Revenue Management under a Nonparametric Ranking Based Choice Model

Alice Paul
School of Operations Research and Information Engineering,
Cornell University, Ithaca, New York 14853, USA
ajp336@cornell.edu

James Mario Davis
School of Operations Research and Information Engineering,
Cornell University, Ithaca, New York 14853, USA
jmd388@orie.cornell.edu

Jacob Feldman
Olin Business School,
Washington University, St. Louis 63108, USA
jbfeldman@wustl.edu

September 28, 2015

Abstract

We consider revenue management problems when customers choose among the offered products according to a nonparametric ranking-based choice model. Under this nonparametric choice model, each customer class is distinguished by a unique ranking of the available products and an arrival probability. Given the arrival of a customer from a particular customer class, this customer will purchase the highest ranking offered product in her respective ranking list. To simplify the revenue management problems that we consider, we restrict the set of customer classes that can exist. Specifically, given a tree where the nodes are the products, we assume that the set of customer classes is derived from paths in the tree, where the order of nodes visited along each potential path gives the corresponding ranking list of a potential customer type. First, we study assortment problems, where the goal is to find a set of products to offer so as to maximize the expected revenue from each customer. We give a dynamic program to obtain the optimal solution. Second, we show how this dynamic programming formulation can be extended to consider the assortment problem when there is a constraint limiting the space consumption of the offered products. Third, we study network revenue management problems, where the goal is to adjust the set of offered products over a selling horizon when the sale of each product consumes a combination of a limited set of resources. A standard linear programming approximation of this problem includes one decision variable for each subset of products. We show that this linear program can be reduced to an equivalent one of substantially smaller size. We give an algorithm to recover the optimal solution to the original linear program from the reduced linear program.

1 Introduction

Overview and Motivation

A recent trend in the revenue management literature is incorporating customer choice into classic revenue management problems. Traditionally, the demand for a product is assumed to be independent of the availability of the other products. This assumption fails to accurately capture customer substitution behavior, the phenomena where a customer chooses an alternative product when their "ideal" product is not available. Customers' willingness to substitute creates a dependence between product demand and product availability. Customer choice models provide a way to model this interaction; a customer choice model maps any assortment of available products to the probabilities the products in the assortment are purchased. The parameters of the customer choice model are estimated from transaction data and the resulting, estimated parameters are used as the input in revenue management problems. A variety of customer choice models exist, each nuanced to capture different aspects of customer purchasing behavior. The ideal choice model is one which is simple to describe, easy to estimate, and whose corresponding revenue management problems admit tractable solutions.

In this paper we consider revenue management problems when customers choose among the offered products according to the nonparametric tree model, which we believe to possess all three of the aforementioned ideal traits. This choice model is a special case of the full nonparametric ranking based choice model first described in [**?**] and [**?**]. In the full nonparametric ranking based choice model, each customer class is distinguished by an arrival probability and a unique ranking on a subset of products. For the remainder of this paper, we use the term preference list to refer to the unique ranking of the products associated with a customer class. When presented with an assortment of products a customer will purchase the highest ranking offered product in her preference list; if there is no offered product in her preference list she leaves without making a purchase. Unfortunately, the full nonparametric ranking based choice model leads to intractable revenue management problems. For example, under this model there is no efficient algorithm to determine the assortment that maximizes a retailer's expected revenue, a fundamental problem.

The nonparametric tree model is a specific case of the full nonparametric model; the set of possible preference lists are paths in an underlying tree. To be more precise, given an undirected tree where each node corresponds to a unique product, the set of all potential customer types is characterized by the set of all possible paths in the tree. We restrict these paths to be linear in the sense that they must either move progressively towards or away from the root node. We formalize this model in Section 2.

The primary distinction of this model is that there are at most $O(n^2)$ customer classes, where n is the number of products under consideration; this greatly simplifies the application of the model in practice. In particular, because of the small number of customer classes, estimation procedures, like those described in [**?**] and [**?**], are highly efficient. By contrast, in the full nonparametric model

these procedures can become computationally intensive since the number of customer types grows exponentially in the number of products

In order to build revenue management systems, though, it is essential to solve the optimization problems that arise in a tractable fashion. We embed the tree model in common revenue management problems and provide polynomial-time algorithms. The combination of efficient estimation procedures for the model and tractable algorithms for optimization problems allows the tree model to form the basis of revenue management systems.

The tree model we consider is motivated by the tree model presented in [**?**]. They consider two customer choice models based on tress, the intree model and outree model, both of which are special cases of the model we consider. These models are similar to ours in that customer classes correspond to paths in an underlying tree. However, both the intree and outree models put significant restrictions on which paths are associated with permissible customer classes.

In the intree model all customer classes must be associated with paths that begin at an interior or leaf node of the tree and terminate at the root. The intree model is appropriate when customers substitute from specific, specialized products, to more general products. General products that are designed to appeal to a wide range of customers would be located towards the root of the tree, with the root being the most general. Products targeted to specific customer segments would be located towards the leaves of of the tree, with the leaves being the most targeted products. The intree model has two critical limitations, though. First, all paths associated with customer preference lists have the same directionality, implying that all customers substitute in the same fashion. Second, all preference lists include the root product implying, in our example, that all customers are willing to substitute to the most general product type. This eliminates any differentiation in pickiness within the customer populations.

In the outree model all customer classes are associated with paths that begin at the root node and terminate at an interior or leaf node. The outree model is appropriate when customers substitute from products with many features to products with less robust feature sets. This is the case, for example, in a product line that is targeted to a wide range of consumer budgets, with more expensive products having richer feature sets. Expensive products with many features would be located towards the root of of the tree, with the most feature rich product at the root node. Less expensive products with less rich feature sets are located toward the leaves of the tree, with the least feature rich located at the leaves. The outree model has similar limitations to the intree model. All paths associated with customer preference lists have the same directionality, implying that all customers substitute in the same fashion. Additionally, all preference lists include the root product as the first choice implying, in our example, that all customers prefer the most expensive feature rich product to all others. This eliminates the notion of budget conscious customers.

The tree model we consider generalizes the intree and outree models and, in doing so, eliminates the shortcomings of both. Specifically, by allowing preference lists to be associated with arbitrary

linear paths we make no implicit assumptions about the substitution behavior of the customer population beyond the underlying tree structure.

Additionally, our more general tree model allows us to capture the closely related Markov chain choice model. This model was introduced in [**?**] and, in its general form, was shown to closely approximate the nonparametric model. The Markov chain model consists of an initial probability vector and a transition matrix. Given an assortment of offered products a customer is modeled as sampling from the initial probability vector to select an initial, most preferred product. If the initial product is offered they purchase it and, otherwise, if that product is not offered, they transition according the the transition matrix in a Markovian way until they either transition to an offered product or transition out of the system. We introduce this model formally in Section **??**.

When the transition matrix of the Markov chain choice model has a tree-like topology we can reduce it to our nonparametric tree model. The reduction critically depends on our more general tree model; a reduction is not possible to either the intree or outree models presented in [**?**]. This reduction provides a further link between the nonparametric and Markov chain choice models. It also allows us to provide, to our knowledge, the first polynomial-time algorithms for constrained assortment optimization problems under a variant of the Markov chain choice model.

Contributions:

We consider fundamental revenue management problems under the nonparametric tree model. These problems fall into two main categories: assortment optimization problems and the network revenue management problem. In both cases, we develop algorithms that solve the respective problems. Below we summarize our results.

First, we consider assortment optimization problems. In the assortment optimization problem the retailer is presented with a collection of products to select from and must choose an assortment of products to offer to customers so as to maximize expected revenue. In this case we show that the assortment problem can be solved with a dynamic program. This dynamic program has a small state space and, consequently, leads to efficient algorithms. The key insight that we make in the dynamic program is that the purchase probability of any item within an arbitrary assortment can be computed recursively with only knowledge of each product's closest offered predecessor in the tree. Because our nonparametric tree model is a generalization of the intree and outree models in [**?**], this dynamic program provides a generalization of their methods.

The dynamic program for the pure assortment problem can be extended to settings where the retailer has additional cost considerations. We consider scenarios where there are fixed costs to include products in the offered assortment and penalties when a customer is forced to substitute to a less preferred product. Substitution penalties model a loss of customer good will, a common consideration for retailers.

Additionally, we extend the dynamic program to the cardinality constrained assortment op-

timization problem. In this problem the available products are grouped into categories and the retailer can offer a limited number of products from each category. In the simplest case all products are a single category and the retailer is constrained to have an assortment of limited size. We show that our dynamic program can be extended to solve the cardinality constrained problem.

The dynamic program used in the cardinality constrained assortment problem provides a framework to solve the related, but more difficult, capacity constrained assortment optimization problem. In this problem each product consumes some arbitrary capacity, which is a limited resource for the retailers. Capacity can represent shelf space, space on delivery trucks, or production capacity. Again, the retailers objective is to offer an assortment that maximizes expected revenue subject to a limit on the total capacity consumption of the offered products. We show that this problem is NP-hard and, consequently, there is no tractable algorithm unless P=NP. However, we develop a fully polynomial time approximation scheme (FPTAS). The FPTAS allows a trade off between running time and solution quality: it can expend additional computational time to produce assortments guaranteed to provide revenue progressively closer to optimal. With enough computational power these algorithms can get arbitrarily close to an optimal solution.

The second problem we consider is known as the network revenue management problem. To our knowledge we are the first to consider the network revenue management problem under a variant of the nonparametric choice model. In this problem setting there are a number of resources and a collection of products, each of which consumes some combination of resources. There is a selling horizon discretized into time periods. In each time period the retailer must asses the available resources and decide which products to offer. Once the retailer offers an assortment of products, a customer arrives and makes a purchasing decision. Again, we model the customer's purchase decision with the nonparametric tree model. The objective is to find a policy to decide which products to offer at each time period so as to maximize the total expected revenue.

We can formulate the network revenue management problem as a dynamic program but this leads to a large state space and is not practical. Instead, we focus on a deterministic linear programming approximation formulated under the assumption that the customer choices take on their expected values. In this linear program there is a decision variable for each subset of products and, consequently, the number of decision variables increases exponentially with the number of products; this leads to an intractable linear program for large numbers of products. Focusing on the deterministic linear program, we show that if the customers choose according to the nonparametric tree model, then the deterministic linear program can immediately be reduced to an equivalent linear program whose number of decision variables and constraints increase only polynomially with the numbers of products and resources. We develop an algorithm to recover the optimal solution to the original deterministic linear program by using the optimal solution to the reduced deterministic linear program. This algorithm allows us to recover the frequency with which we should offer each subset of products to customers. The insight behind this algorithm is twofold. First, the unconstrained assortment optimization problem can be formulated as a concise linear

problem whose feasible region contains that of the reduced linear program. Second, any feasible solution to this linear program induces a distribution over the assortments that is identical to the optimal frequencies with which to offer each subset of products. This algorithm is efficient and easily implemented in practice.

Related Literature:

There are a handful of papers that have considered the assortment optimization problem under the nonparametric choice model. The work that is most closely related to ours is [?], who consider the assortment problem restricted to intrees and outrees. Both of these models have restrictions on which preference lists can be associated with customer classes. We extend the results of this paper by lifting many of these restrictions and working in a more general setting. In [?] they also introduce operational considerations, such as fixed costs for introducing products or penalties when a customer substitutes to a less preferred product. We extend these results by considering more general cost functions and introducing the cardinality and capacity constrained versions of the assortment problem. Our dynamic programming approach closely resembles the approach proposed in [?] for the unconstrained assortment problem under the Markov chain choice model. Both techniques consider how a particular offer decision can "block" demand that would otherwise find the set of products that are already offered.

Two other papers that are closely related to our work are [?] and [?]. The former proves various hardness results related to the assortment problem under the nonparametric choice model. The latter considers the assortment optimization problem under the nonparametric choice model when customer preference lists are associated with structured set systems defined over a single overarching ordering of the products; one such structured set system is a laminar family, for example. The general algorithm provided in this paper can be used to solve the outree case described in [?], but it does not generalize the more complex intree case. The authors of [?] study a consider-then-choose nonparametric choice model. In this model, each customer class is distinguished by a price threshold and a preference list. Once prices have been set, customers purchase the highest ranking offered product that is priced below their price threshold. The authors of this paper study the joint assortment and pricing problem under this two stage buying model.

There is literature on assortment optimization problems under various choice models. [?], [?], [?] and [?] study various versions of the constrained assortment problems when customers choose according to the multinomial logit model. [?], [?] and [?] focus on assortment problems when customer choices are governed by a mixture of multinomial logit models. [?], [?], and [?] develop efficient solution methods for the unconstrained assortment problem when customers choose under the nested logit model. The authors of [?] and [?] study the space and cardinality constrained versions of the assortment problem when customers choose according to the nested logit model. [?] and [?] study related pricing problems under the nested logit model.

It is common to formulate deterministic linear programming approximations for network rev-

enue management problems under the assumption that customer choices take on their expected values. Such approximations appear in [?], [?], [?], [?], [?], [?] and [?]. The authors of [?], [?] and [?] provide tractable methods to approximate the dynamic programming formulations of network revenue management problems. Our work on the network revenue management problem most closely resembles that of [?] who also show that the deterministic linear program can be condensed to a reduced linear program when customers choose according to the Markov chain choice model. In this paper, recovering the optimal solution to the original deterministic linear program from the optimal solution to the reduced linear program requires a tedious projection algorithm. In contrast, our approach for recovering the optimal solution to the deterministic linear program is a simple sampling algorithm.

The remainder of this paper is organized as follows. In Section 2, we describe the nonparametric tree choice model and introduce the assortment optimization problems that we study. In Section 3, we give our dynamic programming approach to solve the unconstrained assortment problem and show how this approach can be extended to consider assortment problems with cost considerations (Section 3.1 as well as the cardinality constrained version of the assortment problem (Section 3.2). Additionally, in Section 5 we show how to reduce a special case of the Markov chain choice model to the nonparametric tree model. In Section 3.2, we give an FPTAS for the space constrained assortment problem which we show to be NP-Hard. Next, we present our results for the network revenue management problem in Section 5. We present computational experiments in Section 6 to validate the efficiency of our dynamic programming approach for the unconstrained assortment problem. In Section 7, we conclude and provide avenues for future work.

2 Nonparametric Tree Choice Model

We will use a nonparametric, ranking-based choice model to model customer buying decisions. A retailer has access to a collection of n substitutable products indexed by $N = \{1, \ldots, n\}$. There is a collection of customer classes $\mathcal{G}$, where each customer class $g \in \mathcal{G}$ is defined by an arrival probability λ_g and a product preference list σ_g defined over a subset of N. The list σ_g represents a customer's product preferences and we let $\sigma_g(i)$ be the position of product i in customer class g's preference list; note that σ_g may not include all products in N. If the retailer offers assortment $S \subseteq N$ and the list σ_g contains an element of S then customer type g will purchase product $\pi_g(S) := \arg\min_{i \in S} \sigma_g(i)$. If σ_g does not contain an element of S then customer type g does not make a purchase; in this case we will abuse notation and let $\pi_g(S) = 0$.

Given the collection of customer classes $\mathcal{G}$ and offer set S the probability that item i is purchased is

$$\Pr_i(S) = \sum_{g \in \mathcal{G}: \pi_g(S) = i} \lambda_g.$$

For every $j \in N$ we let $r_j > 0$ denote the profit margin of product j. When set S is offered, the

expected revenue is

$$R(S) = \sum_{i \in S} \Pr_i(S) r_i.$$

Our objective is to find a set $S^* \subseteq N$ that maximizes expected revenue

$$R^* = \max_S R(S). \tag{1}$$

The authors in [**?**] show that problem (1) is NP-Hard to approximate within a factor of $O(n^{1-\epsilon})$ for any $\epsilon > 0$. Further, the hardness result of [**?**] holds even when the preference lists for each customer type are constructed from a single over-arching ordering, i.e. there exists an ordering $\prec$ on the products where $\sigma_g(i) < \sigma_g(j)$ implies $i \prec j$ for all $g \in \mathcal{G}$. As a result, a natural next step is to simplify the space of potential customer types in order to render the assortment problem tractable while not making too large a sacrifice in terms of modeling flexibility with regards to the underlying choice process.

We will be interested in customer classes based on a rooted undirected tree structure $T = (N, E)$ with root $\bar{r}$ and all nodes, including $\bar{r}$ are the products in N. Any customer class $g \in \mathcal{G}$ will have a product preference list σ_g associated with a path in T. The ordering of the products in σ_g will correspond to the order products are visited in a path through T. We restrict our attention to *linear* paths, paths that visit at most one child of every node. See Figure 1. We can think of these paths as either moving towards or away from $\bar{r}$. When no confusion arises we identify σ_g with the path in the tree and refer the to preference list moving towards or away from $\bar{r}$. In what follows, we will assume the tree T is a binary tree. This assumption is without loss of generality: we can meet this requirement by adding at most n nodes that represent null products that provide no cost or benefit to the retailer.

In addition to this unconstrained problem, we will also be interested in adding constraints. For example, the retailer may be constrained by limited shelf space and can only offer at most k items. In this case the retailer can only offer an assortment $S \in \mathcal{F} = \{S \in N : |S| \leq k\}$. We refer to $\mathcal{F}$ as the collection of feasible assortments. The retailer is then faced with the problem

$$\max_{S \in \mathcal{F}} R(S). \tag{2}$$

In what follows, we will be interested in different constraints, each of which leads to a different collection of feasible sets $\mathcal{F}$. These additional constraints can complicate the assortment problem considerably.

3 Main Algorithm

In this section, we provide a dynamic program for the assortment problem given in (1). The tree T will define the steps of computation in our dynamic program. Before stating the dynamic program we first introduce additional notation and develop specific insights into solving (1) in a tree T.

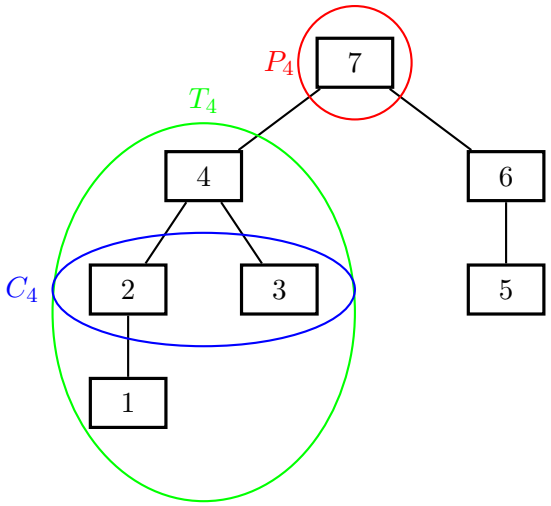

Figure 1: An example of a set of customer classes represented as a rooted binary tree. Possible customer preference lists are all linear paths including $(7, 4, 2, 1)$, $(3, 4)$, and (5). The path $(1, 2, 4, 3)$ is not linear and would not correspond to a possible customer class.

Given a vertex i, we let C_i be the children of i in T. Further, we say that i is the parent of all $j \in C_i$ and define $P_j = i$. Note that for leaves of T, $C_i = \emptyset$. We will be interested in complete subtrees of T. We let T_i be the subtree rooted at i containing all successors of i. When there is no confusion we will also use T_i to refer to the products represented by the nodes of the complete subtree. Without loss of generality, we can index the nodes such that the root node has index n and if $T_i \subset T_j$ then $j > i$.

The tree T will be used to define *blocking* relationships among products. For a customer class g we say i blocks product j when S is is offered if $\pi_g(S) = i$ and $\pi_g(S/i) = j$. More generally, we say i blocks j when S is offered whenever there exists at least one class g where this blocking relationship holds. Intuitively, i blocks j when the removal of i from the offer set induces a customer class to purchase product j. Since T defines the ordered lists for customer classes these blocking relationships are tied to T. We define, for any pair of nodes, the degree to which they block each other. Specifically, we let

$$B_{i,j} = \sum_{g \in G: \pi_g(\{i,j\})=i, \pi_g(\{j\})=j} \lambda_g.$$

Note that $B_{i,j}$ is not identical to $B_{j,i}$ since these two terms involve customer classes moving in opposing directions, which may have different associated probabilities. In addition to describing blocking in terms of probability we will also describe blocking in terms of revenue. We let $r_j B_{i,j}$ be the revenue i blocks from j when $\{i, j\}$ is offered.

Given a subset S and $i \in S$, we define $\phi_i(S)$ to be i's closest predecessor in S and $\delta_i(S) = \{j \in S | \phi_j(S) = i\}$ to be the set of closest successors to i in S. If no predecessor of i is offered in S , let $\phi_i(S) = 0$; if no successors are offered we let $\delta_i(S) = \emptyset$. Further, we use $\Phi(i)$ to represent all of i's predecessors in T including product 0.

If we offer subset S and $i \in S$, any customer class g traveling away from the root that has i in its ranked subset does not end up purchasing i if and only if it purchases a successor j of i. Since all customer classes are linear, this customer class must also contain $\phi_i(S)$ in σ_g before i. Therefore, we know σ_g contains both $\phi_i(S)$ and i but prefers $\phi_i(S)$ to i. Similarly, a customer class traveling up the tree towards the root that has i in its ranked subset does not purchase i if and only if it purchases a successor of i. Since all customer classes are linear, σ_g must contain both a node $j \in \delta_i(S)$ and i but prefers j to i. These considerations allow us to rewrite the probability i is purchased when S is offered using our blocking notation:

$$\Pr_i(S) = \begin{cases} \Pr_i(i) - B_{\phi_i(S),i} - \sum_{j \in \delta_i(S)} B_{j,i} & i \in S \\ 0 & i \notin S \end{cases}. \tag{3}$$

This alternative expression is critical in our dynamic program formulation. Note that this probability does not change if i's closest offered predecessor and closest offered successors in S remain the same. In essence, purchase decisions related to i are local decisions.

Our dynamic program is based on maximizing *adjusted revenues* in complete subtrees of T. Intuitively, given T_i and a fixed set $\bar{S}_i \subseteq N/T_i$ the adjusted revenue of an offer set $S_i \subseteq T_i$ is the revenue received from products in S_i minus the revenue S_i blocks from products in $\bar{S}_i$. However, as described above, to calculate the purchase probabilities for each node $j \in S_i$ we only need to know the closest offered predecessor p of the root i of $\bar{S}_i$. Similarly, we only need to know p to know which customer classes are blocked by S_i from considering nodes in $\bar{S}_i$.

More precisely, given a subset $S_i \subseteq T_i$ and a predecessor p of i, we define the adjusted revenue of S_i to be

$$A(S_i, p) = \sum_{j \in S_i} r_j \Pr_j(S_i \cup \{p\}) - r_p \sum_{k \in \delta_p(S_i \cup \{p\})} B_{k,p}.$$

Note that this expression also holds when $p = 0$. The first term is the revenue received from products in S_i when p is the closest predecessor offered to i. This follows from the fact that calculating the purchase probability of an item only requires knowing the closest offered predecessor and closest offered successors (all of which are in S_i). The second term accounts for the revenue S_i blocks from p (and therefore from $\bar{S}_i$). With this new notation we can rewrite (1) as

$$R^* = \max_{S \subseteq N} A(S, 0).$$

The terminal state of our dynamic program will compute precisely this maximum.

We can now present our dynamic program. Each stage is a product i under consideration for inclusion in S and the one dimensional state space is a product p, possibly equal to 0, that is a predecessor of i in T. Our value function $V_i(p)$ is the maximum adjusted revenue that can be achieved from subsets of T_i when p is the closest offered predecessor of i.

$$V_i(p) = \max\{r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} + \sum_{k \in C_i} V_k(i), \sum_{k \in C_i} V_k(p)\}. \tag{4}$$

For leaves of T, our base case, this simplifies to $V_i(p) = \max\{r_i \Pr_i(i) - r_p B_{i,p} - r_i B_{p,i}, 0\}$.

Theorem 3.1.

$$V_i(p) = \max_{S_i \subseteq T_i} \{A(S_i, p)\}.$$

Proof. First, consider the base case. For the leaves of T, $V_i(p) = \max\{r_i \Pr_i(i) - r_p B_{i,p} - r_i B_{p,i}, 0\}$. This first term is equivalent to $A(\{i\}, p)$ and the second term to $A(\emptyset, p)$ so the claim holds.

Now consider a node i that is not a leaf and suppose that the claim holds for all successors of i. Let l and r be the left and right children of i, respectively. Let $S_i^* \subseteq T_i$ be a subset that maximizes $A(S_i, p)$.

In the first case, $i \in S_i^*$. Then, $\delta_p(S_i^* \cup \{p\}) = \{i\}$ since it is the only node in S_i^* for which p is the closest offered predecessor, i.e. $\phi_i(S_i^* \cup \{p\}) = p$. We have

$$\begin{aligned}
A(S_i^*, p) &= \sum_{j \in S_i^*} r_j \Pr_j(S_i^* \cup \{p\}) - r_p \sum_{k \in \delta_p(S_i^* \cup \{p\})} B_{k,p} \\
&= \sum_{j \in S_i^*} r_j \left(\Pr_j(j) - B_{\phi_j(S_i^* \cup \{p\}), j} - \sum_{k \in \delta_j(S_i^* \cup \{p\})} B_{k,j} \right) - r_p B_{i,p} \\
&= r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} + \sum_{j \in S_i^* - \{i\}} r_j \Pr_j(S_i^* \cup \{p\}) - r_i \sum_{k \in \delta_i(S_i^* \cup \{p\})} B_{k,i} \\
&= r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} + \sum_{j \in S_i^* - \{i\}} r_j \Pr_j(S_i^*) - r_i \sum_{k \in \delta_i(S_i^*)} B_{k,i}
\end{aligned}$$

where the second line follows from (3) and the last line comes from the fact that for each $j \in S_i^* - \{i\}$, the closest predecessor of j is not p (since i is offered) and $\Pr_j(S_i^* \cup \{p\}) = \Pr_j(S_i^*)$.

We can simplify this expression further. The set S_i^* can be decomposed into $\{i\}$ and two additional sets: $S_l^* = S_i^* \cap T_l$ and $S_r^* = S_i^* \cap T_r$. Let $j \in S_i^* - \{i\}$. Without loss of generality, let $j \in T_l$. Then, $\phi_j(S_i^*) = \phi_j(S_l^* \cup \{i\})$ since its closest predecessor must be i or in the same subtree as j. This also shows $\delta_j(S_i^*) = \delta_j(S_l^* \cup \{i\})$ and

$$\Pr_j(S_i^*) = \Pr_j(S_l^* \cup \{i\}).$$

Lastly, we can easily see that

$$\delta_i(S_i^*) = \delta_i(S_l^* \cup \{i\}) \cup \delta_i(S_r^* \cup \{i\}).$$

Continuing from the above expression, these observations allow us to write

$$\begin{aligned}
A(S_i^*, p) = r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} \\
&+ \sum_{j \in S_l^*} r_j \Pr_j(S_l^* \cup \{i\}) - r_i \sum_{k \in \delta_i(S_l^* \cup \{i\})} B_{k,i} \\
&+ \sum_{j \in S_r^*} r_j \Pr_j(S_l^* \cup \{i\}) - r_i \sum_{k \in \delta_i(S_r^* \cup \{i\})} B_{k,i} \\
&= r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} + A(S_l^*, i) + A(S_r^*, i).
\end{aligned}$$

Noting that S_i^* is the maximizer of the above expression and that $A(S_l^*, i)$ and $A(S_l^*, i)$ are completely independent since they do not share any successors in the tree, we see that we can express our optimization problem recursively:

$$\begin{aligned}
\max_{S_i \subseteq T_i : i \in S_i} A(S_i, p) &= r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} + \max_{S_l \subseteq T_l} A(S_l, i) + \max_{S^r \subseteq T_r} A(S_r, i) \\
&= r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} + V_l(i) + V_r(i)
\end{aligned}$$

where we have used the inductive hypothesis.

By very similar analysis, when $i \notin S_i^*$ we get

$$\begin{aligned}
\max_{S_i \subseteq T_i : i \notin S_i} A(S_i, p) &= \max_{S_l \subseteq T_l} A(S_l, p) + \max_{S_r \subseteq T_r} A(S_r, p) \\
&= V_l(p) + V_r(p).
\end{aligned}$$

By combining these expressions we reach the desired claim

$$\begin{aligned}
\max_{S_i \subseteq T_i} A(S_i, p) &= \max\{ \max_{S_i \subseteq T_i : i \notin S_i} A(S_i, p), \max_{S_i \subseteq T_i : i \in S_i} A(S_i, p) \} \\
&= \max\{ r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} + \sum_{k \in C_i} V_k(i), \sum_{k \in C_i} V_k(p) \}.
\end{aligned}$$

□

The special case of Theorem 3.1 when $i = \bar{r}$ and $p = 0$ shows that our dynamic program computes the optimal solution to (1), R^*. We recover the optimal subset corresponding to this revenue using Algorithm 2 in the Appendix.

We can now analyze the computational complexity of computing the necessary $V_i(j)$. Let d be the depth of T. We pre-compute each $\Pr_i$ and $B_{i,j}$. The number of customer classes is $|\mathcal{G}| \leq nd$. Each $g \in \mathcal{G}$ contributes to at most d^2 $B_{i,j}$ values since $|\sigma_g| \leq d$ and we are interested in pairs of nodes in σ_g. Each customer class also contributes to at most d $\Pr_i$ values. Therefore, calculating the $\Pr_i$ and $B_{i,j}$ values has running time $O(nd^3)$. After this pre-computation, each of the $O(nd)$ values $V_i(j)$ can be computed in constant time. This leads to an overall running time of $O(nd^3)$. For a full binary tree, $d = \log n$ leading to a running time of $O(n \log^3 n)$.

We present one final corollary regarding the adjusted revenues. Specifically, the corollary below shows that the revenue of any assortment can be computed by summing adjusted revenues. This corollary becomes useful when we consider extending (4) in subsequent sections.

Corollary 3.2. *Consider any subset $S \subseteq N$ and node i with children l and r. Let $S_i = T_i \cap S$, $S_l = T_l \cap S$, and $S_r = T_r \cap S$. Lastly, let $p = \phi_i(S)$. Then,*

$$A(S_i, p) = \begin{cases} A(i,p) + A(S_l, i) + A(S_r, i) & i \in S, \\ A(S_l, p) + A(S_r, p) & \textit{otherwise.} \end{cases}$$

In particular, this shows that

$$A(S, 0) = \sum_{i \in S} A(i, \phi_i(S)) = R(S).$$

Proof. This follows from the recursive analysis in Proposition 3.1. □

3.1 Product and Substitution Costs

The dynamic program in (4) can be easily extended to incorporate additional cost considerations. In this section we focus on two considerations proposed in [**?**]: a setup cost incurred when offering a product and a substitution penalty incurred when a customer is forced to substitute to less desirable products. To model setup costs we introduce a constant fixed cost of k_i for offering product i, which could represent a set up or stocking cost. To model the substitution penalty we introduce a function $f(l)$ that represents the penalty incurred when a customer purchases their l^{th} most preferred product. If a customer type g purchases product i and $l = \sigma_g(i)$ then the retailer incurs a penalty of $f(l)$. In [**?**] they assume that $f(\cdot)$ is linear and increasing and that $f(1) = 0$; we consider arbitrary functions. Below we present an extension of (4) that includes these costs. This provides a polynomial time algorithm for assortment optimization under these cost considerations: this is an improvement over the exponential time algorithm of [**?**].

We let

$$\text{Pen}_i = \sum_{g \in G : i \in \sigma_g} \lambda_g f(\sigma_g(i)).$$

be the sum of penalties the retailer incurs if set $S = \{i\}$ is offered. When offering i prevents a customer from substituting further down in their list, it can potentially lower the total penalty. This inspires a notion of "blocking" similar to that we introduced in the previous section. Given any pair of nodes, we let

$$Q_{i,j} = \sum_{g \in G : \pi_g(\{i,j\}) = i, \pi_g(\{j\}) = j} \lambda_g f(\sigma_g(j)).$$

be the penalty i blocks from j.

We can now write the modified dynamic program:

$$V_i(p) = \max\{r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} - k_i - \text{Pen}_i + Q_{i,p} + Q_{p,i} + \sum_{k \in C_i} V_k(i), \sum_{k \in C_i} V_k(p)\}. \quad (5)$$

For leaves of T, our base case, this simplifies to $V_i(p) = \max\{r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} - k_i - \text{Pen}_i + Q_{i,p} + Q_{p,i}, 0\}$. The value functions in (5) capture the adjusted revenue for product i and all of its successors given that product p is the closest offered successor of i. Here, $r_i \Pr_i(i) - r_i B_{p,i} - k_i - \text{Pen}_i + Q_{p,i}$ is the revenue received from i, modified to include costs and penalties, when p is the closest offered predecessor and other products in T_i are not offered. The term $-r_p B_{i,p} + Q_{i,p}$ adjusts both the revenue and the penalty term for p since some customers may choose i instead.

The addition of $Q_{i,j}$ and Pen_i only introduce a constant multiplier to the running time of the dynamic program: $Q_{i,j}$ and Pen_i $\Pr_i$ can be precomputed simultaneously with $B_{i,j}$ and $\Pr_i(i)$. As a consequence the running time of this modified algorithm is $O(nd^3)$ where d is the depth of T.

3.2 Cardinality Constraints

Realistically retailers are under many constraints and are not able to offer any arbitrary set of products to their customers. Frequently these constraints are in the form of cardinality constraints where there is some small number of limited resources available to the retailer and each item consumes some integer quantity of each resource. This can model shelf space constraints, limited shipping availability, or limited production capacity. In the simplest case, a single shelf space constraint, each item consumes a single unit of capacity and the retailer has $C \leq n$ units of capacity on their shelves.

Beyond the simple shelf space constraint, the dynamic program in (4) can be extended to handle a wide variety of cardinality constraints. We can introduce a vector of resources available to the retailer and each product can consume any integer quantity of these resources. The vector can represent, for example, the shelf space available in different areas of a physical store, page space available in different parts of a website, space on different trucks that move between warehouses, or production capacity in different facilities. There are three limiting aspects to the cardinality constraints we can model. Firstly, the vector of resources must be a small constant relative to the input. Secondly, the consumption of resources by products must be expressed as an integer. Finally, the total available units of any resources must be small relative to the input; specifically the units of available resources can be expressed as some polynomial function of the size of the input. These limitations still afford a great flexibility in modeling power.

For simplicity, in introducing the extension of (4) to include cardinality constraints we focus on the simplest cardinality constraint: a space constraint. The retailer has C units of space and each product consumes 1 unit of space. It is easy to extend these methods to the more general cardinality

constraints. Formally, we can write the space constrained assortment problem as follows:

$$\max_{S:|S|\leq C} R(S). \tag{6}$$

Adding this space constraint couples decisions across different branches of the tree, complicating the assortment problem at hand. This is the primary difficulty with introducing any cardinality constraint.

We present a dynamic programming approach to solve the space constrained version of the problem. We will have a three dimensional state space: a product i that is under consideration for inclusion in S, a product p (possibly equal to 0) that is a predecessor of i in T, and the remaining units of space for products in T_i. Our value function $V_i(p, c)$ will be the maximum adjusted revenue that can be achieved from subsets of T_i using at most c units of space when p is the closest offered predecessor of i.

$$\begin{aligned} V_i(p,c) = \max\{&r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} + \max_{c_l,c_r:c_l+c_r\leq c-1} V_l(i,c_l) + V_r(i,c_r), \\ &\max_{c_l,c_r:c_l+c_r\leq c} V_l(p,c_l) + V_r(p,c_r)\}. \end{aligned} \tag{7}$$

For leaves of T, our base case, this simplifies to

$$V_i(p,c) = \begin{cases} \max\{r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p}, 0\} & c > 0 \\ 0 & c = 0. \end{cases}$$

This inner maximization represents an optimal allocation of the remaining space to product i's left and right parents, which we represent as nodes l and r respectively for the remainder of this paper.

The value functions for the constrained problem resemble those of the unconstrained problem given in (4), although we add an additional element to the state space to ensure we output a feasible assortment. The preprocessing of the $B_{i,j}$ and $\Pr_i$ values remains identical and takes $O(nd^3)$ time. However, adding the additional state increases the number of necessary $V_i(j)$ values to $O(n^2 d)$ and the computation for each value to $O(n)$. Therefore, the overall runtime is $O(\max\{n^3 d, nd^3\}) = O(n^3 d)$ since $d \leq n$.

4 General Capacity Constraints

In this section we solve the assortment optimization problem given in (2) when the set of feasible assortments $\mathcal{F}$ is given by a knapsack constraint. Specifically, each product consumes c_i units of space and there are C units of space to allocate. In terms of (2) this leads to a set of feasible assortments $\mathcal{F}(C) = \{S \subseteq N : \sum_{i\in S} c_i \leq C\}$ and an optimal revenue

$$R^*_C = \max_{S\subseteq N:\sum_{i\in S} c_i \leq C} R(S). \tag{8}$$

This constraint does not fall into the framework presented in Section 3.2 and is considerably harder to solve. The distinction is that the space consumed by each product, c_i, no longer needs to

take integer values, as it does in Section 3.2, and the space to allocate, C, can be arbitrarily large. The difficulty of the space constrained assortment problem given in (8) is detailed in the following hardness result.

Theorem 4.1. *The space constrained assortment problem is NP-Hard even when each customer class is a singleton.*

Proof. The proof, which we leave to the Appendix B, uses a reduction from the knapsack problem. □

Since the assortment problem given in (8) is NP-Hard we cannot hope to develop a polynomial-time algorithm to solve the problem optimally unless $P = NP$. As a result we seek an approximation scheme, namely an fully polynomial time approximation scheme (FPTAS). The FPTAS that we detail below allows us to find for all ϵ an assortment whose revenue is within a factor of $1+\epsilon$ of the optimal revenue in a running time that grows polynomially in $1/\epsilon$. Our approximation scheme assumes that the $R^* \geq 1$; this is a reasonable assumption since the revenues, r_i, can be given in arbitrarily accurate units, say to the nearest cent, without a substantial increase in the number of bits required to encode the problem. Further, the problem remains NP-Hard even with this assumption.

In order to develop our FPTAS we will focus on the space items occupy, rather than revenues. Specifically, we will be interested in the *minimum* space required to achieve a particular adjusted revenue threshold in each subtree of T. We let $\mathcal{F}(C)$ be the set of feasible assortments when the retailer has capacity C and, expanding on this notation, we let $\mathcal{F}_i(C) = \{S_i \subseteq T_i : \sum_{i \in S_i} c_i \leq C\}$. We let

$$C_i(p, r) = \min_C \left\{ \max_{S_i \in \mathcal{F}_i(C)} A(S_i, p) \geq a \right\}.$$

If there is no value C that can achieve adjusted revenue r in T_i then we let $C_i(p, a) = \infty$. Note that if r is greater than the highest revenue of any single item, $r_{\max}$, then $C_i(p, a) = \infty$ since the maximum revenue achievable by any set is bounded by $r_{\max}$.

Before presenting our FPTAS we present an intractable method of computing R^*. This method provides the basis of our FPTAS. We compute R^* by finding the maximum a such that $C_{\bar{r}}(0, a) \leq C$. This approach requires finding $C_{\bar{r}}(0, a)$ for all $a \in [0, r_{\max}]$; we do this with the following dynamic program. Recall that $A(i, p) = r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p}$ is the adjusted revenue we receive from product i when p is i's closest offered predecessor.

$$V_i(p, a) = \min \begin{cases} c_i + \min\limits_{a_l, a_r : a_l + a_r \geq a - A(i,p)} V_l(i, r_l) + V_r(i, a_r) \\ \min\limits_{a_l, a_r : a_l + a_r \geq a} V_l(p, a_l) + V_r(p, a_r), \end{cases} \tag{9}$$

where l and r are the left and right children of i and the inner minimization is over $a_l, a_r \in [0, r_{max}]$.

Theorem 4.2.

$$V_i(p,a) = C_i(p,a)$$

Proof. Let $C_i(p,a) = C_i^*$ and let $S_i^* = \arg\max_{S_i \in \mathcal{F}_i(C)} A(S_i,p)$. Without loss of generality we can assume $A(S_i^*,p) = a$. Let l and r be the left and right child of i, respectively.

Suppose $i \in S_i^*$. From the proof of Corollary 3.2 we have

$$a = A(S_i^*,p) = A(i,p) + A(S_l,i) + A(S_r,i)$$

where $S_l = S_i^* \cap T_l$ and $S_r = S_i^* \cap T_r$. Each of S_l and S_r consume some capacity; we let $\sum_{j \in S_l} c_j = C_l^*$ and $\sum_{j \in S_r} c_j = C_r^*$. Note that $C_i^* = c_i + C_l^* + C_r^*$. Each set also achieves some adjusted revenue; we let $A(S_l,i) = a_l^*$ and $A(S_r,i) = a_r^*$. Since $S_i^* \subseteq T_i$ achieves adjusted revenue a with *minimum* capacity we get $C_l(i,a_l^*) = C_l^*$ and $C_r(i,a_r^*) = C_r^*$. Further, to minimize capacity and achieve adjusted revenue a in T_i when we must allocate a_l^* adjusted revenue to T_l and a_r^* adjusted revenue to T_r. From these insights we get

$$\begin{aligned} C_i(p,a) &= c_i + C_l^* + C_r^* \\ &= c_i + C_l(i,a_l^*) + C_r(i,a_r^*) \\ &= c_i + \min_{a_l,a_r : a_l + a_r = a - A(i,p)} \{C_l(i,a_l) + C_r(i,a_r)\} \end{aligned}$$

We can now use the inductive hypothesis to show that when $i \in S_i^*$ we have

$$C_i(p,r) = c_i + \min_{a_l,a_r : a_l + a_r = a - A(i,p)} \{V_l(i,a_l) + V_r(i,a_r)\}.$$

Through similar reasoning, when $i \notin S_i^*$ we get

$$C_i(p,r) = \min_{a_l,a_r : a_l + a_r = a} \{V_l(p,a_l) + V_r(p,a_r)\}.$$

Combining these gives us

$$\begin{aligned} C_i(p,a) &= \min \begin{cases} c_i + \min_{a_l,a_r : a_l + a_r = a - A(i,p)} \{V_l(i,a_l) + V_r(i,a_r)\}, \\ \min_{a_l,a_r : a_l + a_r = a} \{V_l(p,a_l) + V_r(p,a_r)\} \end{cases} \\ &= V_i(p,a) \end{aligned}$$

which leads to our desired conclusion. □

The dynamic program in (9) is intractable since the state variable a is continuous. We approximate this DP by placing a grid on this continuous state variable and only computing $V_i(p,a)$ for values of a that lie on the grid. This strategy ultimately leads to a fully polynomial time approximation scheme.

Recall that the state a only take on values in $[0, r_{max}]$. We will grid this state variable using a linear grid scheme with step size ϵ on the interval $[0, 1]$ and an exponential scheme on the interval $[1, r_{max}]$:

$$\text{Grid} = \{(l\epsilon), l = 0, \ldots, L_1\} \cup \{(1+\epsilon)^l, l = 0, \ldots, L_2\},$$

where $L_1 = \lfloor 1/\epsilon \rfloor$, $L_2 = \lceil \log(r_{max})/\log(1+\epsilon) \rceil = O(\log(r_{max})/\epsilon)$. The linear, ϵ-spaced grid on the interval $[0, 1]$ is critical. It generates a polynomial (in $1/\epsilon$) number of grid points and, since $R^*_C \geq 1$, rounding up in this interval can cost us at most ϵR^*_C. Consider the following dynamic program, designed to approximate (9): for product i, predecessor p of i, and $a \in \text{Grid}$, we define

$$\bar{V}_i(p, a) = \min \begin{cases} c_i + \min\limits_{a_l, a_r : a_l + a_r \geq a - A(i,p)} \left[\bar{V}^*_l(i, a_l) + \bar{V}_r(i, a_r)\right] \\ \min\limits_{a_l, a_r : a_l + a_r \geq a} \left[\bar{V}_l(p, a_l) + \bar{V}_r(p, a_r)\right], \end{cases} \tag{10}$$

where the inner minimization is over $a_l, a_r \in \text{Grid}$, and the base cases

$$\bar{V}_i(p, a) = \begin{cases} c_i & A(i,p) \geq a \text{ and } a > 0 \\ 0 & a = 0 \\ \infty & \text{otherwise} \end{cases} \tag{11}$$

when i has is a leaf node.

These value functions are similar to those in (9); the only distinction is that the state variable a and the variables in the inner maximization, a_l and a_r, are now restricted to take values in Grid. To compute an approximation to R^*_C we follow the same outline. First we compute $\bar{V}_{\bar{r}}(0, a)$ for all $a \in \mathcal{G}$, where $\bar{r}$ is the root of the tree. Then we output $\bar{R}_C = \max_a\{\bar{V}_{\bar{r}}(0, a) : \bar{V}_{\bar{r}}(0, a) \leq C\}$ as our approximation to the optimal revenue, R^*_C.

Let S^* be an optimal assortment for problem (8) and let $S^*_i = S^* \cap T_i \; \forall \; i \in N$. The following proposition relates the adjusted revenue of S^*_i to R^*_{cap}.

Proposition 4.3. $A(S^*_i, \phi_i(S^*)) \leq R^*_C \; \forall \; i \in N$

Proof. We observe that $A(S^*_i, \phi_i(S^*_i)) \leq \sum_{j \in S^*_i} r_j \text{Pr}_j(S^*) \leq R^*_C$. The first inequality holds since the right hand side does not subtract the revenue blocked by $\phi_i(S^*)$, a non-negative quantity, and the second inequality holds because the middle term is the expected revenue from a feasible assortment. □

Theorem 4.4. *Suppose S^* achieves the optimal revenue in (8). Let $S^*_i = S^* \cap T_i$ and $\phi_i(S^*) = p$. Then, for any $\epsilon > 0$ grid, we have*

$$\bar{V}_i(p, \lceil A(S^*_i, p) - \epsilon |T_i| R^*_C \rceil) \leq C_i(p, A(S^*_i, p)),$$

where $\lceil x \rceil$ represents rounding x up to the nearest grid point.

Proof. R_C^* using capacity $C_{\bar{r}}(0, R_C^*)$

We first prove the result for leaf node i. If $i \in S^*$ then we have $A(S_i^*, p) = A(i, p)$. In this case, $C_i(p, A(i,p)) = c_i$. On the other hand, we know that $\bar{V}_i(p, \lceil A(i,p) - \epsilon R_C^* \rceil) \leq \bar{V}_i(p, \lceil A(i,p)(1-\epsilon) \rceil) \leq \bar{V}_i(p, \lfloor A(i,p) \rfloor) \leq c_i$, where the first inequality follows by Proposition 4.3. Similarly, if $i \notin S^*$, then $A(S_i^*, p) = 0$ and $V_i^*(j, 0) = \bar{V}_i(j, \lceil 0 - \epsilon \rceil) = 0$, which holds trivially. Thus, the result holds for the leaf nodes.

For the remainder of the proof fix i as an interior node of the tree and let l and r be the left and right children of i, respectively. For induction, assume the result holds for l and r. We consider two cases.

Case 1: i is not in S^*. Then, $C_i(p, A(S_i^*, p)) = C_l(p, A(S_l^*, p)) + C_r(p, A(S_r^*, p))$

Let $a_l = \lceil A(S_l^*, p) - \epsilon |T_l| R_C^* \rceil$ and $a_r = \lceil A(S_r^*, p) - \epsilon |T_r| R_C^* \rceil$. Define $\hat{V} = \bar{V}_l(p, a_l) + \bar{V}_r(p, a_r)$. Our induction hypothesis immediately gives us that $\hat{V} \leq C_i(p, A(S_i^*, p))$. We complete the proof for this case by showing that $\bar{V}_i(p, \lceil A(S_i^*, p) - \epsilon |T_i| R_C^* \rceil) \leq \hat{V}$. To do so, we show that when solving (10) for $\bar{V}_i(p, \lceil A(S_i^*, p) - \epsilon |T_i| R_C^* \rceil)$ the inner minimization considers the pair a_l, a_r. If a_l, a_r is considered then we immediately get $\bar{V}_i(p, \lceil A(S_i^*, p) - \epsilon |T_i| R_C^* \rceil) \leq \hat{V}$, as desired. To show that a_l, a_r is considered we must show that they sum to at least $\lceil A(S_i^*, p) - \epsilon |T_i| R_C^* \rceil$. Consider the following inequalities.

$$\begin{aligned}
\lceil A(S_i^*, p) - \epsilon |T_i| R_C^* \rceil &\leq A(S_i^*, p) + \epsilon R_C^* - \epsilon |T_i| R_C^* \\
&= A(S_l^*, p) - \epsilon |T_l| R_C^* + A(S_r^*, p) - \epsilon |T_r| R_C^* \\
&\leq \lceil A(S_l^*, p) - \epsilon |T_l| R_C^* \rceil + \lceil A(S_r^*, p) - \epsilon |T_r| R_C^* \rceil.
\end{aligned}$$

The first inequality follows from Proposition 4.3: rounding up to a grid point adds at most ϵR_C^*. The first equality follows because $|T_i| - 1 = |T_r| + |T_l|$ and that $A(S_i^*, p) = A(S_l^*, p) + A(S_r^*, p)$. The last inequality follows from the definition of rounding. This completes the proof of the first case.

Case 2: $i \in S^*$. Then, $C_i(p, A(S_i^*, p)) = c_i + C_l(i, A(S_l^*, i)) + C_r(i, A(S_r^*, i))$

This case follows from similar reasoning. As before let $a_l = \lceil A(S_l^*, i) - \epsilon |T_l| R_C^* \rceil$ and $a_r = \lceil A(S_r^*, i) - \epsilon |T_r| R_C^* \rceil$. Define $\hat{V} = c_i + \bar{V}_l(i, \lceil A(S_l^*, i) - \epsilon |T_l| R_C^* \rceil) + \bar{V}_r(i, \lceil A(S_r^*, i) - \epsilon |T_r| R_C^* \rceil)$. We have $\hat{V} \leq C_i(p, A(S_i^*, p))$ since $\phi_l(S^*) = \phi_r(S^*) = i$. We complete the proof for this case by again showing that $a_l + a_r \geq \lceil A(S_i^*, p) - \epsilon |T_i| R_C^* \rceil - A(i, p)$ and, as a consequence, the pair a_l and a_r is considered in the inner minimization in (10).

$$\begin{aligned}
\lceil A(S_i^*, p) - \epsilon |T_i| R_C^* \rceil - A(i, p) &\leq A(S_i^*, p)) + \epsilon R_C^* - \epsilon |T_i| R_C^* - A(i, \phi_i(S^*)) \\
&= A(S_l^*, i) - \epsilon |T_l| R_C^* + A(S_r^*, i) - \epsilon |T_r| R_C^* \\
&\leq \lceil A(S_l^*, i) - \epsilon |T_l| R_C^* \rceil + \lceil A(S_r^*, i) - \epsilon |T_r| R_C^* \rceil.
\end{aligned}$$

The first inequality follows from Proposition 4.3: rounding up to a grid point adds at most ϵR_C^*. The first equality follows because $|T_i| - 1 = |T_r| + |T_l|$ and that $A(S_i^*, p) = A(S_l^*, i) + A(S_r^*, i) +$

$A(i, \phi_i(S^*))$. The last inequality follows from the definition of rounding. This completes the proof of the second case. □

Theorem 4.4 allows us to prove the performance guarantee of the FPTAS, which we summarize in the next theorem.

Theorem 4.5. *For any $\epsilon > 0$ grid, we have that $\bar{R}_C \geq R^*_C(1 - n\epsilon)$.*

Proof. From Theorem 4.4 we have $\bar{V}_{\bar{r}}(0, \lceil A(S^*, 0) - \epsilon|T|R^*_C \rceil) \leq C_{\bar{r}}(0, A(S^*, p)) \leq C$. We now observe

$$\begin{aligned}
\bar{R}_C &= \max_a \{\bar{V}_{\bar{r}}(0, a) : \bar{V}_{\bar{r}}(0, a) \leq C\} \\
&\geq \lceil A(S^*, 0) - \epsilon|T|R^*_C \rceil \\
&\geq R^*_C - \epsilon|T|R^*_C \\
&= R^*_C(1 - n\epsilon)
\end{aligned}$$

□

To complete our analysis we analyze the running time of the FPTAS. Calculating $\bar{R}_C$ involves computing the value functions $\bar{V}_i(p, a)$. There are a total of $O(nd(1 + \log(r_{max}))/\epsilon)$ value functions since there are $O(nd)$ total (i, p) pairs and a can take on $O((1 + \log(r_{max}))/\epsilon)$ values. For this same reason, we can solve the inner minimization by enumerating all splits in the revenue in time that is $O((1 + \log(r_{max})/\epsilon)$. As discussed previously, in order to achieve $R^*_C(1 - \epsilon')$ we must choose $\epsilon = \epsilon'/n$ which gives a total run time of $O(n^3 d(1 + \log(r_{max}))/\epsilon)^2)$ which is polynomial in $1/\epsilon'$. Algorithm 4 shows how to recover the assortment that achieves this revenue guarantee.

5 Network Revenue Management

In this section, we show that when customers choose according to the nonparametric tree model, there is a natural segue from the assortment optimization problems considered in the previous sections to dynamic assortment problems where a retailer must manage a collection of scarce resources over a pre-specified selling horizon. More specifically, we consider a well known version of such a dynamic assortment problem known as the network revenue management problem. In this setting, a retailer has access to a collection of products which each consume different amounts of various resources. During each time period of the selling horizon, the retailer must decide which products to make available for purchase subject to the remaining capacities of each resource. The retailer wants to offer selections that maximize expected revenue over the selling horizon. We assume that customer choice in each time period is governed by the nonparametric tree model. This problem setting captures the scenario where an airline must make offer decisions regarding a collection of itineraries, each comprised of a set of flight legs, over a selling horizon.

Let the set $M = \{1, \ldots, m\}$ index the set of resources and the set $N = \{1, \ldots, n\}$ index the set of products. Additionally, there is a no-purchase option product 0. The incidence matrix $A \in \{0,1\}^{m \times n}$ captures the product-resource relationships, where entry $a_{v,i} \in A$ is 1 if product i consumes resource v and 0 otherwise. A sale of product $i \in N$ generates revenue r_i but also uses up one unit of each resource in $\{v : a_{v,i} = 1\}$. Let $c_v \; \forall \; v \in M$ be the initial capacities of each resource. The selling horizon is discretized into T time periods indexed by $\{1, \ldots, T\}$. At the beginning of each time period, the airline must decide which set of itineraries to make available for purchase subject to the remaining capacities on each flight with the goal of maximizing expected revenue.

An exact solution method exists for this problem via a dynamic programing approach. Let x_{it} denote the remaining capacity on flight leg i at time t so that $x_t = \{x_{v,t} : v \in M\}$ gives the remaining capacities on each flight leg. We let $U(x_t) = \{j \in N : a_{v,j} x_{v,t} \geq 1 \; \forall \; v \in M\}$ represent the set of itineraries each of whose flight legs all have at least one remaining seat. At the beginning of each time period t, the airline thus must choose a set $S \subseteq U(x_t)$ to make available to inquiring customers with the extension of maximizing the expected revenue accrued over the time horizon. The following DP solves the problem optimally:

$$V_t(x_t) = \max_{S \subseteq U(x_t)} \sum_{j \in S} \Pr_j(S)[r_j + V_{t+1}(x_t - \sum_{v \in M} a_{v,j} e_v)] + (1 - \sum_{j \in S} \Pr_j(S)) V_{t+1}(x_t), \qquad (12)$$

where e_i is a $|M|$-dimensional unit vector with a one in its i^{th} component. The boundary conditions are $V_{T+1}(\cdot) = 0$ and $V_t(\vec{0}) = 0 \; \forall \; t$.

Our goal is to solve (12), however, computing the exact value function proves to be intractable for large problem instances since the size of the state space grows exponentially in the number of resources. As a result, in the remainder of this section, we consider a well-known deterministic approximation to the value functions that has been shown to work well in practice.

A popular approximation to the network revenue management problem assumes that the otherwise stochastic demands take on their expected values and are thus allowed to take on non-integer values. For this approach, we introduce decision variables $h(S)$ for all $S \subseteq N$ that represent the fraction of time that each subset of products is offered over the selling horizon. In this case, the expected revenue generated over the selling horizon can be represented as $T \sum_{S \subseteq N} \sum_{i \in S} r_i h(S) \Pr_i(S)$. Similarly, the expected resource consumption over the selling horizon for each resource $v \in M$ can be expressed as $T \sum_{S \subseteq N} \sum_{i \in S} h(S) a_{v,i} \Pr_i(S)$. The Choice-Based Deterministic Linear Program (CBLP) for the network revenue management problem is given below:

$$\begin{aligned} Z^{CBLP} = \max_{h(S) \geq 0} \; & T \sum_{S \subseteq N} \sum_{i \in S} r_i h(S) \Pr_i(S) \qquad \text{(CBLP)} \\ s.t. \; & T \sum_{S \subseteq N} \sum_{i \in S} h(S) a_{v,i} \Pr_i(S) \leq c_v \quad \forall \; v \in M \\ & \sum_{S} h(S) = 1. \end{aligned}$$

The CBLP, as stated, is generally solved using constraint generation, however this approach can become quite tedious since the number of decisions variables grows exponentially in the number of products. Next, we give a simple linear programming formulation for the unconstrained static problem given in (1), which allows us recast the CBLP as an equivalent Reduced Deterministic Linear Program (RDLP) where the number of decision variables and constraints is $O(n^2)$.

Consider the following Linear Program:

$$R_{LP} = \max_{(\alpha,\beta)} \sum_{i \in N} \sum_{p \in \Phi(i)} \alpha_{i,p} A(i,p) \tag{LP}$$

$$s.t.$$

$$\alpha_{n,0} + \beta_{n,0} = 1 \tag{13}$$

$$\alpha_{i,p} + \beta_{i,p} = \beta_{P_i,p} \quad \forall\, i \in \mathcal{N}, p \in \Phi(i) \setminus P_i \tag{14}$$

$$\alpha_{i,p} + \beta_{i,p} = \sum_{k \in \Phi(P_i)} \alpha_{P_i,k} \quad \forall\, i \in \mathcal{N}, p = P_i \tag{15}$$

$$0 \leq \alpha_{i,p}, \beta_{i,p} \leq 1 \tag{16}$$

The decision variables $\alpha_{i,p}$ can be interpreted as the proportion of subsets S that product i is offered and p is i's closest offered predecessor (product $p = 0$ indicates no predecessor is offered). Conversely, $\beta_{i,p}$ is the proportion of subsets in which product i is not offered and p is its closest offered predecessor. The term $A(i,p)$ in the objective is the adjusted revenue function defined in Section 3. Recall that $\Phi(i)$ is the set of all predecessors of product i, including the no-purchase option. Since in problem (1) we consider a one period assortment problem, one would expect there to exist an integer optimal solution which offers one assortment with certainty, and this turns out to be the case. As an intermediate step to proving the correctness of LP, we first show two special properties of its feasible region that will be useful when we come back to the revenue management setting.

Proposition 5.1. *Any feasible solution (α, β) to LP induces a distribution f over the assortment $S \subseteq N$ where*

$$f_S(\alpha,\beta) = \prod_{i \in N} \frac{\mathbb{1}_{i \in S} \alpha_{i,\phi_i(S)} + \mathbb{1}_{i \notin S} \beta_{i,\phi_i(S)}}{\alpha_{i,\phi_i(S)} + \beta_{i,\phi_i(S)}},$$

where we set $\cdot/0 = 0$.

Proof. Since $\alpha, \beta \geq 0$ in LP, we know that $f_S(\alpha,\beta) \geq 0$. All that is left to show is that $\sum_{S \subseteq N} f_S(\alpha,\beta) = 1$. For arbitrary assortment $S' \subseteq N$, define

$$f_S(\alpha,\beta,S') = \prod_{i \in N \setminus S'} \frac{\mathbb{1}_{i \in S} \alpha_{i,\phi_i(S)} + \mathbb{1}_{i \notin S} \beta_{i,\phi_i(S)}}{\alpha_{i,\phi_i(S)} + \beta_{i,\phi_i(S)}}.$$

Consider arbitrary leaf node q, and note that:

$$\begin{aligned}\sum_{S\subseteq N} f_S(\alpha,\beta) &= \sum_{S\subseteq N\setminus q} f_S(\alpha,\beta,N\setminus q)\left(\frac{\alpha_{q,\phi_i(S)}}{\alpha_{q,\phi_i(S)}+\beta_{q,\phi_i(S)}}+\frac{\beta_{q,\phi_i(S)}}{\alpha_{q,\phi_i(S)}+\beta_{q,\phi_i(S)}}\right)\\ &= \sum_{S\subseteq N\setminus q} f_S(\alpha,\beta),\end{aligned}$$

where the first equality follows since we sum over every assortment not considering product q and then we consider either adding product q to this assortment with the $\frac{\alpha_{q,\phi_i(S)}}{\alpha_{q,\phi_i(S)}+\beta_{q,\phi_i(S)}}$ term or not adding it with $\frac{\beta_{q,\phi_i(S)}}{\alpha_{q,\phi_i(S)}+\beta_{q,\phi_i(S)}}$. Applying this result recursively for $n-1$ iterations (choosing a leaf q on the remaining tree) gives:

$$\begin{aligned}\sum_{S\subseteq N} f_S(\alpha,\beta) &= \sum_{S\subseteq N\setminus(N\setminus n)} f_S(\alpha,\beta)\\ &= \alpha_{n,0}+\beta_{n,0}=1,\end{aligned}$$

as desired. □

The above proof utilizes the notion that one can interpret the variables $\alpha_{i,p}$ as the fraction of offered assortments in which $i\in S$ and $p=\phi_i(S)$ and so $\Pr(i\in S)=\sum_{p\in\Phi(i)}\alpha_{i,p}$. Conversely, $\beta_{i,p}$ can be interpreted as the fraction of offered assortments where $i\notin S$ and $p=\phi_i(S)$, which means that $\Pr(i\notin S)=\sum_{p\in\Phi(i)}\beta_{i,p}$. Further, with this interpretation, it is not difficult to derive the conditional probability $\Pr(i\in S|\phi_i(S)=p)=\frac{\alpha_{i,p}}{\alpha_{i,p}+\beta_{i,p}}$, which are the individual terms in our expression for

$$f_S(\alpha,\beta)=\prod_{i\in S}\Pr(i\in S|\phi_i(S)=p)\prod_{i\notin S}\Pr(i\notin S|\phi_i(S)=p).$$

The following corollary uses these expressions to derive an alternate representation for the objective function of LP:

Corollary 5.2. *Let (α^*,β^*) be the optimal decision variables for* LP. *Then we have that $\alpha^*_{i,p}=\sum_{S:i\in S,p=\phi_i(S)} f_S(\alpha^*,\beta^*)$ and so the optimal objective of LP can be written $\sum_{S\subseteq N} f_S(\alpha^*,\beta^*)R(S)$*

Proof. We have that

$$\begin{aligned}\sum_{S:i\in S,p=\phi_i(S)} f_S(\alpha^*,\beta^*) &= \Pr(i\in S|p=\phi_i(S))P(p=\phi_i(S))\\ &= \Pr(i\in S|p=\phi_i(S))\Big(\prod_{k\in(T_p\cap\Phi(i))\setminus p}\Pr(k\notin S|p\in S)\Big)\Pr(p\in S)\\ &= \Big(\frac{\alpha_{i,p}}{\alpha_{i,p}+\beta_{i,p}}\Big)\Big(\prod_{k\in(T_p\cap\Phi(i))\setminus p}\frac{\beta_{k,p}}{\alpha_{k,p}+\beta_{k,p}}\Big)\Big(\sum_{r\in\Phi(p)}\alpha_{p,r}\Big)\\ &= \Big(\frac{\alpha_{i,p}}{\alpha_{i,p}+\beta_{i,p}}\Big)\Big(\frac{\beta_{P_i,p}}{\alpha_{P_i,p}+\beta_{P_i,p}}\Big)\ldots\Big(\frac{\beta_{\bar{C}_p,p}}{\alpha_{\bar{C}_p,p}+\beta_{\bar{C}_p,p}}\Big)\Big(\sum_{r\in\Phi(p)}\alpha_{p,r}\Big)=\alpha_{i,p},\end{aligned}$$

where $\bar{C}_p = \Phi(i) \cap C_p$ is the child of p in $\Phi(i)$. The first and second equality results by noting that the fraction of assortments under f where $i \in S$ and $\phi_i(S) = p$ are simply those that include products i and p and do not include any of the products in the path between these two products, which can be represented as $(T_p \cap \Phi(i)) \setminus p$. The final equality results due cancellation when the constraints of (13)-(15) are applied. Using this result, we get that $\sum_{i \in N} \sum_{p \in \Phi(i)} \alpha_{i,p} A(i,p) = \sum_{i \in N} \sum_{p \in \Phi(i)} A(i,p) \sum_{S: i \in S, p = \phi_i(S)} f_S(\alpha, \beta) = \sum_{S \subseteq N} f_S(\alpha, \beta) \sum_{i \in S} A(i, \phi_i(S)) = \sum_{S \subseteq N} f_S(\alpha, \beta) R(S)$, which completes the proof. The first equality follows from the alternate interpretation of the $\alpha_{i,p}$ given at the beginning of the corollary and the second equality results from flipping the order of summation. The final equality is a result of Corollary 3.2. □

The above Corollary proves useful in showing our next result, namely that the LP solves problem (1).

Theorem 5.3. *$R_{LP} = R^*$ for the unconstrained assortment optimization problem.*

Proof. We leave the proof to Appendix B. □

This result shows that in one optimal solution to LP is $\alpha^*_{i,p}, \beta^*_{i,j} \in \{0,1\}$ given in (17). We can recover the optimal assortment as follows $S^* = \{i \in N : \sum_{j \in \Phi(i)} \alpha^*_{i,p} = 1\}$. With this new formulation for problem (1) we are ready to formulate the RDLP:

$$
\begin{aligned}
Z^{RDLP} = \max_{(\alpha,\beta)} & T \sum_{i \in N} \sum_{j \in \Phi(i) \cup \{0\}} \alpha_{i,j} A(i,j) && \text{(RDLP)} \\
s.t. & \\
& \alpha_{n,0} + \beta_{n,0} = 1 \\
& \alpha_{i,p} + \beta_{i,p} = \beta_{P_i,p} \quad \forall\, i \in \mathcal{N}, p \in \Phi(i) \setminus C_i \\
& \alpha_{i,p} + \beta_{i,p} = \sum_{k \in \Phi(P_i)} \alpha_{P_i,k} \quad \forall\, i \in \mathcal{N}, p = P_i \\
& T \sum_{i \in N} a_{v,i} [\sum_{p \in \Phi(i)} \alpha_{i,p} (\mathrm{Pr}_i(i) - B_{p,i}) - \sum_{j \in T_i \setminus i} \alpha_{j,i} B_{j,i}] \leq c_v \quad \forall\, v \in M \\
& 0 \leq \alpha_{i,p}, \beta_{i,j} \leq 1
\end{aligned}
$$

The decision variables can be interpreted in the same way as they were for LP, namely, $\alpha_{i,p}$ can be interpreted as the probability that product i is offered when its closest offered predecessor is product j, and $\beta_{i,j}$ is the probability that i is not offered when j is its closest offered successor. The following theorem relates the CBLP to the RDLP:

Theorem 5.4. *The* RDLP *is equivalent to the* CBLP *in that they produce the same optimal objective function for any choice of customer classes and revenues $r_j \; \forall\, j \in N$.*

Proof. We leave the proof to Appendix B □

Theorem 5.4 reduces the CBLP to an equivalent linear program where the number of decision variables and constraints is $O(nd)$, where d is the depth of the tree. Though we can solve the RDLP efficiently for large product networks, the optimal solution does not immediately provide a policy that one could implement, as the weights $\{h(S) : S \subseteq N\}$ do. Fortunately, by Theorem 5.4, we can take the optimal solution (α^*, β^*) to the RDLP and then recover the optimal solution to the CBLP by setting $h(S) = f_S(\alpha^*, \beta^*)$ by Proposition 5.1. As a result, sampling from $h(S)$ requires merely sampling from $f_S(\alpha, \beta)$, which can be done in $O(n)$ using Algorithm 1 with the optimal solution to the RDLP as input.

Initialize: $\tilde{S} = \{0\}$;
for $i = n$ *to* 1 **do**
  if $U[0,1] < \alpha_{i,\phi_i(\tilde{S})}/(\alpha_{i,\phi_i(\tilde{S})} + \beta_{i,\phi_i(\tilde{S})})$ **then**
    $\tilde{S} = \tilde{S} \cup \{i\}$;
  end
end
return $\tilde{S}$

Algorithm 1: Sampling from $f_S(\alpha, \beta)$

6 Computational Experiments

In this section, we provide computation experiments which demonstrate the efficiency of the dynamic program presented in (5) to solve the costed assortment problem. We benchmark ourselves against the algorithm provided for intrees in [**?**]. This algorithm has a theoretical runtime that is exponential is the number of products, but has been been shown to work far better in practice. Since this algorithm is only valid when applied to problems where the least preferred product of all customer types is the root node, we restrict our computational experiments to cases of this nature.

6.1 Experimental Setup

In our computational experiments we generate a number of intree instances to test the efficacy of our dynamic program. For each instance, we solve the costed assortment problem using two different strategies. The first strategy utilizes the dynamic program given in (5), which we refer to as DP. The second approach uses the algorithm given in [**?**], which we refer to as ALG3 since it is labeled Algorithm 3 in this paper. Our goal is to compare the performance of DP and ALG3 by measuring the respective CPU seconds required to solve each instance of the assortment problem.

We generate each of the intree instances in the following manner. Each of the instances that we consider consists of customer classes derived from a complete binary tree. In other words, the total number of nodes or products in each intree is $n = 2^d - 1$ where we vary the number of levels in the tree $d \in \{3, 4, \ldots 8\}$. Since ALG3 is only valid when the least preferred product of each customer is

	DP		ALG3	
d	Avg Secs.	Max Secs.	Avg Secs.	Max Secs.
3	2.6×10^{-4}	3.2×10^{-4}	4.4×10^{-4}	$\times10^{-3}$
4	7.7×10^{-4}	8.9×10^{-4}	1.9×10^{-3}	0.017
5	2.1×10^{-3}	2.3×10^{-3}	0.011	0.089
6	5.4×10^{-3}	5.7×10^{-3}	1.00	30.78
7	0.014	0.015	9.92	262.5
8	0.033	0.035	NA	NA
9	0.081	0.084	NA	NA
10	0.19	0.20	NA	NA

Table 1: Comparing DP and ALG3 in terms of CPU seconds required to solve the costed assortment problem

the root node, we restrict the set of customer classes derived from each intree to be of this variety. For each instance, we consider all n customer types and assume that each type arrives with equal probability. So if the root node is given index n, we consider all customer types whose preference orderings take the form $\{i, \ldots, n\}\ \forall\ i \in N$. The revenues of each products are generated uniformly from the interval $[0, n]$. Once the revenues have been generated for a given problem instance, we then generate a fixed offer cost k for each product uniformly over the interval $[0, r_{min}]$, where r_{min} is the smallest randomly generated revenue for the given instance. In this way, we ensure that the cost of offering a product never exceeds the revenue gained from a sale of the product. We leave out substitution costs since they do not necessarily complicate the assortment problems we study and we see no obvious way of estimating such costs.

6.2 Computational Results

Table 1 summarizes our computational results. In all cases we used Python 2.7 on a Dell with an Intel Core i7-2600 Processor with 2.4 GHz and 8GB of RAM. The first column gives the number of levels in the intrees that we consider. For each value of d, we generate 100 unique intrees using the method described in the previous section. The second column gives the average CPU seconds required for DP to solve the 100 instances and the third column gives the maximum CPU seconds for DP over these 100 instances. Columns four and five give these same two statistics for ALG3.

The results in Table 1 indicate that DP significantly outperforms ALG3 in both average performance and worst case performance. Most notably, we confirm that DP does in fact scale polynomially with the number of nodes, while ALG3 appears to be on more of an exponential trajectory. Further, for DP, we observe that the maximum runtime is at most 25% larger than the average runtime over all values of d. On the contrary, when $d = 7$, the maximum runtime for ALG3 exceeded the average runtime by over 2500%. Since the maximum runtime appears to be growing exponentially with d, it was not possible to get a sense of how ALG3 performs on the bigger instances where $d > 7$. On the other hand, DP solves instances of the costed assortment problem with over 1000 products in fractions of a second.

7 Conclusion

In this paper, we began by studying assortment problems on trees. We give the first polynomial time algorithm to solve the general tree case for linear customer classes. We formulate this problem as a dynamic program where the offer decision for each product can be made by simply storing each node's closest predecessor. Through a series of computation experiments we show that our algorithm is more efficient than the algorithm proposed in [**?**] for randomly generated cases of intrees. We then generalize this dynamic program to allow us to solve assortment problems on general trees where arcs are allowed to travel in either direction. Further, we also give a dynamic programming formulation that can be solved in polynomial time for the cardinality constrained assortment problem, where there is a limit on the total number of products that can be included in the offered assortment, as well as for the costed case, where there are costs for offering products and for customers substituting down their respective preference list. The assortment problem for the space constrained version of the assortment problem is NP-Hard and so we give an FPTAS for this problem.

Next, we consider the network revenue management problem. First, we show that the unconstrained assortment problem can be solved through a simple linear program with $O(n^2)$ variables and constraints. We then use this linear program to show that if the customers choose according to the intree choice model, then the deterministic linear program can immediately be reduced to an equivalent linear program whose numbers of decision variables increases only quadratically with the numbers of products and linearly in the number of resources. We develop an algorithm to recover the optimal solution to the original deterministic linear program by using the optimal solution to the reduced linear program. This algorithm allows us to efficiently recover the frequency with which we should offer each subset of products to customers by using the optimal solution to the reduced linear program.

There are many direction for future research. First, it is unknown, at least to our knowledge, whether the assortment remains tractable when the tree structure of the graph is broken by allowing arcs to pass between levels. When this is the case, there can be multiple paths between nodes in the graph and thus the dynamic programming structure of (4) is no longer valid. The result of [**?**] shows that the problem is NP-Hard when arcs are allowed to cross levels arbitrarily, but it is unclear whether or not the assortment problem is tractable when arcs can only cross one level, for example. A second potential direction is considering the assortment problem on a constant number of intrees. Again, to our knowledge, it is unclear whether this problem admits an optimal polynomial time solution. In this case, a product can potentially have different closest predecessors in each tree, which renders our dynamic programming idea ineffective.

A Algorithms

Initialize: $S = \{0\}$;
for $i = n$ *to* 1 **do**
  if $A(i, \phi_i(S)) + \sum_{j \in C_i} V(j, i) > \sum_{j \in C_i} V(j, \phi_i(S))$ **then**
    $S = S \cup \{i\}$;
  end
end
return S

Algorithm 2: Finding the optimal assortment for the general tree case

Initialize: $\tilde{S} = \{\}$, ;
Dictionary which stores capacity allocated to each node;
Initialize: $D[n] = C$;
for $i = n$ *to* 1 **do**
  # Find i's current predecessor;
  $p = \phi_i(\tilde{S})$ #Find the optimal allocation of capacity to i's parents when i is offered;
  $(c_l^o, c_r^o) = \underset{c_l, c_r : c_l + c_r \leq D[i]-1}{\arg\max} V_l(i, c_l) + V_r(i, c_r)$;
  #Find the optimal allocation of capacity to i's parents when i is not offered ;
  $(c_l^n, c_r^n) = \underset{c_l, c_r : c_l + c_r \leq D[i]}{\arg\max} V_l(p, c_l) + V_r(p, c_r)$;
  if $r_i Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} + V_l(i, c_l^o) + V_r(i, c_r^o) > V_l(p, c_l^n) + V_r(p, c_r^n)$ **then**
    $\tilde{S} = \tilde{S} \cup \{i\}$;
    $D[l] = c_l^o$;
    $D[r] = c_r^o$;
  end
  else
    $D[l] = c_l^n$;
    $D[r] = c_r^n$;
  end
end
return $\tilde{S}$

Algorithm 3: Finding the optimal assortment for the capacity constrained intree case

Initialize: $\tilde{S} = \{\}$, ;
Dictionary which stores adjusted revenue to-go allocated to each node;
Initialize: $D[n] = r$;
for $i = n$ *to 1* **do**
    # Find i's current predecessor;
    $p = \phi_i(\tilde{S})$ #Find the optimal allocation of revenue to i's parents when i is offered ;
    $(a_l^o, a_r^o) = \min_{a_l + a_r \geq D[i] - A(i,p)} V_l(i, a_l) + V_r(i, a_r)$;
    #Find the optimal allocation of revenue to i's parents when i is not offered ;
    $(a_l^n, a_r^n) = \min_{a_l + a_r \geq D[i]} V_l(p, a_l) + V_r(p, a_r)$;
    if $c_i + V_l(i, a_l^o) + V_r(i, a_r^o) < V_l(p, a_l^n) + V_r(p, a_k^n)$ **then**
        $\tilde{S} = \tilde{S} \cup \{i\}$;
        $D[l] = a_l^o$;
        $D[r] = a_r^o$;
    end
    else
        $D[l] = r_l^n$;
        $D[r] = r_r^n$;
    end
end
return $\tilde{S}$

Algorithm 4: Finds the minimum weight assortment that achieves a revenue of r.

B Proofs

Proof of Theorem 4.1

Proof. We reduce the knapsack problem to the space constrained assortment problem given in (8). An instance of the knapsack problem is characterized by a set of item $N = \{1, \ldots, n\}$ each with a value v_i and a space consumption a_i. Also given as input is a total size of the knapsack C. The goal is to determine which items to include in a collection so that the total space consumption is less than or equal to C and the total value of the included items is maximized.

The reduction to problem (8) is very simple and works as follows. The set of customer classes will only be singletons and there will be a customer class for every item in the knapsack problem. The arrival probability for each customer class will be $1/n$. The revenue for each product will be nv_i and the space consumption will be a_i. The space limit on the offered assortment is C. As a result, when we offer a product, we accrue a revenue of exactly v_i and consume a_i of space. The reduction is complete, as this is exactly the knapsack problem. □

Proof of Theorem 5.3

Proof. Let S^* be the optimal assortment. By Corollary 5.2, we immediately have that $R^* \geq R_{LP}$ since R_{LP} can be viewed as the expected revenue of an assortment drawn from f. To conclude the proof, we show that $R_{LP} \geq R^*$ and in doing so we show how to recover the optimal assortment from the optimal solution to LP.

Given S^*, we show to construct a feasible solution to LP, which achieves an objective of R^*. Let

$$\begin{cases} \alpha^*_{i,p} = 1,\ \beta^*_{i,p} = 0 \text{ if } i \in S^*,\ \phi_i(S^*) = p \\ \alpha^*_{i,p} = 0,\ \beta^*_{i,p} = 1 \text{ if } i \notin S^*,\ \phi_i(S^*) = p \\ \alpha^*_{i,p} = \beta^*_{i,p} = 0 \text{ if } \ \phi_i(S^*) \neq p \end{cases} \tag{17}$$

We can write the objective value for this solution as

$$\sum_{i \in N} \sum_{j \in \Phi(i) \cup \{0\}} \alpha^*_{i,p} A(i,p) = \sum_{i \in S^*} \sum_{p \in \Phi(i): p = \phi_i(S^*)} \alpha^*_{i,p} A(i,p) = \sum_{i \in S^*} \alpha^*_{i,\phi_i(S^*)} A(i, \phi_i(S^*)) = R^*.$$

The final equality follows since $\alpha^*_{i,\phi_i(S^*)} = 1\ \forall\ i \in S^*$ by (17) and by Corollary 3.2. All that is left to show is that the $\alpha^*_{i,p}, \beta^*_{i,p}$ of (17) is feasible in (13)-(15). Clearly constraint (13) is satisfied because $\phi_n(S) = 0\ \forall\ S \subseteq N$. We consider the constraints (14) and (15) in cases, we start with the constraint in (14):

Case 1: $p = \phi_i(S^*)$

If $i \in S^*$, we have that $\alpha^*_{i,p} = 1$ and if $i \notin S^*$ we have that $\beta^*_{i,p} = 1$ and so the left hand side of constraint (14) reads $\alpha^*_{i,p} + \beta^*_{i,p} = 1$. Further since constraint (14) applies to $p \neq P_i$ we know that $P_i \notin S^*$, since otherwise $\phi_i(S^*) = P_i$. This means means that $\phi_{P_i}(S^*) = p$ and so $\beta^*_{P_i,p} = 1$ as desired.

Case 2: $p \neq \phi_i(S^*)$

The left hand side of constraint (14) reads $\alpha^*_{i,p} + \beta^*_{i,p} = 0$ independent of whether i is in S^* or not. Further, since $p \neq P_i$ we know that $\phi_{P_i}(S^*) \neq p$ so $\beta^*_{P_i,p} = 0$ as desired.

Next, consider the constraints of (15) in these same two cases:

Case 1: $p = \phi_i(S^*)$

The left hand side of this constraint reads $\alpha^*_{i,p} + \beta^*_{i,p} = 1$ since $\alpha^*_{i,p} = 1$ if $i \in S^*$ and $\beta^*_{i,p} = 1$ if $i \notin S^*$. Since $p = P_i$, then we have that $\sum_{p \in \Phi(P_i)} \alpha^*_{P_i,p} = \alpha^*_{P_i,\phi_{P_i}(S^*)} = 1$ since $P_i \in S^*$.

Case 2: $p \neq \phi_i(S^*)$

The left hand side of constraint (15) reads $\alpha^*_{i,p} + \beta^*_{i,p} = 0$ independent of whether $i \in S^*$. Further, since $p = P_i$ we know that $P_i \notin S^*$ and so $\alpha_{P_i,p} = 0 \; \forall \; p \in \Phi(P_i)$.

This shows that the solution given in (17) is feasible to LP and achieves objective R^*. Thus we have $R_{LP} \geq R^*$ and since we earlier proved that $R_{LP} \leq R^*$ we know that $R_{LP} = R^*$ as desired. $\square$

Proof of Theorem 5.4

Proof. Let (α^*, β^*) be the optimal solution to the RDLP. We show how to construct a feasible solution to CBLP that has objective Z^{RDLP}. To do so, set $\hat{h}(S) = f_S(\alpha^*, \beta^*)$ as defined by Proposition 5.1, which is possible since (α^*, β^*) is feasible to LP. By Corollary 5.2 we know that the objective of this solution in Z^{RDLP}. Further we know that $\sum_{S \subseteq N} \hat{h}(S) = 1$ since f is a probability distribution. All that is left to show is that the first constraint of CBLP is satisfied by $\hat{h}(S)$. Re-ordering the sums, we can rewrite this first constraint as $T \sum_{i \in S} a_{v,i} \sum_{S: i \in S} h(S) \text{Pr}_i(S) \leq c_v$. After plugging in the solution $\hat{h}(S)$ and the analytic expression for the purchase probabilities given

in (3), the left hand side of this constraint becomes :

$$
\begin{aligned}
&= T \sum_{i \in S} a_{v,i} [\sum_{S: i \in S} f_S(\alpha^*, \beta^*)(\mathrm{Pr}_i(i) - B_{\phi_i(S),i} - \sum_{j \in \delta_i(S)} B_{j,i})] \\
&= T \sum_{i \in S} a_{v,i} [\sum_{S: i \in S} f_S(\alpha^*, \beta^*)(\mathrm{Pr}_i(i) - B_{\phi_i(S),i}) - \sum_{S: i \in S} f_S(\alpha^*, \beta^*) \sum_{j \in \delta_i(S)} B_{j,i}] \\
&= T \sum_{i \in S} a_{v,i} [\sum_{p \in \Phi(i)} (\mathrm{Pr}_i(i) - B_{p,i}) \sum_{S: i \in S, \phi_i(S) = p} f_S(\alpha^*, \beta^*) - \sum_{j \in T_i \setminus i} B_{j,i} \sum_{S: j \in S, \phi_j(S) = i} f_S(\alpha^*, \beta^*)] \\
&= T \sum_{i \in S} a_{v,i} [\sum_{p \in \Phi(i)} \alpha^*_{i,p} (\mathrm{Pr}_i(i) - B_{p,i}) - \sum_{j \in T_i \setminus i} B_{j,i} \alpha^*_{j,i}] \leq c_v,
\end{aligned}
$$

as desired. This completes the proof that $\hat{S}$ is feasible and thus we have that $Z^{CBLP} \geq Z^{RDLP}$.

Next we show that $Z^{CBLP} \leq Z^{RDLP}$ by taking an optimal solution $h^*(S)$ to CBLP and constructing a feasible solution to RDLP with the same objective as follows:

$$
\begin{aligned}
\hat{\alpha}_{i,p} &= \sum_{S: i \in S, \phi_i(S) = p} h^*(S) \; \forall \; i, p \in N \\
\hat{\beta}_{i,p} &= \sum_{S: i \notin S, \phi_i(S) = p} h^*(S) \; \forall \; i, p \in N
\end{aligned} \tag{18}
$$

Using the proof of Corollary 5.2 with $f_S()$ replaced by $h^*(S)$, it is easy to see that the objective of this solution is Z^{RDLP}. All that is left is to show feasibility. We consider the constraints of RDLP one by one:

First Constraint: Plugging in our definition of $\hat{\alpha}_{i,p}, \hat{\beta}_{i,p}$, this first constraint reads $\sum_{S: n \in S} h(S) + \sum_{S: n \notin S} h(S) = \sum_{S \subseteq N} h(S) = 1$ as desired.

Second Constraint: This constraint becomes $\sum_{S: \phi_i(S) = p} h(S) = \sum_{S: P_i \notin S, \phi_{P_i}(S) = p} h(S) = \hat{\beta}_{P_i,p}$. The first equality follows since the second constraint assumes $p \neq P_i$ and this P_i is not offered.

Third Constraint: This constraint reads $\sum_{S: \phi_i(S) = p} h(S) = \sum_{p \in \Phi(P_i)} \sum_{S: P_i \in S, \phi_{P_i}(S) = p} h(S) = \sum_{p \in \Phi(P_i)} \hat{\alpha}_{P_i,p}$ as desired. The first equality follows since $p = P_i$ and thus we can rewrite the first sum fixing that P_i is offered and summing over all its predecessors. The final equality follows by our definition of $\hat{\alpha}_{i,p}$.

Fourth Constraint: The left hand side of the resource capacity constraint becomes:

$$
\begin{aligned}
&= T \sum_{i \in N} a_{v,i} [\sum_{p \in \Phi(i)} \sum_{S: i \in S, \phi_i(S) = p} h^*(S)(\mathrm{Pr}_i(i) - B_{p,i}) - \sum_{j \in T_i \setminus i} \sum_{S: j \in S, \phi_j(S) = i} h^*(S) B_{j,i}] \\
&= T \sum_{i \in N} a_{v,i} [\sum_{S: i \in S} h^*(S)(\mathrm{Pr}_i(i) - B_{p,i}) - \sum_{S: i \in S} h^*(S) \sum_{j \in \delta_i(S)} B_{j,i}] \\
&= T \sum_{i \in N} a_{v,i} \sum_{S: i \in S} h^*(S) \mathrm{Pr}_i(S) \leq c_v.
\end{aligned}
$$

Since all constraints are satisfied we have that the $\hat{\alpha}_{i,p}, \hat{\beta}_{i,p}$ of (18) is feasible to the RDLP and thus we have that $Z^{CBLP} \leq Z^{RDLP}$. This shows that $Z^{CBLP} = Z^{RDLP}$.

□