau_j-1)/\tau_j} \left[\prod_{v\in \mathsf{path}(j,\ell)} Y_v(\boldsymbol{y})^{(\eta_v-1)/\tau_v}\right] y_\ell^{(1-\tau_\ell)/\tau_\ell} \ ,
\end{aligned}$$

where the second equality is by the induction assumption and the third equality follows by noting that $\frac{1-\tau_j}{\tau_j} + \frac{\tau_{j_{h+1}}-1}{\tau_{j_{h+1}}} = \frac{1}{\tau_{j_{h+1}}}\left(\frac{\tau_{j_{h+1}}}{\tau_j} - 1\right) = \frac{1}{\tau_{j_{h+1}}}\left(\frac{\tau_{j_{h+1}}}{\tau_{j_h}} - 1\right) = \frac{1}{\tau_{j_{h+1}}}\left(\eta_{j_{h+1}} - 1\right)$. □

The following lemma shows a relationship between the preference weights $(V_j(N_j) : j \in \mathsf{V})$ and $(Y_j(\boldsymbol{y}) : j \in \mathsf{V})$, when $(Y_j(\boldsymbol{y}) : j \in \mathsf{V})$ are evaluated at $\boldsymbol{y} = (v_1^{\tau_1}, \ldots, v_n^{\tau_n})$.

Lemma G.2. *Letting $\hat{\boldsymbol{y}} = (v_1^{\tau_1}, \ldots, v_n^{\tau_n})$, for each non-leaf node j, we have*

$$V_j(N_j) = Y_j(\hat{\boldsymbol{y}})^{\eta_j/\tau_j} \ .$$

Proof: We show the result by using induction on the depth of node j. If node j is at depth $d-1$, then we get

$$V_j(N_j) = \left(\sum_{k \in \mathsf{Children}(j)} v_k \right)^{\eta_j} = \left(\sum_{k \in \mathsf{Children}(j)} v_k^{\tau_k/\tau_j} \right)^{\eta_j} = \left(\sum_{k \in \mathsf{Children}(j)} \hat{y}_k^{1/\tau_j} \right)^{\eta_j} = Y_j(\hat{\boldsymbol{y}})^{\eta_j/\tau_j} \ ,$$

where the second equality follows from the fact that $\tau_k = \tau_{\mathsf{Parent}(k)} = \tau_j$ when k is a leaf node, and the last equality follows from the definition of $Y_j(\hat{\boldsymbol{y}})$. Suppose the result holds for nodes at depth $h+1$ and consider node j at depth h. We have

$$\begin{aligned} V_j(N_j) &= \left(\sum_{k \in \mathsf{Children}(j)} V_k(N_k) \right)^{\eta_j} = \left(\sum_{k \in \mathsf{Children}(j)} Y_k(\hat{\boldsymbol{y}})^{\eta_k/\tau_k} \right)^{\eta_j} \\ &= \left(\sum_{k \in \mathsf{Children}(j)} Y_k(\hat{\boldsymbol{y}})^{1/\tau_j} \right)^{\eta_j} = Y_j(\hat{\boldsymbol{y}})^{\eta_j/\tau_j} \ , \end{aligned}$$

where the second equality is by the induction assumption and the third equality follows from the fact that $\eta_k = \tau_k / \tau_{\mathsf{Parent}(k)} = \tau_k/\tau_j$. $\square$

Finally, here is the rest of the proof of Theorem 9.1.

Proof of Theorem 9.1: Properties (a), (b) and (c) in the proof of Theorem follow immediately. We will first focus on property (d). Let $\bar{\mathsf{V}} = \mathsf{V} \setminus \{1, 2 \ldots, n, \mathsf{root}\}$ denote the set of non-leaf nodes in the tree, after excluding the root node. For part (d), it suffices to show that for $k = 1, \ldots, n$,

$$\frac{\partial^k Y_{\mathsf{root}}}{\partial y_k \partial y_{k-1} \cdots \partial y_2 \partial y_1}(\boldsymbol{y}) = \left[\prod_{j=1}^{k} y_j^{(1-\tau_j)/\tau_j} \right] \times \left[\sum_{s \in I_k} c_s(k) \prod_{v \in \bar{\mathsf{V}}} Y_v^{d_s(k,v)}(\boldsymbol{y}) \right] \ ,$$

where I_k is some index set and the exponent $d_s(k,v)$ satisfies $d_s(k,v) \leq 0$ for all $s \in I_k$ and $v \in \bar{\mathsf{V}}$. Most importantly, the coefficients $c_s(k)$ have the following property: if k is odd, then $c_s(k) \geq 0$ for all $s \in I_k$, and if k is even, then $c_s(k) \leq 0$ for all $s \in I_k$. Since $Y_v(\boldsymbol{y}) \geq 0$ for all $\boldsymbol{y}$, the properties of the coefficients give the desired result.

We will prove the above equality by induction on k. Consider the base case of $k = 1$. Since $\tau_{\mathsf{root}} = 1$, it follows from Lemma G.1 that

$$\frac{\partial Y_{\mathsf{root}}}{\partial y_1}(\boldsymbol{y}) = \left[\prod_{v \in \mathsf{path}(\mathsf{root},1)} Y_v(\boldsymbol{y})^{(\eta_v - 1)/\tau_v} \right] y_1^{(1-\tau_1)/\tau_1} = y_1^{(1-\tau_1)/\tau_1} \left[\prod_{v \in \bar{\mathsf{V}}} Y_v(\boldsymbol{y})^{d_1(1,v)} \right] \ ,$$

where we let

$$d_1(1,v) = \begin{cases} \dfrac{\eta_v - 1}{\tau_v} & \text{if } v \in \mathsf{path}(\mathsf{root}, 1) \\ 0 & \text{otherwise .} \end{cases}$$

Note that the exponent $d_1(1,v)$ is always non-positive because $\eta_v \leq 1$. Therefore, the expression for $\frac{\partial Y_{\mathsf{root}}}{\partial y_1}(\boldsymbol{y})$ has the desired form with $I_1 = \{1\}$ and $c_1(1) = 1$. This establishes the base case.

Suppose the claim is true for some k. It follows from the inductive hypothesis and the product rule of differentiation that

$$\begin{aligned}
\frac{\partial^{k+1} Y_{\mathsf{root}}}{\partial y_{k+1} \partial y_k \cdots \partial y_2 \partial y_1}(\boldsymbol{y}) &= \left[\prod_{j=1}^{k} y_j^{\frac{1-\tau_j}{\tau_j}}\right] \times \left[\sum_{s \in I_k} c_s(k) \frac{\partial}{\partial y_{k+1}} \prod_{v \in \bar{\mathsf{V}}} Y_v(\boldsymbol{y})^{d_s(k,v)}\right] \\
&= \left[\prod_{j=1}^{k} y_j^{\frac{1-\tau_j}{\tau_j}}\right] \times \left[\sum_{s \in I_k} c_s(k) \left\{\sum_{u \in \bar{\mathsf{V}}} \left(\prod_{v \in \bar{\mathsf{V}}: v \neq u} Y_v(\boldsymbol{y})^{d_s(k,v)}\right) \frac{\partial Y_u^{d_s(k,u)}}{\partial y_{k+1}}(\boldsymbol{y})\right\}\right] \\
&= \left[\prod_{j=1}^{k} y_j^{\frac{1-\tau_j}{\tau_j}}\right] \times \left[\sum_{s \in I_k} c_s(k) \left\{\sum_{u \in \bar{\mathsf{V}}} \left(\prod_{v \in \bar{\mathsf{V}}: v \neq u} Y_v(\boldsymbol{y})^{d_s(k,v)}\right) \times d_s(k,u)\; Y_u(\boldsymbol{y})^{d_s(k,u)-1} \frac{\partial Y_u}{\partial y_{k+1}}(\boldsymbol{y})\right\}\right] \\
&= \left[\prod_{j=1}^{k} y_j^{\frac{1-\tau_j}{\tau_j}}\right] \times \left[\sum_{s \in I_k} c_s(k) \left\{\sum_{u \in \bar{\mathsf{V}}} \left(\prod_{v \in \bar{\mathsf{V}}: v \neq u} Y_v(\boldsymbol{y})^{d_s(k,v)}\right) \times d_s(k,u)\, Y_u(\boldsymbol{y})^{d_s(k,u)-1} \right.\right. \\
&\qquad \left.\left. \times\, Y_u(\boldsymbol{y})^{(\tau_u - 1)/\tau_u} \left[\prod_{w \in \mathsf{path}(u,k+1)} Y_w(\boldsymbol{y})^{(\eta_w - 1)/\tau_w}\right] y_{k+1}^{(1-\tau_{k+1})/\tau_{k+1}}\right\}\right] \\
&= \left[\prod_{j=1}^{k+1} y_j^{\frac{1-\tau_j}{\tau_j}}\right] \times \left[\sum_{(s,u) \in I_k \times \bar{\mathsf{V}}} c_s(k)\, d_s(k,u) \left(\prod_{v \in \bar{\mathsf{V}}: v \neq u} Y_v(\boldsymbol{y})^{d_s(k,v)}\right)\right. \\
&\qquad \left. \times \left[\prod_{w \in \mathsf{path}(u,k+1)} Y_w(\boldsymbol{y})^{(\eta_w - 1)/\tau_w}\right] Y_u(\boldsymbol{y})^{d_s(k,u) - 1 + (\tau_u - 1)/\tau_u}\right] ,
\end{aligned}$$

where the fourth equality follows by replacing $\partial Y_u(\boldsymbol{y})/\partial y_{k+1}$ by its equivalent form given by Lemma G.1. The last summation above is indexed by $I_k \times \bar{\mathsf{V}}$. We let $I_{k+1} = I_k \times \bar{\mathsf{V}}$. Focusing on each term $(s,u) \in I_k \times \bar{\mathsf{V}}$ in this summation individually and rearranging the terms, we have

$$\begin{aligned}
&\left(\prod_{v \in \bar{\mathsf{V}}: v \neq u} Y_v(\boldsymbol{y})^{d_s(k,v)}\right) \left[\prod_{w \in \mathsf{path}(u,k+1)} Y_w(\boldsymbol{y})^{(\eta_w - 1)/\tau_w}\right] Y_u(\boldsymbol{y})^{d_s(k,u) - 1 + (\tau_u - 1)/\tau_u} \\
&= \left(\prod_{v \in \bar{\mathsf{V}}: v \neq u,\ v \notin \mathsf{path}(u,k+1)} Y_v(\boldsymbol{y})^{d_s(k,v)}\right) \left[\prod_{v \in \bar{\mathsf{V}}: v \neq u,\ v \in \mathsf{path}(u,k+1)} Y_v(\boldsymbol{y})^{d_s(k,v) + \frac{\eta_v - 1}{\tau_v}}\right] Y_u(\boldsymbol{y})^{d_s(k,u) - 1 + \frac{\tau_u - 1}{\tau_u}} \\
&= \prod_{v \in \bar{\mathsf{V}}} Y_v(\boldsymbol{y})^{d_{(s,u)}(k+1,v)} ,
\end{aligned}$$

where we let

$$d_{(s,u)}(k+1,v) = \begin{cases} d_s(k,v) & \text{if } v \neq u \text{ and } v \notin \mathsf{path}(k+1,u) \\ d_s(k,v) + \dfrac{\eta_v - 1}{\tau_v} & \text{if } v \neq u \text{ and } v \in \mathsf{path}(k+1,u) \\ d_s(k,v) - 1 + \dfrac{\tau_v - 1}{\tau_v} & \text{if } v = u \ . \end{cases}$$

Therefore, if we define $c_{(s,u)}(k+1)$ as $c_{(s,u)}(k+1) = c_s(k)\, d_s(k,u)$, then we have that

$$\frac{\partial^{k+1} Y_{\mathsf{root}}}{\partial y_{k+1} \partial y_k \cdots \partial y_2 \partial y_1}(\boldsymbol{y}) = \left[\prod_{j=1}^{k} y_j^{\frac{1}{\tau_j}-1}\right] \times \left[\sum_{(s,u)\in I_{k+1}} c_{(s,u)}(k+1) \prod_{v\in \bar{\mathsf{V}}} Y_v(\boldsymbol{y})^{d_{(s,u)}(k+1,v)}\right] ,$$

which has exactly the desired form. If $k+1$ is odd, then k is even and thus, $c_s(k)\, d_s(k,u) \geq 0$ because $c_s(k) \leq 0$ and $d_s(k,u) \leq 0$ by the inductive hypothesis. Similarly, if $k+1$ is even, then k is odd, and thus $c_s(k)\, d_s(k,u) \leq 0$ because $c_s(k) \geq 0$ and $d_s(k,u) \leq$ by the inductive hypothesis. Thus, the coefficients have the correct sign. Moreover, by the definition of $d_{(s,u)}(k+1,v)$, all of the exponents are non-positive because $\eta_v \leq 1$ for all $v \in \mathsf{V}$ and $d_s(k,v) \leq 0$ for all $v \in \bar{\mathsf{V}}$ by the inductive hypothesis. This completes the induction and property (d) follows.

To complete the proof of Theorem 9.1, we show that the product selection probabilities have the desired form. Let $\hat{\boldsymbol{y}} = (v_1^{\tau_1}, \ldots, v_n^{\tau_n})$. We have $G(\boldsymbol{y}) = \left(\sum_{j\in \mathsf{Children(root)}} Y_k(\boldsymbol{y})^{1/\tau_{\mathsf{root}}}\right)^{\tau_{\mathsf{root}}} = Y_{\mathsf{root}}(y)$ because $\tau_{\mathsf{root}} = 1$. Thus, we can use Lemma G.1 to compute the derivative of $G(\boldsymbol{y})$ with respect to y_ℓ. Evaluating this derivative at $\boldsymbol{y} = \hat{\boldsymbol{y}}$, we get

$$\begin{aligned} \frac{\partial G}{\partial y_\ell}(\boldsymbol{y})\bigg|_{\boldsymbol{y}=\hat{\boldsymbol{y}}} &= \frac{\partial Y_{\mathsf{root}}}{\partial y_\ell}(\boldsymbol{y})\bigg|_{\boldsymbol{y}=\hat{\boldsymbol{y}}} = \left[\prod_{j\in \mathsf{path(root},\ell)} Y_j(\hat{\boldsymbol{y}})^{(\eta_j-1)/\tau_j}\right] \hat{y}_\ell^{(1-\tau_\ell)/\tau_\ell} \\ &= \left[\prod_{j\in \mathsf{path(root},\ell)} V_j(N_j)^{(\eta_j-1)/\eta_j}\right] V_\ell(N_\ell)^{1-\tau_\ell} = \left[\prod_{j\in \mathsf{path(root},\ell)} \frac{V_j(N_j)}{\sum_{k\in \mathsf{Children}(j)} V_k(N_k)}\right] V_\ell(N_\ell)^{1-\tau_\ell} , \end{aligned}$$

where the third equality follows from Lemma G.2 and the fact that $V_\ell(N_\ell) = v_\ell = \hat{y}_\ell^{1/\tau_\ell}$ for the leaf node ℓ, and the last equality uses the fact that $V_j(N_j) = \left(\sum_{k\in \mathsf{Children}(j)} V_k(N_k)\right)^{\eta_j}$. Observe that by Lemma G.2, we have

$$G(\hat{\boldsymbol{y}}) = \sum_{k\in \mathsf{Children(root)}} Y_k(\hat{\boldsymbol{y}}) = \sum_{k\in \mathsf{Children(root)}} V_k(N_k)^{\tau_k/\eta_k} = \sum_{k\in \mathsf{Children(root)}} V_k(N_k) ,$$

where we use the fact that $\eta_k = \tau_k/\tau_{\mathsf{Parent}(k)} = \tau_k/\tau_{\mathsf{root}} = \tau_k$ for all $k \in \mathsf{Children(root)}$. Thus, using the above equality, we have that

$$
\begin{aligned}
\frac{\hat{y}_\ell}{G(\hat{\boldsymbol{y}})} \frac{\partial G}{\partial y_\ell}(\boldsymbol{y})\bigg|_{y=\hat{\boldsymbol{y}}} &= \frac{v_\ell^{\tau_\ell}}{\sum_{k\in \mathsf{Children(root)}} V_k(N_k)} \left[\prod_{h=1}^{d-1} \frac{V_{\mathsf{An}(\ell,h)}(N_{\mathsf{An}(\ell,h)})}{\sum_{k\in \mathsf{Children(An}(\ell,h))} V_k(N_k)}\right] V_\ell(N_\ell)^{1-\tau_\ell} \\
&= \frac{\prod_{h=1}^{d} V_{\mathsf{An}(\ell,h)}(N_{\mathsf{An}(\ell,h)})}{\prod_{h=0}^{d-1} \sum_{k\in \mathsf{Children(An}(\ell,h))} V_k(N_k)} = \frac{\prod_{h=1}^{d} V_{\mathsf{An}(\ell,h)}(N_{\mathsf{An}(\ell,h)})}{\prod_{h=0}^{d-1} \sum_{k\in \mathsf{Sibling(An}(\ell,h+1))} V_k(N_k)} \\
&= \prod_{h=1}^{d} \frac{V_{\mathsf{An}(\ell,h)}\left(N_{\mathsf{An}(\ell,h)}\right)}{\sum_{k\in \mathsf{Sibling(An}(\ell,h))} V_k(N_k)} \ ,
\end{aligned}
$$

where the first equality uses the fact that the nodes in $\mathsf{path}(\mathsf{root}, \ell)$ are precisely the ancestors of the node ℓ at levels $1, 2, \ldots, d-1$, the second equality follows by the fact that $v_\ell^{\tau_\ell} = V_\ell(N_\ell)$ and node ℓ is the ancestor of node ℓ at dept d, and the third equality is by noting that the children of a node are the same as the siblings of one of its children. □

H. Parameter Estimation

Flexibility of the d-level nested logit model can be enhanced by increasing the number of levels in the tree. In this section, our goal is to demonstrate that the parameters of the d-level nested logit model can be estimated relatively easily, and using multiple levels in the tree can help better predict the customer choice behavior.

The rest of this section is organized as follows. We describe the setup for the parameter estimation problem and show our main theoretical result in Section H.1. In Section H.2, we give a numerical example to demonstrate that it is possible to significantly improve the prediction accuracy of the d-level nested logit model by increasing the number of levels in the tree. In Section H.3, we describe the setup of our numerical experiments and summarize the practical performance of d-level nested logit models for an ensemble of data sets. Last but not least, proof of the main theorem is given in Section H.4.

H.1 Properties of the Log-Likelihood Function

To estimate the parameters of the d-level nested logit model, note that this model has a parameter η_j for each non-leaf node j and a parameter v_ℓ for each leaf node ℓ. We capture the parameters of the non-leaf nodes at level h as $\boldsymbol{\chi}_h = (\eta_j : \mathsf{depth}(j) = h)$, where $\mathsf{depth}(j)$ is the depth of node j. Therefore, $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1})$ represents the parameters of the non-leaf nodes. To capture the parameters of the leaf nodes, we assume that $v_\ell = e^{\beta_\ell}$ for some $\beta_\ell \in \mathbb{R}$, in which case, we represent the parameters of the leaf nodes as $\boldsymbol{\beta} = (\beta_\ell : \mathsf{depth}(\ell) = d)$. Thus, the parameters of the d-level nested logit model are given by $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$ and we write the choice probability in (8) as $\theta_\ell(S \,|\, \boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$ to explicitly show its dependence on the parameters. Our goal is to use data to estimate the parameters $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$.

Let us describe the setup for the parameter estimation problem. In the data, we have the offered assortments and the choices of T customers. For each customer $t = 1, \ldots, T$, we use S^t to denote the assortment offered to this customer and c^t to denote the option chosen by this customer. Assuming that the choices of the customers are independent of each other and the customers make their choices according to the d-level nested logit model with parameters $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$, the log-likelihood function is given by

$$\mathcal{L}\left(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta}\right) = \sum_{t=1}^{T} \log \theta_{c^t}(S^t \mid \boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta}) \; . \tag{14}$$

The above log-likelihood function is not jointly concave in $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$, but the following theorem shows that it is concave in each one of the d components $\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}$, and $\boldsymbol{\beta}$. The proof of the theorem is given in Section H.4.

Theorem H.1. *The log-likelihood function in* (14) *satisfies the following properties:*

(a) The function $\mathcal{L}(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$ *is concave in* $\boldsymbol{\beta}$.

(b) For $h = 1, 2, \ldots, d-1$, *the function* $\mathcal{L}\left(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{h-1}, \boldsymbol{\chi}_h, \boldsymbol{\chi}_{h+1}, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta}\right)$ *is concave in* $\boldsymbol{\chi}_h$.

Thus, a natural approach for maximizing the log-likelihood function is coordinate ascent, where we fix all-but-one of the d components $\boldsymbol{\chi}_1, \ldots \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta}$, and maximize the log-likelihood function over one component at a time. We can sequentially maximize over one component at a time and cycle over each component multiple times until no further improvement is possible.

For the two-level nested logit model, Train (2003) suggests estimating the parameters of the leaf nodes first, and then, the parameters of the non-leaf nodes. For trees with more levels, he suggests starting from the leaf nodes and moving up the tree to sequentially estimate the parameters at each level. His approach makes a single pass over the tree and exploits the fact that for a customer at a particular node, the conditional choice probabilities are similar to the choice probabilities under the multinomial logit model. In contrast, we maximize the log-likelihood function over one component at a time and cycle over each component multiple times. Thus, we make multiple passes over the tree. To our knowledge, for the d-level nested logit model, the concavity of the log-likelihood function as a function of the parameters at each level does not appear in the existing literature and Theorem H.1 closes this gap. Furthermore, this result shows that maximizing the log-likelihood function only over the parameters at one level is a tractable problem.

H.2 Demonstration of Improvement in Prediction Accuracy in a Single Data Set

In this section, we provide a numerical example to demonstrate that by increasing the number of levels in the tree, it is possible to significantly improve the prediction accuracy of the d-level nested logit model.

In our example, we consider $n = 8$ products and a mixture of logits model with $K = 20$ distinct customer segments. There are two sets of parameters associated with the mixture of logits model: the proportions of each segment $\boldsymbol{\xi} = (\xi^1, ..., \xi^K)$ with $\sum_{k=1}^{K} \xi^k = 1$ and the preference weights $\boldsymbol{v}^k = (v_1^k, ..., v_n^k) \in \mathbb{R}_+^n$ that customers in segment k associate with the different products for $k = 1, ..., K$. The parameters of the mixture of logits model are presented in Table 8.

k	ξ^k	Preference Weight of Each Product							
		v_1^k	v_2^k	v_3^k	v_4^k	v_5^k	v_6^k	v_7^k	v_8^k
1	1.30E-1	1.00E0	5.44E-1	7.81E-1	2.43E-1	7.67E-1	1.51E-1	1.80E-1	4.23E0
2	1.64E-7	6.64E-3	2.41E0	1.63E-1	6.48E0	4.21E0	1.51E0	3.05E0	9.20E0
3	1.44E-2	1.49E0	3.46E-1	3.54E-1	2.52E-1	1.10E-1	1.15E0	8.29E-2	7.29E0
4	1.59E-5	6.78E-1	1.42E0	4.50E0	2.00E-1	7.42E0	1.06E0	3.87E-1	3.70E0
5	1.18E-4	1.71E0	1.57E0	1.72E0	6.33E-1	4.60E0	1.62E0	9.11E-2	2.44E-1
6	9.20E-3	2.54E-1	9.83E-2	2.01E0	3.05E0	2.39E0	8.38E-1	8.53E-2	4.87E-1
7	1.09E-6	1.68E0	1.18E0	6.70E0	7.91E0	1.19E0	4.59E-1	1.64E-1	8.47E0
8	1.01E-5	1.46E0	7.45E-1	1.13E0	4.82E-1	1.59E-2	6.62E0	2.36E0	7.03E0
9	3.33E-6	6.41E-1	1.46E0	7.74E-1	5.47E0	1.97E0	1.29E0	2.43E-1	4.54E-1
10	1.82E-15	1.43E0	7.15E0	1.72E-1	1.59E-2	7.14E-1	5.00E0	4.52E0	8.94E0
11	1.46E-3	4.18E-1	8.45E-1	1.19E0	4.49E0	1.67E0	3.47E0	1.99E-1	4.72E0
12	3.07E-1	6.34E0	4.90E-1	2.61E-1	1.56E0	3.36E-1	1.90E-1	7.61E-2	4.57E-1
13	2.22E-10	7.35E0	1.31E0	5.84E0	3.50E0	4.85E0	7.63E0	2.83E-1	3.05E-2
14	3.90E-1	3.73E0	2.64E0	1.31E0	3.20E0	2.20E-1	2.19E0	1.81E-1	2.07E0
15	4.74E-13	9.52E-1	5.41E0	1.27E0	1.85E0	6.47E0	4.97E-1	6.20E-1	4.51E-1
16	5.60E-7	9.94E-1	1.12E0	9.09E-1	3.88E0	3.35E-1	4.79E0	3.91E0	7.82E0
17	1.01E-1	8.07E0	5.43E0	2.72E0	7.77E-2	1.36E-2	3.15E0	9.42E0	4.35E-1
18	4.81E-2	1.01E0	1.70E0	2.21E0	9.22E-1	1.22E0	6.31E0	3.60E-1	6.01E0
19	5.51E-11	1.54E0	9.15E-1	2.24E-1	1.62E0	6.81E-1	6.60E0	4.99E0	9.13E-2
20	4.16E-8	8.30E0	2.44E0	2.13E0	9.68E-1	6.68E0	1.41E0	2.95E-1	7.08E-1

Table 8: Parameters of the mixture of logits model.

We consider a collection of 36 unique offered assortments, each of which consists of 6 or 7 products. We generate a training data set that consists of assortments offered and the choices of 10,000 customers, assuming that the choices of the customers are governed by the mixture of logits model. We estimate the parameters for the d-level nested logit models and select the best fitted d-level models for $d = 1, 2, 3$.

Under the mixture of logits model, given an offered assortment S, a customer chooses product $\ell \in S$ with probability $\theta_\ell(S \,|\, \mathsf{Real}) = \sum_{k=1}^{K} \xi^k \frac{v_\ell^k}{\sum_{j \in S} v_j^k}$. For every offered assortment S, we denote the top-selling product under the mixture of logits model by

$$\mathsf{TopSell}(S \,|\, \mathsf{Real}) = \arg\max_{\ell \in S} \theta_\ell(S \,|\, \mathsf{Real}) \ .$$

Moreover, for $d = 1, 2, 3$, we use $\theta_\ell^d(S \,|\, \mathsf{Est})$ to denote the probability that a customer chooses product $\ell \in S$ under the best fitted d-level nested logit model. We define the predicted top-selling product under the d-level nested logit model as

$$\mathsf{TopSell}^d(S \,|\, \mathsf{Est}) = \arg\max_{\ell \in S} \theta_\ell^d(S \,|\, \mathsf{Est}) \ .$$

We are interested in comparing the predictive accuracy among the one, two and three-level nested logit models. In particular, we want to evaluate the ability of the one, two and three-level nested logit models to extrapolate and predict the top-selling products against the ground truth – the mixture of logits model – over any arbitrary offered assortment. Note that we choose to compare our fitted nested logit models directly against the ground truth instead of using validation and testing data sets because in our generated training data, the ground truth, i.e., the parameters of the underlying mixture of logits model, is already available.

We compare the predictive accuracy of the fitted models are follows. Note that the full assortment of 8 products gives rise to $2^8 - 1 = 255$ non-empty unique subsets of products. For each subset of products S, we calculate $\mathsf{TopSell}(\mathsf{S} \,|\, \mathsf{Real})$, the top-selling product under the mixture of logits model and $\mathsf{TopSell}^d(\mathsf{S} \,|\, \mathsf{Est})$, the predicted top-selling product under each of the fitted d-level nested logit models, for $d = 1, 2, 3$. Since there are 255 unique subsets of products, we have 255 choices of top-selling products for each of the mixture of logits model and the fitted one, two and three-level nested logit models. The results for each d-level nested logit model can be summarized in the form of a $n \times n$ confusion matrix. The confusion matrices for the one, two and three-level nested logit models are shown in Figure 4.

For $d = 1, 2, 3$, in each confusion matrix for the d-level nested logit model, the rows correspond to the actual top-selling products under the mixture of logits model; the columns correspond to the predicted top-selling products under the fitted d-level nested logit model. For $i = 1, ..., 8$ and $j = 1, ..., 8$, we denote each entry in the confusion matrix by

$$\mathsf{ConfMatr}_{i,j}^d = \sum_{S \subseteq \{1,...,8\}} \mathbb{1}_{\left\{\mathsf{TopSell}(S \,|\, \mathsf{Real}) = i, \mathsf{TopSell}^d(S \,|\, \mathsf{Est}) = j\right\}} \ ,$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function. Then, $\mathsf{ConfMatr}_{i,j}^d$ corresponds to the number of times that the d-level nested logit model predicts the actual top-selling product i as j. Thus, the diagonal elements in the confusion matrix show the number of correct predictions for each top-selling product. Any non-zero off-diagonal element $\mathsf{ConfMatr}^d_{i,j}, i \neq j$, is highlighted in bold, and it represents the number of times that the d-level nested logit model falsely predicts the actual top-selling product i as j. If all the off-diagonal entries in the confusion matrix are zero, then the d-level nested logit model predicts with 100% accuracy.

We summarize the confusion matrix for each d-level nested logit model. The second last column in each table that contains the confusion matrix for d-level model reports the total number of predictions for each top-selling product under the mixture of logit models. The last column reports the total number of false predictions under the fitted d-level nested logit model for each top-selling product. Finally, the last row reports the total number of predictions for each top-selling product under the fitted d-level nested logit model.

The results in Figure 4 show that the one-level nested logit model predicts 32 correctly out of 64 times, i.e., with $50\% = 100\% \times \frac{32}{64}$ accuracy when the actual top-selling product is product 4. Similarly, the one-level nested logit model predicts with $50\% = 100\% \times \frac{1}{2}$ accuracy when the actual top-selling product is product 5. The two-level nested logit model performs much better than the one-level, improving the predictive accuracy to $77\% = 100\% \times \frac{49}{64}$ and $100\% = 100\% \times \frac{64}{64}$ when the actual selling products are 4 and 5, respectively. The three-level nested logit model further improves the predictive accuracy for top-selling product 4 to 100%. Overall, the one-level nested logit model falsely predicts 35 out of 255 times; the two-level nested logit model falsely predicts only 17 times; the three-level nested logit model significantly reduces the false predictions to just 2 times.

H.3 Numerical Experiments for an Ensemble of Data Sets

In the this section, we provide a numerical study to demonstrate that using larger number of levels in the tree can help better predict the customer choice behavior. We generate data consisting of the assortment offers and choices of multiple customers, under the assumption that the choices of the customers are governed by a mixture of logits model. Using the generated data, we fit the parameters of one, two and three-level nested logit models and compare the performance of the fitted models in predicting the customer choice behavior.

Numerical Setup: Throughout our numerical study, the number of products is $n = 8$. Without loss of generality, we can treat one of the products as the no-purchase option. We generate 50 data sets as the training data sets in our numerical study. To generate each data set, we use a mixture of logits model with $K = 20$ customer segments. As explained before, the parameters of

Confusion Matrix for $d = 1$									No. of Pred.	No. of
Actual Top-	Predicted Top-Selling Product								under Mix. of	False
Selling Product	1	2	3	4	5	6	7	8	Logits Model	Pred.
1	128	0	0	0	0	0	0	0	128	0
2	0	16	0	0	0	0	0	**1**	17	1
3	0	0	4	0	0	**1**	0	0	5	1
4	0	0	0	32	0	0	0	**32**	64	32
5	0	0	0	0	1	0	**1**	0	2	1
6	0	0	0	0	0	7	0	0	7	0
7	0	0	0	0	0	0	1	0	1	0
8	0	0	0	0	0	0	0	31	31	0
No. of Pred. under Est. Model	128	16	4	32	1	8	2	64	255	35

Confusion Matrix for $d = 2$									No. of Pred.	No. of
Actual Top-	Predicted Top-Selling Product								under Mix. of	False
Selling Product	1	2	3	4	5	6	7	8	Logits Model	Pred.
1	128	0	0	0	0	0	0	0	128	0
2	0	16	0	0	0	0	0	**1**	17	1
3	0	0	4	0	0	**1**	0	0	5	1
4	0	0	0	49	0	0	0	**15**	64	15
5	0	0	0	0	2	0	0	0	2	0
6	0	0	0	0	0	7	0	0	7	0
7	0	0	0	0	0	0	1	0	1	0
8	0	0	0	0	0	0	0	31	31	0
No. of Pred. under Est. Model	128	16	4	49	2	8	1	47	255	17

Confusion Matrix for $d = 3$									No. of Pred.	No. of
Actual Top-	Predicted Top-Selling Product								under Mix. of	False
Selling Product	1	2	3	4	5	6	7	8	Logits Model	Pred.
1	128	0	0	0	0	0	0	0	128	0
2	0	16	0	0	0	0	0	**1**	17	1
3	0	0	4	0	0	**1**	0	0	5	1
4	0	0	0	64	0	0	0	0	64	0
5	0	0	0	0	2	0	0	0	2	0
6	0	0	0	0	0	7	0	0	7	0
7	0	0	0	0	0	0	1	0	1	0
8	0	0	0	0	0	0	0	31	31	0
No. of Pred. under Est. Model	128	16	4	64	2	8	1	32	255	2

Figure 4: Confusion matrices for the best fitted one, two and three-level nested logit models.

the mixture of logits model are the proportions of each segment $\boldsymbol{\xi} = (\xi^1, \ldots, \xi^K) \in [0,1]^K$ with $\sum_{k=1}^K \xi^k = 1$, and for each segment $k = 1, \ldots, K$, the preference weights $\boldsymbol{v}^k = (v_1^k, \ldots, v_n^k) \in \mathbb{R}_+^n$ that customers in segment k associate with the different products. Under the mixture of logits model, if we offer the assortment S, then a customer chooses product $\ell \in S$ with probability $\theta_\ell(S \mid \mathsf{Real}) = \sum_{k=1}^K \xi^k \frac{v_\ell^k}{\sum_{j \in S} v_j^k}$.

For each data set, we randomly choose the parameters of the mixture of logits model as follows. We generate $\boldsymbol{\xi} = (\xi^1, \ldots, \xi^K)$ from a K-dimensional Dirichlet distribution with parameters $(\alpha, \ldots, \alpha)$, so that the expectation of ξ^k is equal to $\frac{1}{K}$ for all $k = 1, \ldots, K$. The coefficient of variation of ξ^k is equal to $\sqrt{\frac{K-1}{K\alpha+1}}$, and we calibrate the parameter α so that the coefficient of variation is equal to 2.5. To introduce heterogeneity among different customer segments, we use the following strategy to generate preference weights $(\boldsymbol{v^1}, \ldots, \boldsymbol{v^K})$. For each product $\ell = 1, \ldots, n$, we sample σ_ℓ from the uniform distribution over $[0,1]$. If σ_ℓ is close to one, then product ℓ is a specialty product and customers in different segments associate significantly different preference weights with this product. On the other hand, if σ_ℓ is close to zero, then product ℓ is a staple product and customers in different segments evaluate product ℓ similarly, associating similar preference weights with this product. To generate preference weights with such characteristics, for each product ℓ and segment k, we sample ϑ_ℓ^k from the uniform distribution over $[0,10]$ and let $v_\ell^k = (1-\sigma_\ell)\,\vartheta_\ell^k$ with probability 1/2 and $v_\ell^k = (1+\sigma_\ell)\,\vartheta_\ell^k$ with probability 1/2. In this case, if σ_ℓ is close to zero, then v_ℓ^k takes a value close to ϑ_ℓ^k. If, however, σ_ℓ is close to one, then v_ℓ^k either takes a value close to zero or a value close to $2\,\vartheta_\ell^k$. Therefore, the parameter σ_ℓ indeed captures how much the preference weight of product ℓ differs among the different customer segments.

Once we choose the parameters of the mixture of logits model, we generate choices of 10,000 customers, assuming that the choices of the customers are governed by this choice model. In particular, for each one of the 10,000 customers, we randomly sample an assortment of products offered to this customer. We consider 36 unique offered assortments, each of which consists either 6 or 7 out of the 8 products. Given that the customer is offered a particular assortment, we sample the choice of the customer within this assortment by using the choice probabilities $\{\theta_\ell(S \,|\, \mathsf{Real}) : \ell = 1, \ldots, n,\ S \subseteq \{1, \ldots, n\}\}$. The assortments offered to 10,000 customers and the products chosen by these customers out of the assortments form one data set.

Parameter Estimation and Model Selection: For each data set, we estimate the parameters of the one, two and three-level nested logit models by minimizing the Akaike Information Criterion (AIC) value. Introduced by Akaike (1974), the AIC value is defined as

$$\mathrm{AIC}^d = 2(n_p^d - 2\mathcal{L}^d),$$

where n_p^d denotes the number of parameters in the model, and $\mathcal{L}^d$ denotes the value of the log-likelihood function in (14). The model with the minimum AIC value minimizes the expected Kullback-Leibler distance between the model and the underlying mixture of logits model.

By our construction as elaborated in the following two paragraphs, in each data set, n_p^d is a constant for fixed d. Therefore, minimizing the AIC value is equivalent to maximizing the log-likelihood function, for the models with the same number of levels. To maximize the log-likelihood function, we sequentially maximize over the parameters at each level of the tree and stop when no further improvement in the value of the log-likelihood is possible. We use the convex optimization software CVX to solve the optimization problems; see Grant and Boyd (2008, 2013).

Note that there can be many different tree structures to consider for the two and three-level nested logit models, corresponding to different organizations of the products in the tree. We use the following simple heuristic to obtain a good tree structure and report the results for the best tree structure that we find. We only consider the tree structures in which the non-leaf nodes in the same level have equal number of children nodes. In particular, for the two-level nested logit model, we only consider the tree structures in which the root node has two children and each node in level one has four children. When we consider such tree structures, there are $\binom{8}{4}/2 = 35$ distinct ways of placing the products in the tree. We estimate the parameters of the two-level nested logit model for each one of these 35 tree structures. We choose the model that yields the smallest AIC value as the best two-level nested logit model.

For the three-level nested logit model, we only consider the tree structures where each non-leaf node has two children. When we consider such tree structures, there are $\frac{\binom{8}{4}}{2}\left(\frac{\binom{4}{2}}{2}\right)^2 = 315$ distinct ways of placing the products in the tree. Since the number of possible three-level tree structures is large, we use a simple heuristic to focus on a small number of tree structures. In particular, by using the approach in the previous paragraph, we find the best two-level tree structure. Once we have the best two-level tree structure, we only consider the three-level tree structures that are compatible with the best two-level tree structure. In other words, we let a and b be the two nodes in the first level of the best two-level tree structure. We use $\bar{N}_a$ and $\bar{N}_b$ to denote the sets of products that are included in the subtrees rooted at nodes a and b. In the three-level tree structures we consider, the sets of products that are included in the subtrees rooted at the two nodes in the first level of the tree are identical to $\bar{N}_a$ and $\bar{N}_b$. Considering only such three-level tree structures reduces the number of possibilities to $\frac{\binom{4}{2}}{2}\frac{\binom{4}{2}}{2} = 9$. Once we restrict our attention to 9 possible three-level tree structures, similar to the approach in the previous paragraph, we estimate the parameters of the three-level nested logit model for each one of these 9 tree structures, and choose the model that has the smallest AIC value as the best three-level nested logit model.

In each data set, we also compare the AIC values of the one, two and three-level nested logit models. For the one, two and three-level models we consider, $n_p^1 = 8$, $n_p^2 = 10$ and $n_p^3 = 14$. We find that in general, a nested logit model with a larger number of levels has lower AIC value than a nested logit model with smaller number of levels. This means that the issue of overfitting is under control, if we use AIC to estimate the parameters.

Summary of Results: For each data set, we create three confusion matrices for the chosen one,

Comparison of Total Errors	Avg.	S.E.
$100 \times \frac{\mathsf{TotalErr}^1(k) - \mathsf{TotalErr}^2(k)}{\mathsf{TotalErr}^1(k)}$	8.7	4.0
$100 \times \frac{\mathsf{TotalErr}^1(k) - \mathsf{TotalErr}^3(k)}{\mathsf{TotalErr}^1(k)}$	14.9	4.3

Table 9: Comparison of the total errors for one, two and three-level nested logit models.

two and three-level nested logit models, similar to those in Figure 4. We use the following metric to summarize the confusion matrices for all 50 data sets. For each data set $k = 1, ..., 50$, we define the number of falsely predicted top-selling products by the d-level nested logit model as

$$\mathsf{TotalErr}^d(k) = \sum_{i \neq j} \mathsf{ConfMatr}^d_{i,j}(k) \ .$$

We use $\mathsf{TotalErr}^d(k)$ as our prediction accuracy metric for d-level nested logit model in data set k. Thus, the smaller the value of $\mathsf{TotalErr}^d(k)$, the better predictive accuracy of the d-level nested logit model. If $\mathsf{TotalErr}^d(k) = 0$, then the d-level nested logit model predicts the top-selling products perfectly, with 100% accuracy. We are interested in the percentage reduction in the number of falsely predicted products as we increase the number of levels in the tree.

We report our findings in Table 9. The first row in the table focuses on the percentage deviation between the prediction accuracy metrics for the one-level and two-level nested logit models. In this row, the second column shows the percentage deviation between the prediction accuracy metrics for the one-level and two-level nested logit models, averaged over all 50 data sets. In other words, using $\mathsf{TotalErr}^d(k)$ to denote the prediction accuracy metric of the fitted d-level nested logit model for problem instance k, the second column gives the average of the figures $\{100 \times \frac{\mathsf{TotalErr}^1(k) - \mathsf{TotalErr}^2(k)}{\mathsf{TotalErr}^1(k)} : k = 1, ..., 50\}$. The third column in the first row gives the standard error of the percentage deviation between the prediction accuracy metrics for the one-level and two-level nested logit models, over all 50 data sets. Then, the entries in the first row indicate how much we can reduce the number of falsely predicted top-selling products of the one-level nested logit model by using a two-level one. The format of the second row is similar to that of the first row, but the second row focuses on the percentage deviation between the prediction accuracy metrics for the one-level and three-level nested logit models.

The results in Table 9 indicate that on average, having a greater number of levels in the tree can significantly improve the prediction accuracy of the one-level nested logit model. Note that in all of the 50 data sets, we do not observe any instance in which the one-level nested logit model predicts with 100% accuracy. On average, the two-level nested logit model reduces the number of falsely predicted products of the one-level model by 8.7%. By using a three-level nested logit model, we can improve the predictive accuracy even more significantly. On average, the number of falsely predicted products is reduced by 14.9% when we use a three-level nested logit model. We note that there are data sets in which the number of falsely predicted products of the lower-level nested logit model is smaller than that of the higher-level nested logit model. Due to finite sample estimation errors as we only estimated a limited number of two-level and three-level models, it is not possible to guarantee that a higher-level nested logit model always improves the prediction accuracy of a lower-level model, but our numerical study indicates that using a three-level nested logit model can provide improvements in a great majority of the cases and the improvements can be significant.

H.4 Proof of Theorem H.1

In this section, we present the proof of Theorem H.1.

Proof: Using (8) and denoting $V_j(S^t)$ by V_j^t for simplicity, we write the log-likelihood function as

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\chi}_1,\ldots,\boldsymbol{\chi}_{d-1},\boldsymbol{\beta}) \\
&= \sum_{t=1}^{T}\log\left(\prod_{h=1}^{d}\frac{V^t_{\mathsf{An}(c^t,h)}}{\sum_{j\in\mathsf{Sibling}(\mathsf{An}(c^t,h))}V_j^t}\right)\\
&= \sum_{t=1}^{T}\log\left(e^{\beta_{c^t}}\times\prod_{h=2}^{d}\frac{1}{\left(\sum_{j\in\mathsf{Sibling}(\mathsf{An}(c^t,h))}V_j^t\right)^{1-\eta_{\mathsf{An}(c^t,h-1)}}}\times\frac{1}{\sum_{j\in\mathsf{Sibling}(\mathsf{An}(c^t,1))}V_j^t}\right)\\
&= \sum_{t=1}^{T}\left[\beta_{c^t}-\sum_{h=2}^{d}\left(1-\eta_{\mathsf{An}(c^t,h-1)}\right)\log\left(\sum_{j\in\,\mathsf{Children}(\mathsf{An}(c^t,h-1))}V_j^t\right)-\log\left(\sum_{j\in\,\mathsf{Children}(\mathsf{root})}V_j^t\right)\right]\\
&= \sum_{t=1}^{T}\left[\beta_{c^t}-\sum_{h=2}^{d}\left(1-\eta_{\mathsf{An}(c^t,h-1)}\right)\log\left(\left[V^t_{\mathsf{An}(c^t,h-1)}\right]^{1/\eta_{\mathsf{An}(c^t,h-1)}}\right)-\log\left(\sum_{j\in\,\mathsf{Children}(\mathsf{root})}V_j^t\right)\right],
\end{aligned}$$

where, letting Leaf and $\mathsf{NonLeaf}$ respectively be the leaf and non-leaf nodes in the tree, we use the fact that

$$V_j^t=\begin{cases}0 & \text{if}\quad j\in\mathsf{Leaf}\quad\text{and}\quad j\notin S^t\\ e^{\beta_j} & \text{if}\quad j\in\mathsf{Leaf}\quad\text{and}\quad j\in S^t\\ \left(\sum_{i\in\,\mathsf{Children}(j)}V_i^t\right)^{\eta_j} & \text{if}\quad j\in\mathsf{NonLeaf}\,.\end{cases}$$

In the expression above, the preference weight V_j^t for each node j depends on the parameters $(\boldsymbol{\chi}_1,\ldots,\boldsymbol{\chi}_{d-1},\boldsymbol{\beta})$, but we suppress this dependence for clarity. For each customer t and non-leaf node j, let y_j^t be defined recursively as follows:

$$y_j^t = \begin{cases} \log\left(\sum_{\ell \in \mathsf{Children}(j) \cap S^t} e^{\beta_\ell} \right) & \text{if } \mathsf{depth}(j) = d-1 \\ \log\left(\sum_{i \in \mathsf{Children}(j)} e^{\eta_i \cdot y_i^t} \right) & \text{if } \mathsf{depth}(j) < d-1 \ , \end{cases}$$

where we set $\log 0 = -\infty$. Note that if none of the descendants of node j at the leaves belong to S^t, then $y_j^t = -\infty$ by our construction.

Claim 1: For each customer t and non-leaf node j, $V_j^t = e^{\eta_j \cdot y_j^t}$.

We will prove the claim by induction on the depth of node j. If $\mathsf{depth}(j) = d-1$, then we have that $V_j^t = \left(\sum_{\ell \in \mathsf{Children}(j) \cap S^t} e^{\beta_\ell} \right)^{\eta_j} = e^{\eta_j \cdot y_j^t}$, where the last equality follows from the definition of y_j^t. Suppose the result is true for all nodes at depth $h+1$. Consider an arbitrary node j at depth $h < d-1$. By induction hypothesis, $V_j^t = \left(\sum_{\ell \in \mathsf{Children}(j)} V_\ell^t \right)^{\eta_j} = \left(\sum_{\ell \in \mathsf{Children}(j)} e^{\eta_\ell \cdot y_\ell^t} \right)^{\eta_j} = e^{\eta_j \cdot y_j^t}$, where the last equality follows from the definition of y_j^t. This establishes the claim.

Using the claim above, the log-likelihood function is given by

$$\begin{aligned} \mathcal{L}(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta}) &= \sum_{t=1}^T \left[\beta_{c^t} - \sum_{h=2}^d \left(1 - \eta_{\mathsf{An}(c^t,h-1)}\right) y_{\mathsf{An}(c^t,h-1)}^t - \log\left(\sum_{j \in \mathsf{Children}(\mathsf{root})} e^{\eta_j \cdot y_j^t} \right) \right] \\ &= \sum_{t=1}^T \left[\beta_{c^t} - \sum_{h=1}^{d-1} \left(1 - \eta_{\mathsf{An}(c^t,h)}\right) y_{\mathsf{An}(c^t,h)}^t - y_{\mathsf{root}}^t \right] \\ &= \sum_{t=1}^T \left[\beta_{c^t} - \sum_{h=0}^{d-1} \left(1 - \eta_{\mathsf{An}(c^t,h)}\right) y_{\mathsf{An}(c^t,h)}^t \right] \\ &= \sum_{\ell=1}^n M_\ell \, \beta_\ell \ - \sum_{h=0}^{d-1} \sum_{j \,:\, \mathsf{depth}(j)=h} (1 - \eta_j) \sum_{t \,:\, S^t \cap \mathsf{Descendant}(j) \neq \varnothing} y_j^t \ , \end{aligned} \tag{15}$$

where M_ℓ is the total sales for product ℓ and $\mathsf{Descendant}(j)$ denotes all nodes that are descendants of j, excluding j itself. Note that y_j^t is a function of $\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta}$, so we will now write $y_j^t(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$ to emphasize this dependence. Since $\eta_j \in (0,1]$ for all $j \in \mathsf{NonLeaf}$ and noting the expression above, the following claim establishes the first part of the theorem.

Claim 2: For fixed $\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}$, for all t and j, $y_j^t(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$ is convex in $\boldsymbol{\beta}$.

We will prove the claim by induction on the depth of node j. The result is true when $\mathsf{depth}(j) = d-1$ by the convexity of the log-sum-exp function; see Boyd and Vandenberghe (2004). Suppose the result is true for all nodes of depth $h+1$. Consider node j at level h. It is a well-known results that if the functions $g_i(\cdot)$ are convex, then $\log(\sum_i e^{g_i(\cdot)})$ is also convex. Since $y_j^t = \log(\sum_{i\in \mathsf{Children}(j)} e^{\eta_i \cdot y_k^t})$, we have that $y_j^t(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$ is also convex in $\boldsymbol{\beta}$, completing the induction. This establishes the claim. The following claim is useful to show the second part of the theorem.

Claim 3: For all t, h and j, $y_j^t\,(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{h-1}, \boldsymbol{\chi}_h, \boldsymbol{\chi}_{h+1}, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$ is convex in $\boldsymbol{\chi}_h$.

Fix an arbitrary customer t and $h \in \{1, \ldots, d-1\}$. We show by induction on the node depth that y_j^t is convex in $\boldsymbol{\chi}_h$ for all nodes j. If $\mathsf{Descendant}(j) \cap S^t = \varnothing$, then y_j^t is constant at $-\infty$, which is convex. So, it suffices to consider only the nodes that are ancestors of products in S^t. First, consider nodes j in levels $d-1,\ d-2, \ldots, h$. By definition, y_j^t depends only on $(\eta_i : i \in \mathsf{Descendant}(j))$. In this case, the result automatically holds because y_j^t is independent of $\boldsymbol{\chi}_h$.

Now, consider node j such that $\mathsf{depth}(j) = h-1$. By definition, it holds that $y_j^t = \log\left(\sum_{k\in \mathsf{Children}(j)} e^{\eta_k \cdot y_k^t}\right)$. For each $k \in \mathsf{Children}(j)$, $\mathsf{depth}(k) = h$ and y_k^t is independent of $\boldsymbol{\chi}_h$, which implies that y_j^t depends on $\boldsymbol{\chi}_h$ through $(\eta_\ell : \ell \in \mathsf{Children}(j))$ via the log-sum-exp function. Therefore, y_j^t is convex in $\boldsymbol{\chi}_h$.

Next, consider node j at level $h-2$. Since $y_j^t = \log\left(\sum_{k\in \mathsf{Children}(j)} e^{\eta_k \cdot y_k^t}\right)$ and $\mathsf{depth}(k) = h-1$ for all $k \in \mathsf{Children}(j)$, y_j^t only depends on $\boldsymbol{\chi}_h$ through y_k^t, which we have just shown to be convex in $\boldsymbol{\chi}_h$. Therefore, y_j^t is also convex in $\boldsymbol{\chi}_h$. Exactly the same argument applies to all nodes in levels $h-3, h-4, \ldots, 1$ and this establishes the desired claim.

We will now use Claim 3 to establish the second part of the theorem. Note that $\boldsymbol{\chi}_h$ appears in the log-likelihood function in (15) in two terms

$$-\sum_{j:\mathsf{depth}(j)=h} (1-\eta_j)\left[\sum_{t\,:\,S^t\cap \mathsf{Descendant}(j)\neq\varnothing} y_j^t\right] \text{ and } -\sum_{g=1}^{h-1}\sum_{j:\mathsf{depth}(j)=g} (1-\eta_j)\left[\sum_{t\,:\,S^t\cap \mathsf{Descendant}(j)\neq\varnothing} y_j^t\right].$$

The first term is an affine function in $\boldsymbol{\chi}_h$ because for nodes j in level h, $\sum_{t\,:\,S^t\cap \mathsf{Descendant}(j)\neq\varnothing} y_j^t$ is independent of $\boldsymbol{\chi}_h$, so the first term is concave in $\boldsymbol{\chi}_h$. For the second term, $\boldsymbol{\chi}_h$ only appears through y_j^k, which we know from Claim 3 to be convex in $\boldsymbol{\chi}_h$, and thus, the second term is concave in $\boldsymbol{\chi}_h$. Therefore, the log-likelihood function is concave in $\boldsymbol{\chi}_h$, as desired. $\square$

It is not difficult to generate counterexamples to show that the log-likelihood function is not jointly concave in $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$. Note that fixing $\boldsymbol{\beta}$, the log-likelihood function in (15) contains a product of $(1-\eta_j)$ and y_j^t, which depends on $(\eta_\ell : \ell \in \mathsf{Descendant}(j))$. Product of two variables is not a jointly concave or convex function of the two variables. Thus, it is not difficult to see that the log-likelihood function is not jointly concave in $(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_{d-1}, \boldsymbol{\beta})$.

Online Appendix References

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19 (6): 716–723.

Boyd, S., and L. Vandenberghe. 2004. *Convex Optimization.* Cambridge, UK: Cambridge University Press.

Davis, J. M., G. Gallego, and H. Topaloglu. 2014. Assortment optimization under variants of the nested logit model. *Operations Research* 62 (2): 250–273.

Grant, M., and S. Boyd. 2008. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, ed. V. Blondel, S. Boyd, and H. Kimura, Lecture Notes in Control and Information Sciences, 95–110. Springer-Verlag Limited. `http://stanford.edu/~boyd/graph_dcp.html`.

Grant, M., and S. Boyd. 2013, September. CVX: Matlab software for disciplined convex programming, version 2.0 beta. `http://cvxr.com/cvx`.

Li, G., and P. Rusmevichientong. 2014. A greedy algorithm for the two-level nested logit model. *Operations Research Letters* 42 (5): 319–324.

Talluri, K., and G. J. van Ryzin. 2004. Revenue management under a general discrete choice model of consumer behavior. *Management Science* 50 (1): 15–33.

Train, K. 2003. *Discrete Choice Methods with Simulation.* New York, NY: Cambridge University Press.