A Greedy Algorithm for the Two-Level Nested Logit Model

Guang Li, Paat Rusmevichientong*

Marshall School of Business, University of Southern California

Abstract

We consider the assortment optimization problem under the classical two-level nested logit model. We establish a necessary and sufficient condition for the optimal assortment and develop a simple and fast greedy algorithm that iteratively removes at most one product from each nest to compute an optimal solution.

Keywords: revenue management, customer choice, assortment optimization, nested logit model, greedy algorithm, optimality condition

1. Introduction

In an assortment optimization problem, a firm wishes to offer the most profitable set of products to satisfy customer demand. As shown in an excellent survey in [1], this problem has many important applications in retail and revenue management. Our paper focuses on assortment optimization in revenue management, where, unlike in the retail setting, the focus is primarily on product selections, without any inventory replenishment.

The seminal work in [2] introduced the first assortment optimization problem under the multinomial logit (MNL) model [3]. Although the MNL model admits a tractable solution, it suffers from the independence of irrelevant alternatives (IIA) property, which may lead to a paradox when alternatives are closely related and result in biased estimates [4]. The deficiency of the MNL model has led many researchers to consider assortment optimization under more complex choice models; see, for example, [5–7]. The nested logit model extends the MNL model by grouping similar alternatives into a nest and allowing differential substitution patterns within and across nests, thereby partially relaxing the IIA restriction [8]. Under the nested logit model, a consumer first chooses a nest of products and then chooses a product from the chosen nest. Existing literature on assortment optimization under the nested logit model makes use of a linear programming framework; see, for example, [9–11].

In this paper, we present an alternative solution to assortment optimization under the two-level nested logit model. We establish a necessary and sufficient condition for an optimal assortment (Theorem 2.3), which is, to our knowledge, the first such optimality condition for this problem. In addition, we reveal a "lumpy" structure of the optimal assortment: surprisingly, within each nest, a certain set of "consecutive" products always appears together in the optimal solution. Moreover, as shown in Theorem 2.4, by looking at a certain index of each product, we can determine *in advance* which products will appear together! Exploiting the optimality condition and the "lumpy" structure, our greedy algorithm iteratively removes (at most) one product from each nest and terminates with an optimal assortment in $O(n\,m\log m)$ time, for the problem with m nests, with each nest having n products. This is the fastest known running time for this problem.

Moreover, our assortment optimization can be used as a subroutine for the network revenue management problem under the nested logit model. A revenue model over a single-leg flight is first considered in [2]. For extensions of this work to multiple resources, see, for example, [12–14]. The main approach in these papers is to formulate a variety of deterministic linear programming approximations and solve the linear programs using column generation. The column generation subproblem in this setting is precisely the assortment optimization considered in this paper, when customers choose according to a two-level nested logit model.

*Corresponding Author: Marshall School of Business, University of Southern California, Los Angeles, CA 90089, Email: rusmevic@marshall.usc.edu

We perform sensitivity analysis to study the effect of reducing product revenues on the optimal assortment. We show that the entire profile of the optimal revenue and assortment when every product revenue is reduced by δ can be computed simultaneously for all $\delta \in \mathbb{R}_+$ in a single run of the greedy algorithm in $O(n\,m\,\log m)$ time. Moreover, we show that the optimal solution to the multi-period capacity allocation problem with total allocation periods T and total capacity C can be computed in $O(TC + n\,m\log m)$ operations, by applying the greedy algorithm just once.

Problem Description. We have m nests indexed by $i \in \{1, 2, \ldots, m\}$, and each nest i has n products indexed by $j \in \{1, 2, \ldots, n\}$. All of the analysis easily extends to the case in which some nests have fewer than n products. A product j in nest i is denoted by ji, and in settings in which the nest index is clear from context, we refer to it simply as product j. We denote the no-purchase option by product 0, which is the only product in nest 0. Each product ji has a revenue $r_{ji} \geq 0$, with $r_0 = 0$. Without loss of generality, we assume that the products in each nest are sorted in a descending order of revenues; that is, for each nest $i \in \{1, 2, \ldots, m\}$, $r_{1i} \geq r_{2i} \geq \cdots \geq r_{ni} \geq 0$.

We assume that the customer is a utility maximizing agent, and the utility that she assigns to each product is:

$$U_0 = \lambda_0 + \epsilon_0 \qquad \text{and} \qquad U_{ji} = \lambda_{ji} + \epsilon_i + \epsilon_{ji}, \qquad i = 1, \ldots, m,\ \ j = 1, \ldots, n,$$

where λ_{ji} and λ_0 are the deterministic components and $\{\epsilon_0, \epsilon_i, \epsilon_{ji} : i = 1, \ldots, m,\ \ j = 1, \ldots, n\}$ are independent random errors. The random variable ϵ_0 has a Gumbel distribution with a location parameter of zero and a scaling parameter of one, which we write as $\epsilon_0 \sim \text{Gumbel}(0, 1)$. For each nest i, $\epsilon_{ji} \sim \text{Gumbel}(0, 1/\tau_i)$ for all j, where $\tau_i \in (0, 1]$ is a nest-specific parameter. Finally, for each i, ϵ_i is independent of $\{\epsilon_{ji} : j = 1, \ldots, n\}$, and it is assumed to have a distribution such that $\epsilon_i + \epsilon_{ji} \sim \text{Gumbel}(0, 1)$ for all j; see [15] for the existence of such a distribution.

Let $\boldsymbol{S} = (S_1, \ldots, S_m)$ denote an assortment of products, where S_i is a subset of the n products $\{1i, 2i, \ldots, ni\}$ in nest i, for all $i \in \{1, 2, \ldots, m\}$. As the nest index i is explicit, we simply write $S_i \subseteq \{1, 2, \ldots, n\}$. For each product ji, let $v_{ji} = e^{\lambda_{ji}/\tau_i}$ denote its preference weight, and let $v_0 = e^{\lambda_0}$. As shown in [16], the probability $Q_{ji}(\boldsymbol{S})$ that a customer chooses a product $j \in S_i$ under assortment $\boldsymbol{S}$ is given by:

$$Q_{ji}(\boldsymbol{S}) = Q_{j|i}(S_i) \times Q_i(\boldsymbol{S}), \quad \text{where} \quad Q_{j|i}(S_i) = \frac{v_{ji}}{\sum_{\ell \in S_i} v_{\ell i}} \quad \text{and} \quad Q_i(\boldsymbol{S}) = \frac{\left(\sum_{\ell \in S_i} v_{\ell i}\right)^{\tau_i}}{v_0 + \sum_{k=1}^{m} \left(\sum_{\ell \in S_k} v_{\ell k}\right)^{\tau_k}}.$$

Here, $Q_i(\boldsymbol{S})$ denotes the probability that a customer chooses a product in nest i, and $Q_{j|i}(S_i)$ denotes the conditional probability of the customer's selecting product ji, given that nest i is chosen.

Thus, the total expected revenue $\mathsf{Rev}(\boldsymbol{S})$ for any assortment $\boldsymbol{S} = (S_1, \ldots, S_m)$ is given by:

$$\mathsf{Rev}(\boldsymbol{S}) = \sum_{i=1}^{m} \sum_{j \in S_i} r_{ji} Q_{ji}(\boldsymbol{S}) = \sum_{i=1}^{m} Q_i(\boldsymbol{S}) \sum_{j \in S_i} r_{ji} Q_{j|i}(S_i) = \sum_{i=1}^{m} Q_i(\boldsymbol{S}) \mathsf{Rev}_i(S_i),$$

where $\mathsf{Rev}_i(S_i) = \sum_{j \in S_i} r_{ji} Q_{j|i}(S_i) = \frac{\sum_{\ell \in S_i} r_{\ell i} v_{\ell i}}{\sum_{\ell \in S_i} v_{\ell i}}$ denotes the expected revenue from S_i in nest i, and we set it to be zero if S_i is empty. The assortment optimization problem is thus given by:

$$Z^* \ = \ \max_{\boldsymbol{S} = (S_1, \ldots, S_m)} \mathsf{Rev}(\boldsymbol{S}),$$

and we denote an optimal assortment by $\boldsymbol{S}^* = (S_1^*, \ldots, S_m^*)$ and let $Z^* = \mathsf{Rev}(\boldsymbol{S}^*)$ denote the optimal revenue. If there are ties, we can choose the optimal assortment according to any predetermined tie-breaking rule.

2. Characterization of the Optimal Assortments

Let $\mathcal{N}_+ = \{\{1\}, \{1, 2\}, \ldots, \{1, 2, \ldots, n\}\}$ denote the collection of revenue-ordered subsets. As shown in the following lemma, at the optimal solution, any nonempty nest is revenue-ordered. The lemma is a restatement of Theorem 4 in [9], and we refer the reader to the paper for the details of the proof.

Lemma 2.1 (Optimal Nest Structure). *For $i = 1, \ldots, m$, if $S_i^* \neq \varnothing$, then $S_i^* \in \mathcal{N}_+$.*

Although Lemma 2.1 shows that $S_i^* \in \{\varnothing\} \cup \mathcal{N}_+$ for all i, exhaustive search is not feasible because there are $(n + 1)^m$ possible configurations of revenue-ordered subsets among m nests. To develop an efficient algorithm, we

will establish a *necessary and sufficient* condition for an optimal assortment. For any integers k_1 and k_2, let

$$[k_1, k_2] = \begin{cases} \{k_1, k_1+1, \ldots, k_2\} & \text{if } k_1 \le k_2, \\ \varnothing & \text{otherwise.} \end{cases}$$

Also, for each nest i and $1 \le k_1 \le k_2 \le n$, let $G_i(k_1, k_2)$ be defined by:

$$G_i(k_1, k_2) = \mathsf{Rev}_i([1, k_2]) \; - \; \frac{\mathsf{Rev}_i\left([1, k_2]\right) - \mathsf{Rev}_i\left([k_1, k_2]\right)}{f_i\left(1 \; - \; \sum_{\ell \in [k_1, k_2]} v_{\ell i} / \sum_{\ell \in [1, k_2]} v_{\ell i}\right)},$$

where $f_i : [0,1] \to [0,1]$ is a continuous, strictly increasing function defined by:

$$f_i(x) = \begin{cases} (x^{1-\tau_i} - x)/(1-x) & \text{if } x \in [0,1) \\ \tau_i & \text{if } x = 1 \end{cases}.$$

Moreover, we define $0/0 = 0$, thus $G_i(1, k_2) = \mathsf{Rev}_i([1, k_2])$ for all k_2.

Interpretation of $G_i(\cdot,\cdot)$. Suppose $S_i = [1, k_2]$, and consider $k_1 \in \{1, \ldots, k_2\}$. The function $G_i([k_1, k_2])$ can be used to evaluate the net effect of removing $[k_1, k_2]$ from nest i on the total revenue, without any computation on the resultant assortment. This intuition is confirmed in the following lemma, which establishes a precise condition on when we can use $G_i(\cdot,\cdot)$ to determine if a subset can be removed from S_i to improve the total revenue.

Lemma 2.2 (When Is Removing Subsets Beneficial?). *For any assortment $\boldsymbol{S} = (S_1, \ldots, S_m)$, a collection of subsets $\{A_i \subseteq S_i \; : \; i = 1, 2, \ldots, m\}$ can be removed from $\boldsymbol{S}$ to achieve a greater revenue if and only if*

$$\frac{\sum_{i=1}^m \left[\, V(S_i)^{\tau_i} \mathsf{Rev}_i(S_i) - V(S_i \setminus A_i)^{\tau_i} \mathsf{Rev}_i(S_i \setminus A_i) \,\right]}{\sum_{i=1}^m \left[\, V(S_i)^{\tau_i} - V(S_i \setminus A_i)^{\tau_i} \,\right]} \; < \; \mathsf{Rev}(\boldsymbol{S}),$$

where for any set X, $V(X)$ denotes the sum of the preference weights of the products in X.

Moreover, if $S_i = [1, p_i]$ with $p_i \ge 1$, $A_i = [k_i, p_i]$ with $k_i \le p_i$, and $A_\ell = \varnothing$ for all $\ell \ne i$, then removing A_i from S_i leads to a greater revenue if and only if $G_i(k_i, p_i) < \mathsf{Rev}(\boldsymbol{S})$; removing A_i from S_i leads to the same revenue if and only if $G_i(k_i, p_i) = \mathsf{Rev}(\boldsymbol{S})$.

Proof. For $i = 1, \ldots, m$, let $\widehat{S}_i = S_i \setminus A_i$. Assume that $A_i \ne \varnothing$ for some i; otherwise, the result is trivially true. As $v_{ji} > 0$ for all ji, $\sum_{i=1}^m V(\widehat{S}_i) < \sum_{i=1}^m V(S_i)$. By definition, $\mathsf{Rev}(\widehat{\boldsymbol{S}}) = \frac{\sum_{i=1}^m V(\widehat{S}_i)^{\tau_i} \mathsf{Rev}_i(\widehat{S}_i)}{v_0 + \sum_{i=1}^m V(\widehat{S}_i)^{\tau_i}}$. Then,

$$\begin{aligned} \mathsf{Rev}(\boldsymbol{S}) = \frac{\sum_{i=1}^m V(S_i)^{\tau_i} \mathsf{Rev}_i(S_i)}{v_0 + \sum_{i=1}^m V(S_i)^{\tau_i}} &= \frac{v_0 + \sum_{i=1}^m V(\widehat{S}_i)^{\tau_i}}{v_0 + \sum_{i=1}^m V(S_i)^{\tau_i}} \times \mathsf{Rev}(\widehat{\boldsymbol{S}}) \\ &+ \frac{\sum_{i=1}^m \left[V(S_i)^{\tau_i} - V(\widehat{S}_i)^{\tau_i}\right]}{v_0 + \sum_{i=1}^m V(S_i)^{\tau_i}} \times \frac{\sum_{i=1}^m \left[V(S_i)^{\tau_i} \mathsf{Rev}_i(S_i) - V(\widehat{S}_i)^{\tau_i} \mathsf{Rev}_i(\widehat{S}_i)\right]}{\sum_{i=1}^m \left[V(S_i)^{\tau_i} - V(\widehat{S}_i)^{\tau_i}\right]}. \end{aligned}$$

Thus, $\mathsf{Rev}(\boldsymbol{S})$ is a convex combination of $\mathsf{Rev}(\widehat{\boldsymbol{S}})$ and $\frac{\sum_{i=1}^m \left[\, V(S_i)^{\tau_i} \mathsf{Rev}_i(S_i) - V(\widehat{S}_i)^{\tau_i} \mathsf{Rev}_i(\widehat{S}_i) \,\right]}{\sum_{i=1}^m \left[\, V(S_i)^{\tau_i} - V(\widehat{S}_i)^{\tau_i} \,\right]}$, which gives the desired result.

Let $S_i = [1, p_i]$, $A_i = [k_i, p_i] \ne \varnothing$, and $A_\ell = \varnothing$ for all $\ell \ne i$. As $f_i\left(1 - \frac{V(A_i)}{V(S_i)}\right) = \frac{V(S_i)^{\tau_i} - V(\widehat{S}_i)^{\tau_i}}{V(\widehat{S}_i)^{\tau_i - 1} V(S_i) - V(\widehat{S}_i)^{\tau_i}}$ and $\mathsf{Rev}_i(\widehat{S}_i) = \frac{V(S_i)}{V(\widehat{S}_i)} \mathsf{Rev}_i(S_i) + \left(1 - \frac{V(S_i)}{V(\widehat{S}_i)}\right) \mathsf{Rev}_i(A_i)$, it is easy to verify that

$$V(S_i)^{\tau_i} \mathsf{Rev}_i(S_i) - V(\widehat{S}_i)^{\tau_i} \mathsf{Rev}_i(\widehat{S}_i) = \left(V(S_i)^{\tau_i} - V(\widehat{S}_i)^{\tau_i}\right) G_i(k_i, p_i).$$

Therefore, $\frac{V(S_i)^{\tau_i} \mathsf{Rev}_i(S_i) - V(\widehat{S}_i)^{\tau_i} \mathsf{Rev}_i(\widehat{S}_i)}{V(S_i)^{\tau_i} - V(\widehat{S}_i)^{\tau_i}} \le \mathsf{Rev}(\boldsymbol{S})$ if and only if $G_i(k_i, p_i) \le \mathsf{Rev}(\boldsymbol{S})$. The inequality is strict if and only if $G_i(k_i, p_i) < \mathsf{Rev}(\boldsymbol{S})$. □

Lemma 2.2 is consistent with our interpretation of $G_i(\cdot,\cdot)$: for $p_i \geq 1$, if $S_i = [1, p_i]$ and $G_i(k_i, p_i) < \mathsf{Rev}(\boldsymbol{S})$, then removing $[k_1, k_2]$ from S_i is beneficial. If $S_i^* = [1, p_i]$ in an optimal assortment $\boldsymbol{S}^*$, then $G_i(k_i, p_i) \geq \mathsf{Rev}(\boldsymbol{S}^*)$ should hold for all $k_i \leq p_i$. The intuition is confirmed by the main result of this section, which is stated in the following theorem. The theorem establishes a necessary and sufficient condition for an optimal assortment, and it forms the basis for our greedy algorithm in Section 3.

Theorem 2.3 (Optimality Condition). *Consider any assortment $\boldsymbol{S} = ([1, p_1], \ldots, [1, p_m])$ such that $S_i^* \subseteq [1, p_i]$ for all i. Then, $\boldsymbol{S}$ is optimal if and only if for every nest i such that $p_i \geq 1$,*

$$\min_{j=1,\ldots,p_i} G_i(j, p_i) \geq \mathsf{Rev}(\boldsymbol{S}).$$

Proof. Suppose that $\boldsymbol{S} = (S_1, \ldots, S_m)$ is an optimal solution, where $S_i = [1, p_i]$ for all i. If $p_i \geq 1$, then $S_i \neq \varnothing$. As $\boldsymbol{S}$ is optimal, removing a subset $[k_i, p_i]$ from S_i cannot improve the revenue. Thus, it follows from Lemma 2.2 that $G_i(k_i, p_i) \geq \mathsf{Rev}(\boldsymbol{S})$ for all $k_i \in \{1, 2, \ldots, p_i\}$, which is the desired result.

To establish sufficiency, consider any assortment $\boldsymbol{S} = (S_1, \ldots, S_m)$ such that $S_i^* \subseteq S_i = [1, p_i]$ for all i, and $\boldsymbol{S}$ satisfies the condition of the theorem. We will show that $\mathsf{Rev}(\boldsymbol{S}) = Z^*$. For each i, let $E_i = S_i \backslash S_i^* = [k_i, p_i]$ for some index k_i. By our hypothesis, for every nest i such that $E_i \neq \varnothing$, $G_i(k_i, p_i) \geq \mathsf{Rev}(\boldsymbol{S})$, thus removing E_i from S_i does not increase the revenue. It then follows from Lemma 2.2 that $\mathsf{Rev}(\boldsymbol{S}) \leq \frac{V(S_i)^{\tau_i}\mathsf{Rev}_i(S_i) - V(S_i \setminus E_i)^{\tau_i}\mathsf{Rev}_i(S_i \setminus E_i)}{V(S_i)^{\tau_i} - V(S_i \setminus E_i)^{\tau_i}}$, where for any set X, $V(X)$ is the sum of the preference weights of the products in X. Thus,

$$\begin{aligned}
\mathsf{Rev}(\boldsymbol{S}) &\leq \min_{i:E_i \neq \varnothing} \frac{V(S_i)^{\tau_i}\mathsf{Rev}_i(S_i) - V(S_i \setminus E_i)^{\tau_i}\mathsf{Rev}_i(S_i \setminus E_i)}{V(S_i)^{\tau_i} - V(S_i \setminus E_i)^{\tau_i}} \\
&\leq \frac{\sum_{i:E_i \neq \varnothing} [V(S_i)^{\tau_i}\mathsf{Rev}_i(S_i) - V(S_i \setminus E_i)^{\tau_i}\mathsf{Rev}_i(S_i \setminus E_i)]}{\sum_{i:E_i \neq \varnothing} [V(S_i)^{\tau_i} - V(S_i \setminus E_i)^{\tau_i}]},
\end{aligned}$$

where the second inequality follows because for any $\boldsymbol{x} \in \mathbb{R}^k$ and $\boldsymbol{y} \in \mathbb{R}^k_+$, $\frac{\sum_{i=1}^k x_i}{\sum_{i=1}^k y_i} \geq \min_{i=1,\ldots,k} \frac{x_i}{y_i}$, and for all i, $V(S_i)^{\tau_i} - V(S_i \setminus E_i)^{\tau_i} \geq 0$. Note that if $E_i = \varnothing$, then $S_i = S_i^*$. It then follows from Lemma 2.2 that $\mathsf{Rev}(\boldsymbol{S}) \geq \mathsf{Rev}(S_1 \setminus E_1, S_2 \setminus E_2, \ldots, S_m \setminus E_m) = \mathsf{Rev}(S_1^*, \ldots, S_m^*) = Z^*$, which is the desired result. □

Surprisingly, as shown in the following theorem, the optimal assortment in each nest is "lumpy," with certain consecutive products always appearing together. Moreover, for each nest i, by looking at the index $G_i(j, j)$ of each product j, we can determine *in advance* which products will appear together!

Theorem 2.4 (Lumpiness of the Optimal Assortments). *For every nest i, if there exist products j and k, such that $j < k$ and*

$$G_i(j, j) < G_i(j+1, j+1) < \cdots < G_i(k, k),$$

then either $\{j, j+1, \ldots, k\} \subseteq S_i^*$ or $\{j, j+1, \ldots, k\} \cap S_i^* = \varnothing$.

Proof. It suffices to prove the result when $k = j + 1$; that is, we need to show that if $G_i(j, j) < G_i(j+1, j+1)$, then either $\{j, j+1\} \subseteq S_i^*$ or $\{j, j+1\} \cap S_i^* = \varnothing$. There are two cases to consider.

Case 1: $j + 1 \in S_i^*$. In this case, $\{j, j+1\} \subseteq S_i^*$ because $S_i^* \in \mathcal{N}_+$ by Lemma 2.1.

Case 2: $j + 1 \notin S_i^*$. Suppose that $j \in S_i^*$. Then, $S_i^* = [1, j]$. Let $\widehat{\boldsymbol{S}} = (S_1^*, \ldots, [1, j+1], \ldots, S_m^*)$. By definition, $\mathsf{Rev}(\widehat{\boldsymbol{S}}) \leq \mathsf{Rev}(\boldsymbol{S}^*)$, and as $\boldsymbol{S}^*$ is obtained from $\widehat{\boldsymbol{S}}$ by removing product $j+1$ from nest i, it follows from Lemma 2.2 that $G_i(j+1, j+1) \leq \mathsf{Rev}(\widehat{\boldsymbol{S}})$. So, we have that $G_i(j, j) < G_i(j+1, j+1) \leq \mathsf{Rev}(\widehat{\boldsymbol{S}}) \leq \mathsf{Rev}(\boldsymbol{S}^*)$, which implies that $\mathsf{Rev}\,(S_1^*, \ldots, S_i^* \setminus \{j\}, \ldots, S_m^*) > \mathsf{Rev}(\boldsymbol{S}^*)$. This contradicts the optimality of $\boldsymbol{S}^*$. Thus, $j \notin S_i^*$. □

Note that the "lumpiness" property is nest-specific and is independent of v_0. We can view the occurrence of "lumpiness" as the consequence of the property of $G_i(\cdot,\cdot)$ and the revenue-ordered property of the optimal assortment. Suppose $S_i \supseteq S_i^*$, and $G_i(j, j) < G_i(j+1, j+1)$ for products j and $j+1$ in S_i. If $G_i(j+1, j+1) < \mathsf{Rev}(\boldsymbol{S})$, then it is beneficial to remove both products j and $j+1$ from nest i. Otherwise, the revenue-ordered property implies that if $j + 1 \in S_i^*$, then $j \in S_i^*$. Thus, products j and $j+1$ always appear together in the optimal solution; that is, they can be merged into a single product when computing the optimal assortment, as illustrated in the next section.

3. A Greedy Algorithm

Our proposed GREEDY ALGORITHM generates a sequence of assortments $\{\boldsymbol{S}^t : t = 0, 1, \ldots\}$, terminating with an assortment that satisfies the optimality condition in Theorem 2.3. In Stage 1 of the algorithm, we exploit the lumpy structure of the optimal assortments shown in Theorem 2.4, by combining products that will always appear together into a single group. In Stage 2, we make use of the optimality condition and iteratively remove at most one product from each nest until an optimal assortment is obtained. A formal description of the algorithm is given as follows.

Stage 1 (Lumping): Compute the index $G_i(j, j)$ for every product ji. For $i = 1, \ldots, m$, if $G_i(j, j) < G_i(j+1, j+1)$, then it follows from Theorem 2.4 that either $\{j, j+1\} \subseteq S_i^*$ or $\{j, j+1\} \cap S_i^* = \varnothing$. Thus, we can combine the two products, and replace products j and $j+1$ with a single "new" product with a revenue $\mathsf{Rev}_i([j, j+1])$ and a preference weight $v_{ji} + v_{j+1,i}$. Assign the new product the index j, and calculate the new $G_i(j, j)$. Repeat this process until we obtain a list of indices $G_i(j, j)$ that is non-increasing in j. Without loss of generality, assume that at the end of Stage 1, each nest i has n products such that $G_i(1, 1) \geq G_i(2, 2) \geq \cdots \geq G_i(n, n)$.

Stage 2 (Removal): Let $\boldsymbol{S}^1 = ([1, n], \ldots, [1, n])$. For iteration $t \geq 1$, given $\boldsymbol{S}^t = ([1, J_1^t], [1, J_2^t], \ldots, [1, J_m^t])$, let

$$\mathsf{Min}^t \;=\; \min_{i\,:\,J_i^t \geq 1} G_i(J_i^t, J_i^t) \qquad \text{and} \qquad \mathsf{Index}^t \;=\; \arg\min_{i\,:\,J_i^t \geq 1} G_i(J_i^t, J_i^t).$$

If $\mathsf{Min}^t \geq \mathsf{Rev}(\boldsymbol{S}^t)$, the algorithm terminates and outputs $\boldsymbol{S}^t$. Otherwise, if $\mathsf{Min}^t < \mathsf{Rev}(\boldsymbol{S}^t)$, then the algorithm generates a new assortment $\boldsymbol{S}^{t+1} = (S_1^{t+1}, \ldots, S_m^{t+1})$ as follows:

$$S_i^{t+1} = \begin{cases} S_i^t & \text{if } \; i \neq \mathsf{Index}^t \\ S_i^t \setminus \{J_i^t\} = [1, J_i^t - 1] & \text{if } \; i = \mathsf{Index}^t \,. \end{cases}$$

Thus, in each iteration, we remove J_i^t from S_i^t if $G_i(J_i^t, J_i^t) = \mathsf{Min}^t$ **and** the product J_i^t violates the optimality condition given in Theorem 2.3; that is, $G_i(J_i^t, J_i^t) < \mathsf{Rev}(\boldsymbol{S}^t)$.

The following lemma shows that $\boldsymbol{S}^t$ always contains the optimal assortment.

Lemma 3.1 (Containment). *For all t, $\boldsymbol{S}^* \subseteq \boldsymbol{S}^t$.*

Proof. We will prove the result by induction. It is true for $t = 1$ by our construction. Suppose that the lemma is true for $t > 1$; that is, $\boldsymbol{S}^* \subseteq \boldsymbol{S}^t$. We will show that $S_i^* \subseteq S_i^{t+1}$ for all i. Consider an arbitrary nest i. There are two cases to consider: $S_i^* = S_i^t$ and $S_i^* \subset S_i^t$.
Case 1: Suppose that $S_i^* = S_i^t = [1, J_i^t]$. If $J_i^t = 0$, then $i \neq \mathsf{Index}^t$ and $S_i^{t+1} = S_i^t$ by our construction. So, suppose that $J_i^t \geq 1$. As $\mathsf{Rev}\,(\boldsymbol{S}^*) \geq \mathsf{Rev}(S_1^*, \ldots, S_i^* \setminus \{J_i^t\}, \ldots, S_m^*)$, $G_i(J_i^t, J_i^t) \geq \mathsf{Rev}(\boldsymbol{S}^*) \geq \mathsf{Rev}(\boldsymbol{S}^t)$ follows from Lemma 2.2 and the optimality of $\boldsymbol{S}^*$. As $G_i(J_i^t, J_i^t) \geq \mathsf{Rev}(\boldsymbol{S}^t)$, $S_i^{t+1} = S_i^t$, which is the desired result.
Case 2: Suppose that $S_i^* \subset S_i^t = [1, J_i^t]$. Then, $J_i^t \notin S_i^*$ and $J_i^t \geq 1$. As $S_i^* \in \mathcal{N}_+$, $S_i^* \subseteq [1, J_i^t - 1]$. By our construction, we will either remove J_i^t from S_i^t or do nothing, and in both cases, we have $S_i^* \subseteq S_i^{t+1}$. □

The main result of this section is stated in the following theorem.

Theorem 3.2 (Correctness). *The* GREEDY ALGORITHM *terminates with an optimal assortment in $O(n\, m \log m)$ time.*

Proof. We will first establish the running time of the algorithm. Note that $\mathsf{Rev}_i([1, j])$ is a convex combination of $\mathsf{Rev}_i([1, j-1])$ and r_{ji}, and by definition, $G_i(j, j) = \mathsf{Rev}_i([1, j]) - \frac{\mathsf{Rev}_i([1,j]) - r_{ji}}{f_i\left(1 \,-\, v_{ji} / \sum_{\ell=1}^{j} v_{\ell i}\right)}$. For simplicity, assume that we can compute the function $f_i(x)$ for each x and the index $G_i(j, j)$ for every product ji in $O(1)$ time. For each nest i, we will show that the "lumping" process takes $O(n)$ operations. The lumping process requires two types of operations: 1) comparison between two indices and 2) merging of two products. Whenever we merge two products to create a new one, the total number of products is reduced by one. As we start with n products, the number of mergings is at most $n - 1$. We perform two types of comparisons. Starting from product 1, if the ordering of indices is correct up to product j, we compare $G_i(j, j)$ *forward* with $G_i(j+1, j+1)$. If no violation occurs, then $G_i(j, j) \geq G_i(j+1, j+1)$, and we proceed to product $j+1$. If a violation occurs, then we must merge products j and $j+1$ to create a new product j. To ensure the correct ordering of indices, we need to compare the new index $G_i(j, j)$ *backward* with $G_i(j-1, j-1)$. Given n products, there are $O(n)$ forward comparisons. As a backward

comparison only occurs after each merging, and there are $O(n)$ mergings, the number of backward comparisons is also $O(n)$. Thus, the total number of operations is $O(n)$. With m nests, the total running time for Stage 1 is $O(nm)$.

In Stage 2, the algorithm will continue to run as long as a single product is removed from any nest. Given nm products, the algorithm will terminate in $O(nm)$ iterations. We will show that each iteration takes $O(\log m)$ time.

At the beginning of Stage 2, we create a *self-balancing binary search tree* (SB-BST) with m nodes, where for $i = 1, 2, \ldots, m$, node i in the tree corresponds to the index $G_i(n, n)$. This takes $O(m \log m)$ operations; see Chapter 6 in [17] for more details. We use the SB-BST as our data structure because such a tree always maintains a height of $O(\log m)$, allowing for efficient search operations. In each iteration $t \geq 1$, we perform the following three operations:

SEARCH: The algorithm searches for the nest with the minimum index $G_i(J_i^t, J_i^t)$ in $O(\log m)$ time under SB-BST.

DELETE: If the minimum index $G_i(J_i^t, J_i^t)$ is greater than or equal to the revenue of the current assortment, the algorithm terminates. Otherwise, the algorithm removes the product J_i^t from S_i^t in nest i and also removes node i from the tree. The removal is done in $O(\log m)$ operations because the tree may need to re-balance its heights.

INSERT: If the product J_i^t in nest i is removed and $J_i^t > 1$, the algorithm adds a new node with a corresponding index $G_i(J_i^t - 1, J_i^t - 1)$ into the tree. Again, the insertion of a new node in SB-BST takes $O(\log m)$ operations.

Thus, each iteration takes $O(\log m)$ time. As we have $O(nm)$ iterations, the total running time is $O(n\, m \log m)$.

Let $\boldsymbol{S} = ([1, p_1], \ldots, [1, p_m])$ denote the assortment at the termination of the algorithm. For every nest i such that $p_i \geq 1$, by our construction, $G_i(p_i, p_i) \geq \mathsf{Rev}(\boldsymbol{S})$. To complete the proof, we will show that $\boldsymbol{S}$ satisfies the optimality condition in Theorem 2.3; that is, $\min_{j=1,2,\ldots,p_i} G_i(j, p_i) \geq \mathsf{Rev}(\boldsymbol{S})$ for each i such that $p_i \geq 1$. Suppose on the contrary that $\min_{j=1,2,\ldots,p_i} G_i(j, p_i) < \mathsf{Rev}(\boldsymbol{S})$. Let $k_i \in \{1, 2, \ldots, p_i - 1\}$ denote the largest index at which the optimality condition is violated; that is, $G_i(k_i, p_i) < \mathsf{Rev}(\boldsymbol{S}) \leq G_i(k_i + 1, p_i)$. Then it follows from Lemma 2.2 that $\mathsf{Rev}(S_1, \ldots, S_i \setminus [k_i, p_i], \ldots, S_m) > \mathsf{Rev}(\boldsymbol{S}) \geq \mathsf{Rev}(S_1, \ldots, S_i \setminus [k_i + 1, p_i], \ldots, S_m)$. As $S_i \setminus [k_i, p_i] = [1, k_i - 1]$ and $S_i \setminus [k_i + 1, p_i] = [1, k_i]$, by Lemma 2.2, $G_i(k_i, k_i) < \mathsf{Rev}(S_1, \ldots, S_i \setminus [k_i + 1, p_i], \ldots, S_m) \leq \mathsf{Rev}(\boldsymbol{S})$. As we lump the products in Stage 1 and $k_i < p_i$, we have $G_i(p_i, p_i) \leq G_i(k_i, k_i) < \mathsf{Rev}(\boldsymbol{S})$, which contradicts our hypothesis on p_i ! Therefore, $\min_{j=1,2,\ldots,p_i} G_i(j, p_i) \geq \mathsf{Rev}(\boldsymbol{S})$. □

4. Sensitivity Analysis

In this section, we investigate how the optimal assortment changes with product revenues. Our goal is to tractably compute the optimal assortment that maximizes the total revenue when the revenue of every product is reduced by δ, for all $\delta \in \mathbb{R}_+$ simultaneously. That is, we want to solve the following optimization problem:

$$Z(\delta) = \max_{\boldsymbol{S}=(S_1,\ldots,S_m)} \left\{ \sum_{i=1}^{m} \sum_{j \in S_i} Q_{ji}(\boldsymbol{S})(r_{ji} - \delta) \right\} = \max_{\boldsymbol{S}=(S_1,\ldots,S_m)} \left\{ \mathsf{Rev}(\boldsymbol{S}) - [1 - Q_0(\boldsymbol{S})]\delta \right\},$$

for all $\delta \in \mathbb{R}_+$, where $Q_0(\boldsymbol{S})$ is the probability that a customer chooses the no-purchase option.

Before we present the main result of this section, we first describe the structural properties of $Z(\delta)$ in the following lemma. The result is standard because $\{Z(\delta) : \delta \in \mathbb{R}_+\}$ is the maximum of linear functions, each of which is decreasing in δ, and we omit the detail of the proof.

Lemma 4.1. *For all $\delta \in \mathbb{R}_+$, the function $\delta \mapsto Z(\delta)$ is convex, decreasing and piecewise linear.*

Let $\boldsymbol{X}(\delta)$ represent the corresponding optimal assortment associated with the optimization problem for $Z(\delta)$. Throughout this section, we assume that $\boldsymbol{X}(\delta)$ corresponds to the optimal assortment obtained by running the GREEDY ALGORITHM with reduced revenue $r_{ji} - \delta$ for every product ji. Clearly, a naive application of the GREEDY ALGORITHM to compute $Z(\delta)$ for *every single* $\delta \in \mathbb{R}_+$ is infeasible. We will make use of the structural properties of $Z(\delta)$ to develop an efficient algorithm that computes the entire profile of $Z(\delta)$ and $\boldsymbol{X}(\delta)$, simultaneously for all $\delta \in \mathbb{R}_+$, with a running time of $O(n\, m \log m)$. The main result of this section is stated in the following proposition.

Proposition 4.2 (Computing $Z(\delta)$). *For all $\delta \in \mathbb{R}_+$, the function $\delta \mapsto Z(\delta)$ has at most nm breakpoints. Moreover, the entire profile of $Z(\delta)$, $\boldsymbol{X}(\delta)$ and the breakpoints can be computed simultaneously for all $\delta \in \mathbb{R}_+$ in a **single run** of the* GREEDY ALGORITHM *with a running time of $O(n\, m \log m)$.*

Before we prove Proposition 4.2, it is easy to verify that if we reduce the revenue of every product by δ, the index of each product ji is given by $G_i(j,j)-\delta$, which is simply the difference between the original index $G_i(j,j)$ defined in Section 2 and δ. Moreover, both the sequence of lumping and the ordering of the index of each product after lumping remain the same as we apply the GREEDY ALGORITHM with reduced revenue $r_{ji}-\delta$ for every product ji.

The following lemma shows the relationship between $\boldsymbol{X}(\delta)$ and $\boldsymbol{S}^*$, where $\boldsymbol{S}^*$ is the optimal assortment associated with the assortment optimization problem for Z^* as defined in Section 1.

Lemma 4.3 (Optimal Assortment for $Z(\delta)$). *For all $\delta \geq 0$, $\boldsymbol{X}(\delta) \subseteq \boldsymbol{X}(0) = \boldsymbol{S}^*$.*

Proof. The optimization problem for $Z(0)$ is identical to the one for Z^*; therefore, $\boldsymbol{X}(0) = \boldsymbol{S}^*$. Suppose the GREEDY ALGORITHM that computes $\boldsymbol{S}^*$ terminates in T iterations. For each $t = 1, \ldots, T$, let $\boldsymbol{S}^t = ([1, J_i^t], [1, J_2^t], \ldots, [1, J_m^t])$ denote the assortment in iteration t. Then, the optimality condition implies that $\min_{i:J_i^t \geq 1} G_i(J_i^t, J_i^t) < \mathsf{Rev}(\boldsymbol{S}^t)$, for $t < T$. Now compare this algorithm with the GREEDY ALGORITHM that computes $\boldsymbol{X}(\delta)$ with reduced revenue $r_{ji} - \delta$ for every product ji. As the ordering of the index of each product is constant for all $\delta \geq 0$, the sequence of products removed remains the same before the termination of either algorithm. Moreover, in each iteration $t < T$,

$$\min_{i:J_i^t \geq 1} (G_i(J_i^t, J_i^t) - \delta) < \mathsf{Rev}(\boldsymbol{S}^t) - \delta < \mathsf{Rev}(\boldsymbol{S}^t) - [1 - Q_0(\boldsymbol{S}^t)]\delta,$$

implying that the GREEDY ALGORITHM that computes $\boldsymbol{X}(\delta)$ terminates in iteration T or later. Thus, $\boldsymbol{X}(\delta) \subseteq \boldsymbol{S}^*$. □

We denote the breakpoints of the function $\delta \mapsto Z(\delta)$ by $\{\delta^t : t = 0, 1, \ldots\}$, $0 = \delta^0 \leq \delta^1 \leq \ldots$ and denote the optimal assortment for $\delta \in [\delta^t, \delta^{t+1}]$ by $\boldsymbol{X}^t = \boldsymbol{X}(\delta)$. Recall that the optimal assortment $\boldsymbol{X}^t$ for $\delta \in [\delta^t, \delta^{t+1}]$ corresponds to the optimal assortment obtained by running the GREEDY ALGORITHM with reduced revenue $r_{ji} - \delta$ for every product ji. Thus, every product in $\boldsymbol{X}^t$ has undergone the lumping process, and the index $G_i(j,j) - \delta$ in every nest i is non-increasing in j. Exploiting this fact, the next lemma shows that given $\boldsymbol{X}^{t-1}$, we can compute $\boldsymbol{X}^t$ simply by removing a product with the lowest index.

Lemma 4.4 (Computing an Optimal Assortment). *Suppose $\boldsymbol{X}^{t-1}$ is optimal for $\delta \in [\delta^{t-1}, \delta^t]$; an optimal assortment $\boldsymbol{X}^t$ for $\delta \in [\delta^t, \delta^{t+1}]$ can be obtained by removing a product with the minimum index from $\boldsymbol{X}^{t-1}$.*

Proof. As δ^t is a breakpoint, $\boldsymbol{X}^t \neq \boldsymbol{X}^{t-1}$. Thus by Lemma 4.3, $\boldsymbol{X}^t \subset \boldsymbol{X}^{t-1}$. Suppose the last product k in nest ℓ has the minimum index value in $\boldsymbol{X}^{t-1}$. By continuity, $Z(\delta^t) = \mathsf{Rev}(\boldsymbol{X}^{t-1}) - [1 - Q_0(\boldsymbol{X}^{t-1})]\delta^t = G_\ell(k,k) - \delta^t$, where the second equality follows from the optimality condition. As $G_\ell(k,k) - \delta^t$ is equal to $Z(\delta^t)$, which is the total revenue of $\boldsymbol{X}^{t-1}$ with reduced revenue $r_{ji} - \delta^t$ for every product ji, it follows from Lemma 2.2 that the removal of product $k\ell$ leaves the total revenue unchanged. That is, $Z(\delta^t) = \mathsf{Rev}(\boldsymbol{X}^{t-1} \setminus \{k\ell\}) - [1 - Q_0(\boldsymbol{X}^{t-1} \setminus \{k\ell\})]\delta^t$. Thus, $\boldsymbol{X}^{t-1} \setminus \{k\ell\}$ is also optimal at δ^t. Letting $\boldsymbol{X}^t = \boldsymbol{X}^{t-1} \setminus \{k\ell\}$ gives the desired result. □

Lemma 4.4 implies that starting with an optimal assortment $\boldsymbol{X}^0 = \boldsymbol{S}^*$, we can compute $\boldsymbol{X}^t$, for $t = 1, 2, \ldots$ sequentially *without* any knowledge on the breakpoints δ^t. The following lemma shows that given two consecutive optimal assortments $\boldsymbol{X}^{t-1}$ and $\boldsymbol{X}^t$, we can compute the breakpoint δ^t.

Lemma 4.5 (Computing Breakpoints). *Suppose $\boldsymbol{X}^{t-1}$ is an optimal assortment for $\delta \in [\delta^{t-1}, \delta^t]$, and $\boldsymbol{X}^t$ is an optimal assortment for $\delta \in [\delta^t, \delta^{t+1}]$. Then,*

$$\delta^t = \frac{\mathsf{Rev}(\boldsymbol{X}^{t-1}) - \mathsf{Rev}(\boldsymbol{X}^t)}{Q_0(\boldsymbol{X}^t) - Q_0(\boldsymbol{X}^{t-1})}.$$

Proof. By continuity, we have that $Z(\delta^t) = \mathsf{Rev}(\boldsymbol{X}^{t-1}) - [1 - Q_0(\boldsymbol{X}^{t-1})]\delta^t = \mathsf{Rev}(\boldsymbol{X}^t) - [1 - Q_0(\boldsymbol{X}^t)]\delta^t$, as both $\boldsymbol{X}^{t-1}$ and $\boldsymbol{X}^t$ give the same revenue $Z(\delta^t)$ at the breakpoint δ^t. Solving for δ^t gives us the desired result. □

Here is the proof of Proposition 4.2.

Proof. Starting with the full assortment, we first apply the GREEDY ALGORITHM to compute $\boldsymbol{X}^0 = \boldsymbol{X}(0) = \boldsymbol{S}^*$. Given $\boldsymbol{X}^0$, we obtain $\boldsymbol{X}^t$ for each $t \in \{1, 2, \ldots\}$ by applying Lemma 4.4. Then, given $\boldsymbol{X}^{t-1}$ and $\boldsymbol{X}^t$, we apply Lemma 4.5 to calculate the breakpoint δ^t. Finally, given $\boldsymbol{X}^{t-1}$ for $\delta \in [\delta^{t-1}, \delta^t]$, the optimal revenue is computed as $Z(\delta) = \mathsf{Rev}(\boldsymbol{X}^{t-1}) - [1 - Q_0(\boldsymbol{X}^{t-1})]\delta$. There are at most nm products in $\boldsymbol{X}^0$. As we remove a single product with the minimum index in each iteration, there are at most nm removals, corresponding to at most nm breakpoints of the function $\delta \mapsto Z^\delta$. The running time of the algorithm is $O(n\, m \log m)$ by Theorem 3.2. □

Application to the Multi-Period Capacity Allocation Problem. We apply our GREEDY ALGORITHM to solve the multi-period capacity allocation problem described in [2] under the two-level nested logit model. We denote the total capacity of seats on a flight leg by C and the total number of allocation periods by T. There are m groups of fare classes, with each group containing n fare classes. Each fare class j in group i has revenue r_{ji}, where $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, m$. Let $\boldsymbol{S} = (S_1, S_2, \ldots, S_m)$ denote the assortment of fare classes. In each period, a single customer chooses a fare class according to a two-level nested logit model. Let $J_t(x)$ denote the maximum expected revenue when we have x seats and t periods left. Then, $J_t(x)$ satisfies the following dynamic programming equation:

$$J_t(x) = \max_{\boldsymbol{S}=(S_1,\ldots,S_m)} \left\{ \sum_{i=1}^{m} \sum_{j \in S_i} Q_{ji}(\boldsymbol{S})(r_{ji} - \triangle J_{t-1}(x)) \right\} + J_{t-1}(x),$$

where $\triangle J_{t-1}(x) = J_{t-1}(x) - J_{t-1}(x-1)$, and $J_0 \equiv 0$. It is a standard result that $\triangle J_t(x) \geq 0$ for all x and t. Then, the first term on the right-hand side is exactly the optimization problem for $Z(\delta)$ with $\delta = \triangle J_{t-1}(x)$.

The traditional approach to solve the multi-period capacity allocation problem is by dynamic programming, in which $J_t(x)$ is computed for every single x and t, requiring a total running time of $O(TC\, n\, m \log m)$ if we apply the GREEDY ALGORITHM in each subproblem. As shown in the following proposition, we can do much better.

Proposition 4.6 (Solving Multi-Period Capacity Allocation). *Under the two-level nested logit model, the multi-period capacity allocation problem with total allocation periods T and total capacity C can be solved using a single run of the* GREEDY ALGORITHM *with a total running time of $O(TC + n\, m \log m)$.*

Proof. Computing $\{\boldsymbol{X}(\delta) : \delta \in \mathbb{R}_+\}$ and breakpoints $\{\delta^t : t = 1, 2, \ldots\}$ takes $O(n\, m \log m)$ time by Proposition 4.2. We can then build a hash table to store the intervals of breakpoints and the corresponding optimal assortments in $O(n\, m)$ time. Suppose that for every $\delta \in \mathbb{R}_+$, we can look up in the hash table for $\boldsymbol{X}(\delta)$ in $O(1)$ time. As x can take $C + 1$ values, for each t, the value function $J_t(\cdot)$ can be computed in $O(C)$ operations, simply by looking up the stored values in the hash table. There are T such value functions; thus it takes $O(TC)$ time to determine $J_t(\cdot)$ for all $t \in \{1, \ldots, T\}$. Therefore, the total running time is bounded by $O(TC + n\, m \log m)$. □

Acknowledgments

We would like to thank Huseyin Topaloglu for introducing us to the surprising connection between the assortment optimization problem under the two-level nested logit model and its equivalent linear programming formulation. We would like to thank the area editor, Sridhar Seshadri, the associate editor and the referee for their insightful and detailed comments. Their suggestions greatly improve the presentation and quality of our paper.

References

[1] A. G. Kök, M. L. Fisher, R. Vaidyanathan, Assortment planning: Review of literature and industry practice, in: Retail Supply Chain Management, Springer US, 2009, pp. 99–153.

[2] K. Talluri, G. van Ryzin, Revenue management under a general discrete choice model of consumer behavior, Mgmt. Sci. 50 (1) (2004) 15–33.

[3] D. McFadden, Conditional logit analysis of qualitative choice behavior, in: P. Zarembka (Ed.), Frontiers in Econometrics, Academic Press, 1974, pp. 105–142.

[4] K. Train, Discrete Choice Methods with Simulation, Cambridge University Press, 2009.

[5] V. Gaur, D. Honhon, Assortment planning and inventory decisions under a locational choice model, Mgmt. Sci. 52 (10) (2006) 1528–1543.

[6] D. Honhon, V. Gaur, S. Seshadri, Assortment planning and inventory decisions under stockout-based substitution, Oper. Res. 58 (5) (2010) 1364–1379.

[7] V. F. Farias, S. Jagabathula, D. Shah, Assortment optimization under general choice, Working Paper, MIT, Cambridge, MA (2011).

[8] D. McFadden, Modelling the choice of residential location, in: A. Karlqvist, L. Lundqvist, F. Snickars, J. Weibull (Eds.), Spatial Interaction Theory and Planning Models, North-Holland, Amsterdam, 1978, pp. 75–96.

[9] J. M. Davis, G. Gallego, H. Topaloglu, Assortment optimization under variants of the nested logit model, to appear in Oper. Res. (2014).

[10] P. Rusmevichientong, Z.-J. M. Shen, D. B. Shmoys, A PTAS for capacitated sum-of-ratios optimization, OR Letters 37 (4) (2009) 230–238.

[11] G. Gallego, H. Topaloglu, Constrained assortment optimization for the nested logit model, to appear in Mgmt. Sci. (2014).

[12] G. Gallego, G. Iyengar, R. Phillips, A. Dubey, Managing flexible products on a network, Computational Optimization Research Center Technical Report TR-2004-01, Columbia University, New York, NY (2004).

[13] Q. Liu, G. van Ryzin, On the choice-based linear programming model for network revenue management, M&SOM 10 (2) (2008) 288–310.

[14] S. Kunnumkal, Randomization approaches for network revenue management with customer choice behavior, to appear in POM (2013).

[15] N. S. Cardell, Variance components structures for the extreme-value and logistic distributions with application to models of heterogeneity, Econometric Theory 13 (02) (1997) 185–213.

[16] M. Ben-Akiva, S. Lerman, Discrete Choice Analysis: Theory and Application to Travel Demand, MIT Press, Cambridge, MA, 1985.

[17] D. E. Knuth, The Art of Computer Programming, 2nd Edition, Vol. 3, Addison-Wesley Professional, 1998.