UNIVERSITY OF CALIFORNIA
Los Angeles

Prescribing for the Irrational:
A Data-Driven Approach to Modeling Irrational Choice

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Management

by

Yi-Chun Chen

2022

ABSTRACT OF THE DISSERTATION

Prescribing for the Irrational:
A Data-Driven Approach to Modeling Irrational Choice

by

Yi-Chun Chen
Doctor of Philosophy in Management
University of California, Los Angeles, 2022
Professor Velibor Mišić, Chair

Modeling consumer choice plays a central role in modern business operations and demand prediction. Precedented approaches assume consumers are rational, i.e., assume that consumer choice models follow rational choice theory and are based on the random utility maximization (RUM) principle. However, abundant evidence from marketing science, psychology, and behavioral economics has shown that consumer choice is not always consistent with the RUM assumption.

In this thesis, we introduce a nonparametric and data-driven choice model that is capable of representing any consumer choice, including those that are outside the RUM class. We theoretically characterize the model complexity and propose two practical estimation procedures to learn the model from data. Using real-world transaction data, we demonstrate the out-of-sample prediction ability of the proposed model and extract business insights.

We further transform the proposed model into effective prescriptions. We consider a mixed-integer optimization approach to find the optimal assortment that maximizes expected revenue under the proposed model. We introduce three formulations, analyze the necessary

conditions for integrality, and solve them at a large scale by applying Benders decomposition method. Using synthetically generated data, we demonstrate the tractability of our approach and its edge over heuristic approaches from the literature.

Finally, we generalize the estimation procedure of the proposed model as a general solution method for solving large-scale linear optimization problems. The proposed solution method is a randomized algorithm that first samples a set of columns and then solves the linear program that only consists of sampled columns. We theoretically characterize the algorithm's convergence property and apply it to a wide range of applications, including Markov decision processes, covering and packing problems, portfolio optimization, and choice model estimation.

The dissertation of Yi-Chun Chen is approved.

Christopher Siu Tang

Kumar Rajaram

Charles J. Corbett

Felipe Caro

Velibor Mišić, Committee Chair

University of California, Los Angeles

2022

To my late grandmother 徐玉容女士

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Velibor, for his support and guidance over the past five years. Velibor helped me become who I am right now, a version of Yi-Chun that I couldn't imagine five years ago when I joined the Ph.D. program. Velibor taught me how to be a researcher. I learned from him how to identify important research questions, marry ideas from different research disciplines, tell a compelling story, and change perspectives when research does not go as expected. Velibor also helped me become a better presenter. I used to feel nervous, shy, and have sweaty hands when presenting. Thanks to Velibor's feedback, encouragement, and the opportunities he created for me to present at various conferences, I now enjoy giving talks and sharing research ideas with great minds in our research communities through presentations. People always say Ph.D. is about struggling and hard work. I agree. However, thanks to Velibor's guidance and support, these efforts paid off, and I believe I had the best and most joyful five years one can have as a Ph.D. student. Velibor: it has been a great pleasure to work with you and learn from you. I always feel lucky for having you as my advisor. I thank you from the bottom of my heart.

I would like to thank Chris, who also played an important role during my Ph.D. studies. His endless energy and passion for research inspired me to set my goals high. I always look up to what he has overcome, which motivates me to be brave, resilient, extremely open-minded, and authentic. I would also like to thank Charles, Felipe, and Kumar, for being my thesis committee members, giving me critical feedback on my research, and supporting me in the academic job market. Charles: thank you for encouraging me to think beyond analytical results and make my research accessible to a broader audience. Felipe: thank you for teaching dynamic programming during my first year at UCLA. It was an eye-opening course that showed me the elegance of operations research. Also, thank you for giving me the opportunity to stay an additional year at DOTM as a postdoc researcher and learn from

you. Kumar: thank you for encouraging me to pursue an academic career since the very beginning of my Ph.D. studies. Also, thank you for giving me critical feedback on my mock job talk, which helped me significantly improve my presentation.

I am also very thankful to other faculty members and staff at DOTM. Auyon: thank you for listening to my INFORMS presentation practices many times over the past years and giving me great feedback. Elisa: thank you for inspiring me to be a better presenter and showing me how OR research can go beyond analytic results and touch people's lives. Fernanda: thank you for the guidance during my first two years at UCLA when you were the Ph.D. liaison. You created an excellent academic environment where my cohort and I throve and expanded our knowledge base. Francisco: thank you for sharing your job market experience and doing a mock interview with me. It helped me a lot. Neli: thank you for helping us reserve classrooms, apply for reimbursement, and assist us with many other important things. You are a hero to DOTM.

The DOTM Ph.D. office has been a wonderful "second home" to me for the last five years, thanks to those who set a good DOTM tradition and graduated earlier, Anna, Ali, Araz, Bobby, Prashant, and Taylor. In particular, thank you, Anna, Ali, and Prashant, for supporting me and sharing your valuable job market experience with me. Thank you, Taylor, for always bringing laughter and energy to the office. Thank you, Bobby, for your support in my fourth year when I had a big setback. I will miss the time that we had the mini-basketball games together in B501.

I owe a million thanks to my Ph.D. cohort, Nur and Pankaj. I will miss the time when we took classes together and prepared for the qualifying exam. I was fortunate to have you two as my cohort. Nur: Thank you for the support during the job market season. I am glad that we both made it! Your energy and courage always inspire me to be brave and adventurous. I would also like to thank my junior peers, especially Mirel, Jingwei, Jingyuan, Xinyi, and Zach. Without you, my last three years in the Ph.D. program would have been totally devastated by the pandemic.

Outside DOTM, I am very thankful to my undergrad classmates, Neng-Chieh and Hsiao-Yi, who also pursued Ph.D. degrees in Los Angeles around the same time as me. For international students, it is easy to feel lost and lonely when studying abroad. I am grateful that I had you two in Los Angeles. Neng-Chieh: Thank you for the many suggestions you gave me in the past. Your economist thinking always helped me make decisions rationally and wisely. I will miss the time we were roommates and talked about career goals all day. Hsiao-Yi: Thank you for being a listener to me since undergrad and supporting me with a no-repayment-schedule loan of one thousand dollars during my first summer when I ran out of my stipend (Yes, although Neng-Chieh helped me make rational decisions, I still misspent money on unnecessary things.) I will miss the time that we three went to Sun Nong Dan together to enjoy LA's best Korean beef soup, especially when A-Yao joined us when she was visiting LA from New Jersey.

I also want to thank my parents and sister for their love and support. I have been stubborn since I was young and sometimes unwilling to take any suggestions about career decisions. I thank my family for supporting me unconditionally and always believing in me. It has been nine years since I came to study in the US. I miss my family and friends in Taipei badly.

Lastly and most importantly, I want to thank my partner, İrem. Meeting you is the most beautiful thing that ever happened to me. Because of you, there hasn't been a moment in the last couple of years that I felt lost or lonely. Thank you for your love, support, and companionship. I don't know what I would have done without you. I look forward to moving to London with you and starting a new chapter of our life together.

(I dedicate this thesis to my late grandma, who always lives in my heart.)

VITA

2013	B.S. (Physics), National Taiwan University
2017	M.S. (Computational and Mathematical Engineering), Stanford University
2017	Anderson Fellowship, UCLA Anderson School of Management
2019	First Place, Student Paper Competition, Decision Analysis Society, Institute of Operations Research and Management Sciences
2019	Second Place, Jeff McGill Student Paper Competition, Revenue Management and Pricing Sector, Institute of Operations Research and Management Sciences
2020	Honorable Mention, George E. Nicholson Student Paper Competition, Institute of Operations Research and Management Sciences
2021	Dissertation Year Fellowship, UCLA
2022	The Dreze Dissertation Award, UCLA Anderson

PUBLICATIONS

Chen, Y.-C. and Mišić, V.V. (2022). Decision Forest: A Nonparametric Approach to Modeling Irrational Choice. *Management Science* (accepted)

Chen, Y.-C. and Mišić, V.V. (2021). Assortment Optimization under the Decision Forest

Model. *arXiv preprint arXiv:2103.14067*

Chen, Y.-C. and Mišić, V.V. (2021). Column-randomized Linear Programs: Performance Guarantees and Applications. *arXiv preprint arXiv:2007.10461v2*

Chen, Y.-C., Kochenderfer, M., Spaan, M. (2018). Improving Offline Value-Function Approximations for POMDPs by Reducing Discount Factors. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*

Chen, Y.-C., Wheeler, T., Kochenderfer, M. (2017). Learning Discrete Bayesian Networks from Continuous Data. *Journal of Artificial Intelligence Research*, 59, 103-132

Menda, K., Chen, Y.-C., Grana, J., Bono, J., Tracey, B., Kochenderfer, M., Wolpert, D. Deep Reinforcement Learning for Event-Driven Multi-Agent Decision Processes. *IEEE Transactions on Intelligent Transportation Systems*, 20(4), 1259-1268.

CHAPTER 1

Introduction

A common problem in business is to decide which products to offer to customers by using historical sales data. Specifically, a firm offers a set of products (*an assortment*) to a group of customers. Each customer makes a decision to either purchase one of the products or not purchase any of the products. The goal of the firm is to decide which products to offer, so as to maximize the expected revenue when customers exercise their preferences.

In order to make such decisions, it is critical to have access to a model for predicting customer choices. *Customer choice models* have been used to model and predict the substitution behavior of customers when they are offered different assortments of products. In general, a choice model can be thought of as a conditional probability distribution over all purchase options given an assortment that is offered. A rich literature spanning marketing, psychology, economics, and operations management has contributed to the understanding of choice models.

A widely-used assumption is that customers are rational, i.e., choice models are assumed to follow *rational choice theory* and are based on the *random utility maximization* (RUM) principle. The RUM principle requires that each product is endowed with a stochastic utility. When a customer encounters the assortment and needs to make a purchase decision, all utilities are realized and the customer will choose the product from the assortment with the highest realized utility. A consequence of the RUM principle is that whenever we add a product to an assortment, the choice probability of each incumbent product either stays the same or decreases. This property is known as *regularity* or *weak rationality*.

Table 1.1: Behavioral experiment involving subscriptions to *The Economist*; reproduced from [7]

Option	Price	Num. of Subscribers
Internet-Only	\$59.00	68
Print-&-Internet	\$125.00	32

Option	Price	Num. of Subscribers
Internet-Only	\$59.00	16
Print-Only	\$125.00	0
Print-&-Internet	\$125.00	84

However, customers are not always rational. There is an increasing body of experimental evidence, arising in the fields of marketing, economics, and psychology, which suggests that the aggregate choice behavior of individuals is not always consistent with the RUM principle and often violates the weak rationality property. A well-known example is the experiment involving subscriptions to *The Economist* magazine from [7], which is re-created in Table 1.1. One hundred MIT students were asked to make decisions given two different assortments of subscription options. In the first assortment in Table 1.1, two subscription options are given: "Internet-Only" (\$59.00) and "Print-&-Internet" (\$125.00). The first option is chosen by the majority of the students (68 out of 100). In the second assortment, the students are given one more option: "Print-Only" (\$125.00). For this second assortment, due to the obvious advantage in "Print-&-Internet" over "Print-Only", no one chose the latter option. But with the addition of the the "Print-Only" option, the number of subscribers of the "Print-&-Internet" option actually *increased* from 32 to 84, thus demonstrating a violation of the weak rationality property. Here, the option "Print-Only" serves as a decoy or an anchor: its presence can influence an individual's preference over the two other options "Internet-Only" and "Print-&-Internet". While this example comes from a classroom experiment, there has been an extensive peer-reviewed research literature on this phenomenon, known as the *decoy*

or *attraction* effect, since the seminal work of [70].

The example that we have described above is important for two reasons. First, even for this very simple example, no choice model based on RUM can perfectly capture the subscribers' observed behaviors; as such, choice predictions based on RUM models will be inherently biased if customers do not behave according to a RUM model. Second, the presence of irrationality in customer choice behavior can have significant operational implications on which products should be offered. As a concrete example, observe that in Table 1.1, assuming that customers have no outside option, the expected per-customer revenue arising from the first assortment is \$80.12, whereas the expected per-customer revenue of the second assortment is \$114.44 – an increase of more than 40%! Indeed, outside of experimental settings (as in the above example), deviations from rational behavior have been observed – and exploited – in business practice. For example, when Williams-Sonoma observed low sales of a bread bakery machine priced at \$275, it introduced a larger and more expensive version priced at \$429; few customers bought the new model, but sales of the original model almost doubled [96].

Motivated by the observations of non-rational choice in real-world scenarios, we want to address the following question:

"How can we model non-rational choice
from a data-driven perspective to create value?"

Here is our contribution:

1. **Decision Forest Model (Chapter 3)**: We propose a new type of choice model (Section 3.1), called the *decision forest* model, that is capable of modeling non-rational choice behavior, i.e., choice behavior that is inconsistent with the RUM principle. In this choice model, one assumes that the customer population can be described as a finite collection of customer types, where each customer type is associated with a binary

decision tree, together with a probability distribution over those types. Each decision tree defines a sequence of queries that the customer follows in order to reach a purchase decision, where each query involves checking whether a particular product is contained in the assortment or not. We provide several examples of how well-known behavioral anomalies, such as the decoy effect and the preference cycle, can be represented by this model. We also prove a key theoretical result: *any choice model*, whether it obeys the RUM property or not, can be represented as a probability distribution over binary decision trees (Theorem 1). We theoretically characterize the depth of the forest needed to fit a data set of historical assortments and prove that with high probability, a forest whose depth scales logarithmically in the number of assortments is sufficient to fit most data sets (Section 3.2). We also propose two practical algorithms – one based on column generation and one based on random sampling – for estimating such models from data (Section 3.3). Using real transaction data exhibiting non-rational behavior, we show that the model outperforms both rational and non-rational benchmark models in out-of-sample predictive ability (Section 3.4).

2. **Assortment Optimization Under the Decision Forest Model (Chapter 4)**: We further study the problem of finding the assortment that maximizes expected revenue under the decision forest model. This problem is of practical importance because it allows a firm to tailor its product offerings to profitably exploit deviations from rational customer behavior, as we have demonstrated in the *Economist* example in Table 1.1. We approach this problem from a mixed-integer optimization perspective and propose three different formulations (Section 4.1). We theoretically compare these formulations in strength, and analyze when these formulations are integral in the special case of a single tree. We propose a methodology for solving these problems at a large-scale based on Benders decomposition, and show that the Benders subproblem can be solved efficiently by primal-dual greedy algorithms when the master solution is fractional for two of our formulations, and in closed form when the master solution

is binary for all of our formulations (Section 4.2). Using synthetically generated instances, we demonstrate the practical tractability of our formulations and our Benders decomposition approach, and their edge over heuristic approaches (Section 4.3).

3. **Column-randomized Linear Program (Chapter 5)**: We generalize the estimation method of decision forest model as a solution method for linear programs that consist of a large number of columns but a relatively small number of constraints. Such large-scale linear programs are commonly used in business analytics for statistical estimation, revenue management, and vehicle routing. Our proposed method involves sampling a collection of columns according to a user-specified randomization scheme and solving the linear program consisting of the sampled columns (Section 5.2). We derive an upper bound on the optimality gap that holds with high probability and converges with rate $1/\sqrt{K}$, where K is the number of sampled columns, to the value of a linear program related to the sampling distribution. We further apply the proposed method to various applications, such as linear programs with totally unimodular constraints, Markov decision processes, covering problems and packing problems, and derive problem-specific performance guarantees (Section 5.3). We also generalize the method to the case that the sampled columns may not be statistically independent (Section 5.4). Finally, we numerically demonstrate the effectiveness of the proposed method in the cutting-stock problem and in nonparametric choice model estimation (Section 5.5).

We review the related literature of choice modeling and assortment optimization in Chapter 2. We conclude this thesis in Chapter 6. We relegate proofs and additional numerical results to the end of the thesis as appendices.

CHAPTER 2

Background and Literature

In this chapter, we overview the basic concept of choice modeling (Section 2.1) and assortment optimization (Section 2.2), and comment on our contribution to the literature. We also discuss related works in other disciplines (Section 2.3).

2.1 Choice Modeling

In consumer choice modeling, one concerns the purchase behavior of consumers when a set of products, i.e., an *assortment*, is available. Specifically, consider a market of N products, denoted by the set $\mathcal{N} \equiv \{1, 2, \ldots, N\}$. The full set of purchase options is denoted by $\mathcal{N}_+ \equiv \{0, 1, 2, \ldots, N\}$, where 0 corresponds to an outside or "no-purchase" option and $i \in \mathcal{N}$ corresponds to the action of buying product i. An assortment S is a subset of $\mathcal{N}$. When offered the assortment S, a customer may choose to purchase one of the products in S, or choose the no-purchase option 0. We denote S_+ as $S \cup \{0\}$ for all $S \in \mathcal{N}$.

The behavior of a customer population is represented through a *discrete choice model*. A discrete choice model is defined as a conditional probability distribution $\mathbf{P}(\cdot \mid \cdot) : \mathcal{N}_+ \times 2^{\mathcal{N}} \to [0, 1]$ that gives the probability of an option in $\mathcal{N}_+$ being purchased when the customer is offered a particular set of products; that is, $\mathbf{P}(o \mid S)$ is the probability of the customer choosing the option $o \in S \cup \{0\}$, when offered the assortment $S \subseteq \mathcal{N}$. Note that $\mathbf{P}(o \mid S) = 0$ whenever $o \notin S \cup \{0\}$, which models the fact that the customer cannot choose a product o that is not in the assortment S.

2.1.1 Rational Choice Model

The most common way to construct the choice probability distribution $\mathbf{P}(\cdot \mid \cdot)$ is to follow *random utility maximization* (RUM) principle. In a RUM discrete choice model, each option $o \in \mathcal{N}_+$ is associated with a random variable V_o, which corresponds to a stochastic utility for the option o. When offered the assortment $S \subseteq \mathcal{N}$, the customer's choice is given by the random variable $\arg\max_{o \in S_+} V_o$; in other words, the utilities $V_0, \ldots, V_N$ are realized, and the customer chooses the option from S_+ that offers the highest utility. Under such a choice model, the choice probabilities are given by

$$\mathbf{P}(o \mid S) = \mathbf{P}(V_o > V_j \text{ for all } j \in S^+, j \neq o). \tag{2.1}$$

Since consumers choose the option with highest utility, one also refers RUM choice models as *rational* choice models [102]. By specifying the joint distribution of the random vector $(V_0, V_1, \ldots, V_N)$, one can obtain many different types of choice models. Here we introduce two examples.

Definition 1 (Multinomial Logit Model (MNL)) *In a MNL model, one assumes that $V_i = v_i + \epsilon_i$ for $i \in \mathcal{N}^+$, where v_i is a deterministic constant that represents the expected utility of option i and ϵ_i is a standard Gumbel random variable. Also, without loss of generality, we set $v_0 = 0$. One can show that in a MNL model,*

$$\mathbf{P}\left(i \mid S\right) = \frac{\exp(v_i)}{1 + \sum_{j \in S} \exp(v_j)}, \quad \forall i \in S.$$

Note that MNL model is a parametric model characterized by parameters $(v_1, \ldots, v_N)$.

Definition 2 (Mixed-MNL Model) *In a mixed-MNL model, one assumes that there are K customer segments in the market. Each customer segment k accounts for the ratio λ_k of the total population and follows a MNL model with parameters $(v_{k1}, \ldots, v_{kN})$ to make decisions. The resulting choice probability is given by*

$$\mathbf{P}\left(i \mid S\right) = \sum_{k=1}^{K} \lambda_k \cdot \frac{\exp(v_{ki})}{1 + \sum_{j \in S} \exp(v_{kj})}, \quad \forall j \in S.$$

MNL and mixed-MNL models are among the most commonly-used choice model; we refer the reader to [9] and [115] for more details. We denote that both MNL and mixed-MNL models can be estimated by maximum likelihood estimation (MLE). The MLE of MNL model can be solved exactly as a concave maximization problem, while the MLE of mixed-MNL model is generally solved by expectation-maximization (EM) algorithm.

There has been a significant effort to develop "universal" choice models within the RUM class. A well-known universality result in choice modeling comes from the paper of [85], which showed that any RUM choice model can be approximated to an arbitrary precision by a mixture of MNL models, i.e., mixed-MNL model. Outside of logit models, earlier research proposed the ranking-based model (also known as the *stochastic preference* model), in which one represents a choice model as a probability distribution over rankings.

Definition 3 (Ranking-based choice model [20, 52]) *A ranking or a permutation σ over options $\mathcal{N}_+$ is a bijection from $\mathcal{N}_+$ to $\mathcal{N}_+$ such that option i is preferred to option j if and only if $\sigma(i) < \sigma(j)$. A ranking-based model assumes that there are K customer segments in the market. Each segment k accounts for the ratio λ_k of the total customer population and makes decisions according to a ranking σ_k. The choice probability of a ranking-based model is given by*

$$\mathbf{P}\left(i \mid S\right) = \sum_{k=1}^{K} \lambda_k \cdot \mathbb{I}\{i = \arg\max_{i \in S_+} \sigma_k(i)\}, \quad \forall i \in \mathcal{N}_+, \tag{2.2}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function, i.e., $\mathbb{I}\{B\} = 1$ if B is true and 0 otherwise.

[20] showed that the class of RUM choice models is equivalent to the class of ranking-based models. Later, the seminal paper of [52] developed a data-driven approach for making revenue predictions via the ranking-based model; specifically, the method involves computing the worst-case revenue of a given assortment over all ranking-based models that are consistent with the available choice data. Subsequent research on ranking-based models has studied other estimation approaches [72, 73, 87, 119], as well as methods for obtaining optimal or

near-optimal assortments; see Section 2.2. Another universal RUM choice model is the Markov chain model of customer choice [19]. By modeling substitution behavior between products as transitions between states in the Markov chain, the model provides a good approximation to any choice model based on the RUM principle. Since the original paper of [19], later research has considered other methods of estimating such models from limited data [111] as well as methods for solving core revenue management problems under such models [40, 42, 56].

A property that is satisfied by rational choice models, i.e., RUM choice models, is the *regularity* or *weak rationality property*, which corresponds to the following family of inequalities.

Definition 4 (Weak Rationality)

$$\mathbf{P}(i \mid S \cup \{j\}) \leq \mathbf{P}(i \mid S), \quad \forall\ S \subseteq \mathcal{N},\ i \in S_+,\ j \in \mathcal{N} \setminus S. \tag{2.3}$$

In words, whenever we add a new product j to an assortment S, the choice probability of each existing product in S cannot increase. Note that every RUM choice model satisfies the regularity property; however, there exist discrete choice models that satisfy the regularity property and that are outside of the RUM class [20].

2.1.2 Non-rational Choice Model

The study of non-rational choice has its roots in the seminal work of [74], which demonstrated how expected utility theory fails to explain certain choice phenomena, and proposed prospect theory as an alternative model. Since this paper, significant research effort has been devoted to the study of non-rational decision making. Within this body of research, the concept of choice modeling beyond RUM class, i.e, non-rational choice modeling, relates to the significant empirical and theoretical work in behavioral economics on context-dependent choice [118], which includes important context effects such as the attraction effect [70], the compromise effect [110], and the similarity effect [117].

Recently, new choice models have been proposed for modeling behavior outside of the RUM class. Within behavioral economics, examples include the generalized Luce model [48] and the perception-adjusted Luce model (PALM) [49]. The focus of these papers is descriptive, in that they develop axiomatic theories for new models. In contrast, the focus of this thesis is prescriptive, as we develop optimization-based methods for estimating our proposed model from data and making operational decisions.

Within operations management, examples of new choice models include the generalized stochastic preference (GSP) model [10], the general attraction model (GAM) [62], and the HALO-MNL model [82]. The main difference between the proposed model in this thesis and these prior models is in expressive power. As we will show in Section 3.1.5, our choice model is universal and is able to represent *any* discrete choice model that may or may not be in the RUM class; in contrast, for each of the generalized Luce model, PALM, GAM, HALO-MNL model and GSP model, there either exist choice models that do not obey the RUM principle and cannot be represented by the model, or the representational power of the model is unknown.

2.1.3 Universal Choice Models

Within the economics literature, there are two classes of non-rational models that also have the universal expressive power (as our Theorem 1 in Section 3.1.5). We review these two classes of models in details, followed by the contribution of our proposed model to this "universality" paradigm.

The first is the class of game tree models and randomized game tree models. The game tree model was proposed by [126] as a model for how an option is deterministically chosen from a given choice set. In a game tree model, the tree encodes a hierarchy where each leaf corresponds to a product, and each non-leaf node correspond to a decision maker that is endowed with a ranking over the product universe. To make a decision, one starts at the non-leaf nodes whose children are leaves, and the decision maker at each such non-leaf

node chooses its most preferred product according to its ranking. The parent nodes of those non-leaf nodes then choose from the products chosen by their children. This process repeats until reaching the decision maker at the root of the tree who makes the final decision. The model can be thought of as a representation of how *an organization*, through several rounds of decision making, reaches a decision. This type of model bears a superficial similarity to ours in that both models involve trees. However, the details differ significantly: in our trees, as we will see in Section 3.1.1, the decision process starts at the root (rather than the leaves) and involves checking for the existence/non-existence of a product in the assortment, until reaching a leaf. In addition, the trees that we describe are always binary, whereas game trees can in general be non-binary trees. Since the paper of [126], other research has extended this type of model in different ways. For example, [69] considers game trees where all of the decision makers follow the same ranking. In the subsequent literature, the paper of [77] considers the randomized game tree model, where one assumes a probability distribution over the tuple of rankings for the non-leaf nodes; that paper shows that this model can represent any discrete choice model, which is similar to our universality result (Theorem 1 in Section 3.1.5).

The second type of model that has the universality property is the pro-con model in the working paper of [44]. In this choice model, one considers two sets of rankings: "pro" rankings and "con" rankings. Then, over the union of the pro and con rankings, one posits a signed probability distribution, where the pro rankings receive positive probabilities, and the con rankings receive negative probabilities. The choice probability of a product given an assortment is the sum of the (positive) probabilities for the pro rankings for which that product is highest ranked, plus the sum of the (negative) probabilities for the con rankings for which that product is lowest ranked. The model aims to represent the idea of a decision maker who makes decisions by listing the pros and cons of an option, adding up the pros and subtracting the cons. The main result of [44] is the result that every choice model can be represented as a pro-con model, which is again similar to our universality result (Theorem 1

in Section 3.1.5).

While these two classes of models also have the universality result, to date, all research on the game tree models and the pro-con model has been descriptive in nature, and has focused on categorizing and relate these models to existing models. These papers do not include methodological contribution: specifically, there has not been any research that answers the question of how to efficiently estimate these models from data, and that empirically validates these models on real data. In contrast, in our paper, we show that the proposed model can be estimated from data in two tractable ways (Section 3.3), and its performance can be validated using real transaction data (Section 3.4). In other words, the decision forest model is not only theoretically rich in its expressive power, but also practical and ready to be used by practitioners.

2.2 Assortment Optimization

Assortment optimization is a basic operational problem faced by many firms. In its simplest form, the problem can be posed as follows. A firm has a set of products that it can offer, and a set of customers who have preferences over those products; what is the set of products (an *assortment*) the firm should offer so as to maximize the revenue that results when the customers choose from these products? We formally define the problem as follows.

Definition 5 (Assortment optimization problem)

$$\underset{S\subseteq\mathcal{N}}{maximize} \quad \sum_{i=1}^{N} \bar{r}_i \cdot \mathbf{P}\left(i \mid S\right), \tag{2.4}$$

where $\bar{r}_i$ is the marginal revenue of product i, $\mathbf{P}(\cdot \mid \cdot)$ is a choice model, and the objective is the expected revenue.

The problem of assortment optimization has been extensively studied in the operations management community; we refer readers to [63] for a recent review of the literature. The

literature on assortment optimization has focused on developing approaches for finding the optimal assortment under many different RUM choice models, such as the MNL model [112, 113], the latent class MNL model [25, 86, 106], the nested logit model [38, 2] the Markov chain choice model [41, 56] and the ranking-based model [4, 5, 55].

Assortment optimization problem is also related to the literature on product line design found in the marketing community. While assortment optimization is more often focused on the tactical decision of selecting which existing products to offer, where the products are ones that have been sold in the past and the choice model comes from transaction data involving those products, the product line design problem involves selecting which new products to offer, where the products are candidate products (i.e., they have not been offered before) and the choice model comes from conjoint survey data, where customers are asked to rate or choose between hypothetical products. Research in this area has considered different approaches to solve the problem under the ranking-based/first-choice model [8, 13, 84] and the multinomial logit model [30, 105]; for more details, we refer the reader to the literature review of [13].

The assortment optimization technique proposed in this thesis is most closely related to [8] and [13], both of which present integer optimization formulations of the product line design problem when the choice model is a ranking-based model. As we will see later, our formulations LeafMIO and SplitMIO can be viewed as generalizations of the formulations of [8] and [13], respectively, to the decision forest model. In addition, the paper of [13] develops a specialized Benders decomposition approach for its formulation, which uses the fact that one can solve the subproblem associated with each customer type by applying a greedy algorithm. We will show in Section 4.2 that this same property generalizes to two of our formulations, LeafMIO and SplitMIO, leading to tailored Benders decomposition algorithms for solving these problems at scale.

Beyond these specific connections, the majority of the literature on assortment optimization and product line design considers rational choice models, whereas our paper contributes

a methodology for non-rational assortment optimization. Fewer papers have focused on choice modeling for irrational customer behavior; see Section 2.1.2. An even smaller set of papers has considered assortment optimization under non-rational choice models, which we now review. The paper of [58] considers assortment optimization under the two-stage Luce model, and develops a polynomial time algorithm for solving the unconstrained assortment optimization problem. The paper of [104] considers a context-dependent utility model where the utility of a product can depend on other products that are offered and that can capture compromise, attraction and similarity effects; the paper empirically demonstrates how incorporating context effects leads to a predicted increase of 5.4% in expected profit.

Relative to these papers, the assortment optimization problem considered in this thesis differs in that it considers the decision forest model. As noted earlier, the decision forest model can represent any type of choice behavior, and as such, an assortment optimization methodology based on such a model is attractive in terms of allowing a firm to take the next step from a high-fidelity model to a decision. In addition, our methodology is built on mixed-integer optimization. This is advantageous because it allows a firm to leverage continuing improvements in solution software for integer optimization (examples include commercial solvers like Gurobi and CPLEX), as well as continuing improvements in computer hardware. At the same time, integer optimization allows firms to accommodate business requirements using linear constraints, which further enhances the practical applicability of the approach. Lastly, integer optimization also allows one to take advantage of well-studied large-scale solution methods for integer optimization problems. One such method that we focus on in this paper is Benders decomposition, which has seen an impressive resurgence in recent years for delivering state-of-the-art performance on large-scale problems such as hub location [35], facility location [57] and set covering [36]; see [101] for a review of the recent literature. Stated more concisely, *the main contribution of our approach is a general-purpose methodology for assortment optimization under a general-purpose choice model.*

2.3 Tree Ensembles in Machine Learning and Other Fields

Our decision forest model is also related to the rich literature on tree models in machine learning. Many machine learning methods construct binary tree models that can be used for classification or regression, such as ID3 [97], C4.5 [98] and classification and regression trees (CART; [22]). In addition, there are also many predictive models that consist of ensembles or forests of trees, such as random forests [21] and boosted trees [60]. Recently, tree ensemble models are also used for estimating and inferring treatment effects [125]. The main difference between our work and prior work in machine learning is in the use of forests for discrete choice modeling, that is, using a forest to probabilistically model how customers choose from an assortment. To the best of our knowledge, the use of tree ensemble models for discrete choice modeling has not been proposed before.

Finally, we note that the term *decision forest* coincides with the general name for tree ensemble models in the literature; we refer readers to the survey paper [103]. However, same as *decision tree* which has different meanings in different research communities, the term *decision forest* also has been used in various ways to refer to a collection of trees. In this thesis, we restrict our discussion on choice modeling.

CHAPTER 3

Decision Forest

In this chapter[1], we present a new model for customer choice based on representing the customer population as tree ensemble (Section 3.1). We theoretically characterize the depth of the trees needed to fit a data set of historical assortments (Section 3.2). We also propose two practical algorithms for estimating the presented model from data (Section 3.3). Using real transaction data exhibiting non-rational behavior, we demonstrate the out-of-sample predictive ability of the model and further extract business insights from the data (Section 3.4). All proofs in this chapter are relegated to Section A.1 in Appendix.

3.1 The Decision Forest Model

In this section, we present our decision forest choice model. We first introduce binary decision trees and define how customers make purchases according to such decision trees. We then define our choice model, compare it to the ranking-based model, describe a couple of well-known examples of behavioral anomalies that can be represented by our model. Finally, we establish our first key theoretical result, namely that decision forest models can represent *any* customer choice model.

[1]This chapter is based on my doctoral research work "Decision Forest Model: A Nonparametric Approach to Modeling Irrational Choice" [34].

3.1.1 Decision Trees

The choice model that we will define is based on representing the customer population through a collection of customer *types*. A customer type is associated with a purchase decision tree t, which is structured as a directed binary tree graph. We use $\mathbf{leaves}(t)$ and $\mathbf{splits}(t)$ to denote the sets of leaf nodes and non-leaf nodes (also called *split* nodes) of decision tree t, respectively. For each split node s in $\mathbf{splits}(t)$, we define $\mathbf{LL}(s)$ and $\mathbf{RL}(s)$ as the sets of leaves that belong to the left and right subtree rooted at split node s, respectively. Similarly, for each leaf node ℓ, we define $\mathbf{LS}(\ell)$ and $\mathbf{RS}(\ell)$ as the sets of all split nodes for which ℓ is to the left or to the right, respectively. We use $r(t)$ to denote the root node of tree t. Each node in the tree, whether it is a split or a leaf, is associated with a purchase option; let x_v denote the purchase option associated with node v.

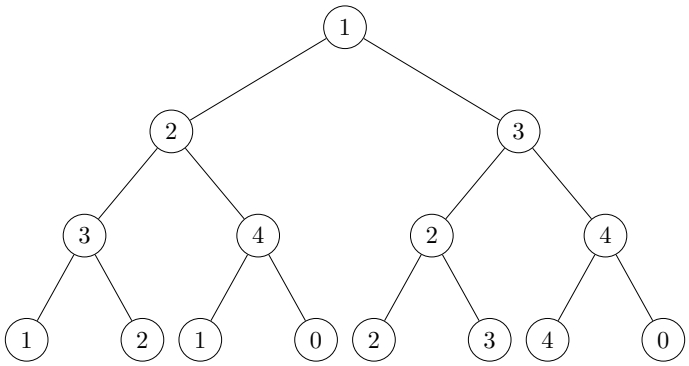


Figure 3.1: An example of a decision tree.

Given an assortment $S \subseteq \mathcal{N}$ and a customer following purchase decision tree t, the customer will make their purchase decision as follows: starting at the root node $r(t)$, the customer will check whether the purchase option of that node is contained in the assortment S or not. If this option is a member of S, the customer proceeds to the left child node; otherwise, if it is not in the assortment S, the customer proceeds to the right child node. The process then repeats until a leaf node is reached. The purchase option o that corresponds to the leaf node is then the customer's purchase decision.

Figure 3.1 visualizes an example of a purchase decision tree. Consider a customer following the tree in Figure 3.1, and consider three assortments: $S_A = \{1, 2, 4\}$, $S_B = \{2, 4\}$, and $S_C = \{1, 3\}$. When offered S_A, she will choose product 2; when offered S_B, she will choose product 4; and finally, when offered S_C, she will choose the no-purchase option 0.

To ensure that a purchase decision tree is well-defined, we impose three additional requirements on it.

Definition 6 (Structural requirements of purchase decision trees) *A purchase decision tree t must satisfy the following three requirements:*

1. *For each split $s \in \mathbf{splits}(t)$, $x_s \in \mathcal{N}$.*
2. *For each leaf $\ell \in \mathbf{leaves}(t)$, $x_\ell \in \left(\{0\} \cup \bigcup_{s \in \mathbf{LS}(\ell)} \{x_s\}\right)$.*
3. *For each leaf $\ell \in \mathbf{leaves}(t)$ and any two distinct splits s and s' from set $\mathbf{LS}(\ell) \cup \mathbf{RS}(\ell)$, $x_s \neq x_{s'}$.*

Requirement 1 is needed because the no-purchase option can never belong to the assortment; thus, setting $x_s = 0$ at a particular split will force the decision process to always proceed to the right. Requirement 2 is needed to ensure that each possible purchase decision is consistent with the path followed in the tree and that the customer is only able to select products that have been observed to exist in the assortment. An example of a tree that does not satisfy the second requirement is given in Figure 3.2. Observe that if the assortment $\{1, 2\}$ is offered to a customer following this tree, the customer will choose to purchase product 3, which is not part of the assortment. As another example, if the assortment $\{2, 3\}$ is offered, the customer would choose product 1, which again does not exist in the assortment. Finally, Requirement 3 enforces that each product appears at most once in the split nodes on the path from the root $r(t)$ to any leaf ℓ. This requirement ensures that for each leaf in the tree, there exists some assortment that will be mapped to it. An example of a tree that does not satisfy the requirement is given in Figure 3.3, where product 1 appears twice on the

path from the root to the third leaf node from the left. In order to reach this leaf, product 1 must simultaneously be included and not included in the assortment, which is impossible. As a result, this leaf node can never be reached given any assortment.

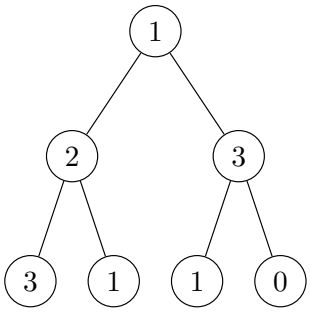


Figure 3.2: A decision tree that violates Requirement 2.

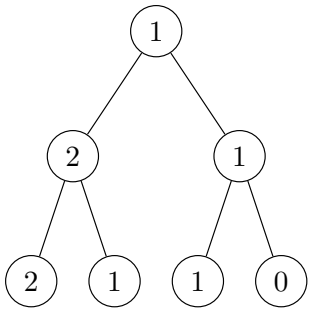


Figure 3.3: A decision tree that violates Requirement 3.

Before describing our choice model, we introduce two useful definitions. We define the *depth* of tree t as $\text{Depth}(t) = \max\{\text{dist}(r(t), \ell) + 1 \mid \ell \in \textbf{leaves}(t)\}$, where the distance $\text{dist}(r(t), l)$ is the number of edges connecting leaf ℓ and root $r(t)$. Note that our definition of depth starts at 1, i.e., a tree consisting of a single leaf would have $\text{Depth}(t) = 1$. We also say that a tree is *balanced* if and only if all leaves in the tree have same distance to the root. For example, the tree in Figure 3.1 is a balanced tree of depth 4. Lastly, notice that Requirement 3 implies that all purchase decision trees have depth at most $N+1$. Therefore, there are only finitely many purchase decision trees that satisfy Requirement 1-3.

3.1.2 Decision Forest Model

We now present our choice model based on purchase decision trees. Consider a collection F of purchase decision trees; we will refer to F as a *forest*. Let $\boldsymbol{\lambda} : F \to [0, 1]$ be a probability distribution over all decision trees in forest F. Each tree t in the decision forest F can be thought of as a customer type. For each type t, the probability λ_t can be thought of as the percentage of customers in the population that behave according to the purchase decision

tree t; alternatively, one can think of λ_t as the probability that a random customer will choose according to tree t. Define $\hat{A}(S,t) \in \mathcal{N}_+$ as the purchase option that a customer associated with decision tree t would choose when an assortment S is given. Therefore, for any assortment S, the probability that a random customer selects option $o \in \mathcal{N} \equiv \{0,1,\ldots,N\}$ is

$$\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = \sum_{t \in F} \lambda_t \cdot \mathbb{I}\{o = \hat{A}(S,t)\}, \tag{3.1}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function. Note that if a product $p \in \mathcal{N}$ is not in assortment S, i.e., $p \notin S$, then $\mathbf{P}^{(F,\boldsymbol{\lambda})}(p \mid S) = 0$; this is a consequence of Requirement 2 from Definition 6 in Section 3.1.1, that is, for any leaf, we must have $x_\ell \in \left(\{0\} \cup \bigcup_{s \in \mathbf{LS}(\ell)} \{x_s\}\right)$. We refer to the pair $(F, \boldsymbol{\lambda})$ as a *decision forest model*.

3.1.3 Comparison to Ranking-based Model

Our decision forest model resembles the ranking-based model of [52]; see Definition 3. In the model of [52], each customer type corresponds to a *ranking* over all products and the no-purchase option. When offered an assortment, a customer will choose the product in the assortment that is most preferred according to that customer's ranking. A ranking-based model can be represented as a collection Σ of rankings and a probability distribution $\boldsymbol{\lambda}$ over rankings in Σ. The resulting choice probability is given by Equation (2.2).

The ranking-based model and the decision forest model are structurally similar, in that they are both probability distributions over a collection of "primitive" choice models. However, it turns out that the decision forest model is more general than the ranking-based model, which we formalize in the proposition below.

Proposition 1 *Let $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ be a collection of rankings and $\boldsymbol{\lambda}$ be a probability distribution over them. Then there exists a forest F such that, for all $o \in \mathcal{N}_+$ and $S \subseteq \mathcal{N}$,*

$$\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = \mathbf{P}^{(\Sigma,\boldsymbol{\lambda})}(o \mid S).$$

Note that the class of RUM choice models is equivalent to the class of ranking-based models [20]. Thus, Proposition 1 also implies that we can represent any RUM choice model by a decision forest model. The proof of Proposition 1 is presented in Section A.1.1, where we explicitly represent each ranking in Σ by a purchase decision tree. We illustrate the same idea in the following example.

Example 1 (RUM choice model) *Consider a ranking-based model* $(\Sigma, \boldsymbol{\lambda})$ *that consists of two rankings* $\sigma_1 = \{2 \succ 3 \succ 0\}$ *and* $\sigma_2 = \{3 \succ 2 \succ 1 \succ 0\}$*, where* $a \succ b$ *denotes that* a *is preferred to* b*, and distribution* $\boldsymbol{\lambda} = (0.4, 0.6)$*. This ranking-based choice model can be represented by a decision forest model* $(F, \boldsymbol{\lambda})$ *such that* F *consists of two trees* t_1 *and* t_2 *(see Figure 3.4). The ranking* σ_1 *and the decision tree* t_1 *give the same decision process: if product* 2 *is in the assortment, then the customer buys it; otherwise, if product* 2 *is not in the assortment but* 3 *is, then the customer buys product* 3*; otherwise, if both* 2 *and* 3 *are not available, the customer will not buy anything. The equivalence between the ranking* σ_2 *and the tree* t_2 *can be argued in the same way. By using the same probability distribution, the decision forest model* $(F, \boldsymbol{\lambda})$ *is equivalent to the ranking-based model* $(\Sigma, \boldsymbol{\lambda})$*.*

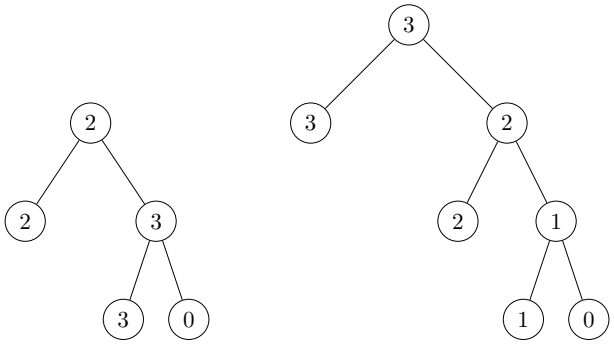


Figure 3.4: Decision tree t_1 (left) and t_2 (right) give the same purchase decisions as the two rankings σ_1 and σ_2 in Example 1, respectively.

We remark that the reverse statement of Proposition 1 is not true. That is, there exist decision forest models that cannot be represented as a ranking-based model. For instance,

consider a decision forest model that consists of a single purchase decision tree t as in Figure 3.5, for which the probability λ_t must be 1. This decision tree t gives the following choice probabilities: $\mathbf{P}(1 \mid \{1,2\}) = 1$ and $\mathbf{P}(1 \mid \{1\}) = 0$. Since the inequality $\mathbf{P}(1 \mid \{1,2\}) > \mathbf{P}(1 \mid \{1\})$ violates the regularity property (inequality (2.3)), no RUM choice model can satisfy both $\mathbf{P}(1 \mid \{1,2\}) = 1$ and $\mathbf{P}(1 \mid \{1\}) = 0$.

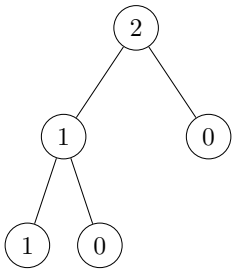


Figure 3.5: A decision tree that cannot be modeled by a ranking-based model.

Before continuing, it is worth interpreting how choices are made by a purchase decision tree and differentiating them from those of a ranking. Example 1 shows how a ranking is effectively a purchase decision tree that is constrained to always grow to the right. In addition, each purchase decision of a leaf corresponds to the product on its parent split (with the exception of the right-most leaf, which is always the no-purchase option). A customer who chooses according to a ranking behaves in the following way: they check the assortment in accordance with a sequence of products (their ranking); as soon as they reach a product that is contained in the assortment, they choose it; and if they go through their entire sequence without successfully finding a product, they choose the no-purchase option. Such a decision process is always forced to immediately choose a product when the existence of the product in the assortment has been verified. In contrast, for a purchase decision tree, the decision process can be more complicated: if the customer checks for a product and finds that it is indeed contained in the assortment, the customer is not forced to immediately choose the product; instead, the customer can continue checking for other products in the assortment before making a purchase decision. This difference is why purchase decision trees

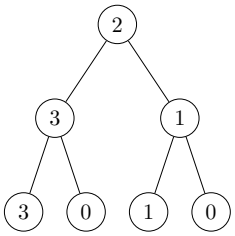
are potentially valuable: a purchase decision tree can model more complicated, assortment-dependent customer behavior than a ranking can. Indeed, in Section 3.1.4, we will see some simple examples of non-rational behavior that can be represented in the decision forest framework.

3.1.4 Modeling Irrational Behavior by Decision Forest Models

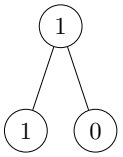
Research in marketing, psychology, and economics has documented numerous examples of choice behavior that is inconsistent with the RUM principle. We show how two well-known examples of irrational choices, the decoy effect and the preference cycle, can be modeled by decision forests.

Example 2 (Decoy Effect) *In marketing, the decoy effect is the phenomenon whereby consumers tend to change their preference between two options when a third option exists and it is asymmetrically dominated. The experiment involving the* The Economist *from [7], shown in Table 1.1, is an example of this effect. When the option "Print-Only" is strictly dominated by option "Print-&-Internet" (same price but with additional online access), the preference between the other two options changes.*

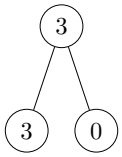
We model the example in Table 1.1 as follows: denote the subscription options "Internet-Only", "Print-Only", and "Print-&-Internet" as products 1*,* 2*, and* 3*, respectively. Define* $F = \{t_1, t_2, t_3\}$ *as in Figure 3.6 and the corresponding distribution as* $\boldsymbol{\lambda} = (0.52, 0.16, 0.32)$*; it can be verified that this model leads to the choice probabilities in Table 1.1. Note that customers following* t_2 *will always choose "Internet-Only"(option* 1*), regardless of whether "Print-Only"(option* 2*) is available or not. Similarly, customers following* t_3 *will always choose "Print-&-Internet" (option 3). But for customers following* t_1*, the preference between "Internet-Only" (option* 1*) and "Print-&-Internet" (option* 3*) changes when"Print-Only" (product* 2*) exists. As in Figure 3.6a, if product* 2 *is included in the assortment, the decision process proceeds to the left subtree and chooses according to the ranking* $\{3 \succ 0\}$*, i.e., if*



(a) Tree t_1 with $\lambda_1 = 0.52$



(b) Tree t_2 with $\lambda_2 = 0.16$



(c) Tree t_3 with $\lambda_2 = 0.32$

Figure 3.6: The forest-distribution pair $F = \{t_1, t_2, t_3\}$ and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ that can model the decoy effect in *The Economist* subscription example in Table 1.1.

product 3 *exists then we buy it; otherwise, we do not buy anything. If product* 2 *is not included in the assortment, the decision process proceeds to the right subtree and chooses according to the ranking* $\{1 \succ 0\}$*, i.e., if product* 1 *exists then we buy it; otherwise, we do not buy anything. Thus, customers of type* t_1 *account for the decoy effect observed in Table 1.1.*

Example 3 (Preference Cycle) *The preference cycle is a behavioral anomaly in which the preference relation is not transitive. A classic example is given by [116] and is re-created in Table 3.1 (see also [102]). Participants were offered gambles varying in winning probabilities but with similar payoffs. One group of participants behaved in the following way: when offered two gambles with similar probabilities, they preferred the gamble with the larger payoff. Specifically, they preferred* A *to* B*,* B *to* C*,* C *to* D*, and* D *to* E*. However, when offered gambles where the winning probabilities were significantly different, they would prefer the gamble with the higher winning probability, e.g., preferring* E *to* A*.*

We can use a purchase decision tree to model this type of preference cycle, as in Figure 3.7. It is easy to see that, when assortments $\{A, B\}$*,* $\{B, C\}$*,* $\{C, D\}$*,* $\{D, E\}$*,* $\{A, E\}$ *are given, participants who follow the decision tree would choose* A*,* B*,* C*,* D*, and* E*, respectively. Note that the right subtree of the root node corresponds to the ranking* $\{A \succ B \succ C \succ D \succ E\}$ *but the left subtree corresponds to the ranking* $\{E \succ A\}$*, therefore leading to the cycle.*

Gamble	Prob. of Winning	Payoff
A	7/24	5.00
B	8/24	4.75
C	9/24	4.50
D	10/24	4.25
E	11/24	4.00

Table 3.1: Gambles to demonstrate preference cycle [116].

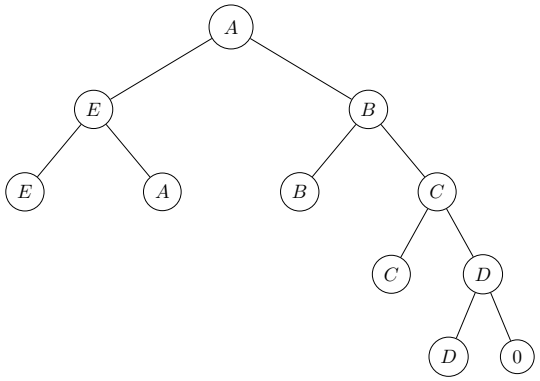


Figure 3.7: A decision tree representation of a preference cycle.

3.1.5 Decision Forest Models are Universal

As we have shown that two classic examples of irrational choices can be modeled by decision forest model, a natural question to ask is: what is the class of the choice models that can be represented by a decision forest model? Stated differently, for any given general choice model $\mathbf{P}(\cdot \mid \cdot)$, does there exist a forest F and a probability distribution $\boldsymbol{\lambda}$ such that $\mathbf{P}(o \mid S) = \mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S)$ for every assortment S and purchase option o? The answer, given by Theorem 1, is in the affirmative.

Theorem 1 *Assume a universe of N products. Let F_d be the collection of all purchase decision trees that satisfy Requirement 1-3 in Definition 6 (Section 3.1.1) and are of depth at most d. For any customer choice model $\mathbf{P}(\cdot \mid \cdot)$, there exists a distribution $\boldsymbol{\lambda}$ over F_{N+1} such that*

$$\mathbf{P}(o \mid S) = \mathbf{P}^{(F_{N+1},\boldsymbol{\lambda})}(o \mid S), \tag{3.2}$$

for any assortment S and purchase option $o \in \mathcal{N}_+$.

The proof of Theorem 1, given in Section A.1.2, follows by explicitly constructing a forest of balanced trees of depth $N+1$ that gives identical choice probabilities to $\mathbf{P}$, where each tree corresponds to a possible combination of purchase decisions on all 2^N assortments and the probability of each tree is given by the product of the choice probabilities of those purchase decisions. Theorem 1 shows that the decision forest model is *universal*: *any* choice model can be represented by a decision forest model. Additionally, Theorem 1 gives another way to prove Proposition 1: since any choice model can be modeled by the decision forest model, ranking-based choice models are thus included as a special case.

3.2 Model Complexity of the Decision Forest Model

In this section, we theoretically analyze the problem of estimating a decision forest model $(F, \boldsymbol{\lambda})$ from data corresponding to a set of M historical assortments. While Theorem 1 implies that a forest of trees of depth at most $N + 1$ is sufficient to fit any data set, this choice may not be attractive when N is large, as the trees will be extremely deep and contain an exponentially large number of leaves. Therefore, we ask the question of whether it is possible to fit the data using a "simple" decision forest model. In Section 3.2.1, we define the estimation problem precisely and provide further motivation for considering simple decision forest models. We then show how the number of assortments relates to the depth of the trees, number of leaves of the trees, and number of trees in the forest.

3.2.1 Motivation for Simple Decision Forests

To motivate the value of considering simple forests, let us assume that we have access to sales rate information for a collection of historical assortments $S_1, \ldots, S_M$, and let v_{o,S_m} denote the probability with which customers selected option $o \in \mathcal{N}_+$ when assortment S_m was offered, for $m = 1, 2, \ldots, M$. We let $\mathbf{v}_S$ denote the vector of $v_{o,S}$ values for each historical assortment S, and we use $\mathcal{S} = \{S_1, \ldots, S_M\}$ to denote the set of historical assortments.

We will make the assumption that v_{o,S_m} is known exactly, that is, $v_{o,S_m} = \mathbf{P}(o \,|\, S_m)$ for every (o, S_m), where $\mathbf{P}(\cdot \,|\, \cdot)$ is the ground truth choice model. This is a reasonable assumption if the number of transaction records for each assortment is large enough that each v_{o,S_m} will be close to the true choice probability $\mathbf{P}(o \,|\, S_m)$. Later, in Section 3.3.4, we will discuss how our estimation methodology can be readily adapted to the setting where the v_{o,S_m} values are derived from limited data.

We now define the estimation problem. For now, let us assume that we have fixed a collection of candidate trees F. For each tree $t \in F$, let us define $A_{t,(o,S)}$ to be 1 if tree t chooses option o when offered assortment S, and 0 otherwise. Let us also define $\mathbf{A}_{t,S}$ to be the vector of $A_{t,(o,S)}$ values for $o \in \mathcal{N}_+$ with a given assortment S. Let $\boldsymbol{\lambda} = (\lambda_t)_{t \in F}$ be the probability distribution over F. With these definitions, to find the probability distribution for the decision forest model, we must find a vector $\boldsymbol{\lambda}$ that satisfies the following system of constraints:

$$\sum_{t \in F} \mathbf{A}_{t,S} \lambda_t = \mathbf{v}_S, \quad \forall\ S \in \mathcal{S}, \tag{3.3a}$$

$$\mathbf{1}^T \boldsymbol{\lambda} = 1, \tag{3.3b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{3.3c}$$

In the above constraint system, constraints (3.3b) and (3.3c) model the requirement that $\boldsymbol{\lambda}$ be a probability distribution, while constraint (3.3a) requires that for each assortment S in the data, the vector of predicted choice probabilities, $\sum_{t \in F} \mathbf{A}_{t,S} \lambda_t$, is equal to the vector of actual choice probabilities, $\mathbf{v}_S$. Thus, if we could select a reasonable set of candidate trees for our decision forest, then we could, at least in theory, solve the feasibility problem (3.3) to obtain the corresponding probability distribution $\boldsymbol{\lambda}$.

Notwithstanding any computational questions surrounding problem (3.3), the remaining question is how one should choose the forest F of candidate trees. According to Theorem 1, decision forest models that are defined with $F = F_{N+1}$, where F_{N+1} is the set of trees of depth at most $N + 1$, are sufficient to represent any choice model, whether it belongs to the

RUM class or not. Thus, an immediate choice of F is F_{N+1}, and we would simply solve the feasibility problem (3.3) with F_{N+1} to obtain the corresponding probability distribution $\boldsymbol{\lambda}$. However, upon closer examination, this particular choice of F is problematic. The flexibility of the F_{N+1} decision forest model that we established in Theorem 1 implies that, without any additional structure, it is impossible to learn this model from data. Specifically, a consequence of Theorem 1 is that there always exists a distribution and a set of trees of depth at most $N+1$ such that (i) the model perfectly fits the training data $\{(S, \mathbf{v}_S)\}_{S\in\mathcal{S}}$, and (ii) the model also perfectly fits *any other possible choice probabilities on the assortments outside of the training data*. For example, there exists a forest model that is consistent with the training data $\{(S, \mathbf{v}_S)\}_{S\in\mathcal{S}}$, but always chooses the no-purchase option for every other assortment, i.e., $\mathbf{P}(0 \mid S) = 1$ for any $S \notin \mathcal{S}$.

This challenge with estimating the decision forest model motivates the need to impose some form of structure on the set F of candidate trees that may be used in the decision forest model. While there are many ways to quantify the size or complexity of a tree, we will primarily focus on two measures: (i) depth and (ii) number of leaves. Both of these measures are commonly applied in tree-based models found in machine learning. For example, the method of limiting the depth of decision trees has been widely used in machine learning algorithms, such as in CART [22], to avoid overfitting. Similarly, limiting the number of leaves can also prevent overfitting and has been adapted in tree boosting methods [31]. Both depth and number of leaves are closely linked to model complexity: intuitively, as the purchase decision trees in the forest become deeper or have more leaves (which is equivalent to having more splits), each tree is able to exhibit a wider range of behavior as the assortment varies.

There are three advantages to estimating decision forest models consisting of simple trees:

1. **Generalization**. Given two decision forest models that perfectly fit a set of training assortments, it is reasonable to expect that the decision forest that is simpler will be more likely to yield good predictions on new assortments outside of the training set.

2. **Tractability**. It is also reasonable to expect that the estimation problem will become more tractable, as the set of possible trees will be much smaller than the set of all possible trees of depth $N+1$ as required by Theorem 1.

3. **Behavioral plausibility**. Lastly, forests of simple trees are more behaviorally plausible than trees of depth $N+1$. As discussed in [68], customers often make purchase decisions by first forming a consideration set (a small set of products out of the whole assortment) and then choosing from among the considered products. Restricting the depth or limiting the number of leaves of each tree implies that customers only check for a small collection of products before making their purchase decision, and is congruent with empirical research on how customers choose.

Before presenting the results, we require some additional definitions. We define the *size* of a forest F as $|F|$, the number of trees in the forest. We define the *size* of a forest F as $|F|$, the number of trees in the forest; the *depth* of a forest F as $\max_{t \in F} \text{Depth}(t)$, the maximal depth of any tree in the forest F; and the *leaf complexity* of a forest F as $\max_{t \in F} |\mathbf{leaves}(t)|$, the maximal number of leaves of any tree in the forest F.

3.2.2 Forests of Simple Tree are Sufficient to Fit Data

Previously, we motivated the estimation of forests comprised of simple trees, i.e., trees whose depth is bounded by some value d or trees whose number of leaves is bounded by some value L. However, selecting the right depth d and L is not straightforward. While Theorem 1 guarantees the existence of a forest of depth $N+1$ that is consistent with the training data $\{(S, \mathbf{v}_S)\}_{S \in \mathcal{S}}$, it is not clear whether there exists a forest of depth $d \ll N+1$ that is consistent with the training data. Additionally, the trees guaranteed by Theorem 1 may have up to 2^N leaves.

In this section, we explore the relation between depth, leaf complexity, and size of a decision forest model to the number of historical assortments M. We propose two theoretical

results that provide guidance on how these complexity parameters may be selected.

Theorem 2 *For any training data $\mathcal{S}$ with M distinct historical assortments, $\mathcal{S} = \{S_1, \ldots, S_M\}$, there exists a probability distribution $\boldsymbol{\lambda}$ and a forest F of depth at most $\min\{N+1, M+1\}$, of leaf complexity at most $2M$, and of size at most $M(N+1)+1$ such that*

$$\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$$

for all $S \in \mathcal{S}$ and $o \in \mathcal{N}_+$.

The proof of Theorem 2 (see Section A.1.3) involves mathematical induction and polyhedral theory. In terms of depth, while Theorem 1 guarantees that we can fit the data with a forest of depth $N+1$, Theorem 2 ensures that we can fit the data with a forest of depth at most $\min\{N+1, M+1\}$. This result is particularly attractive when $M < N$. For example, if a seller has only offered 5 historical assortments over 20 products, then instead of building a decision forest of depth 21 as in Theorem 1, the seller can fit the customer behavior in the data by a forest of depth 6. In terms of leaf complexity, while Theorem 1 implicitly bounds the leaf complexity by 2^N, Theorem 2 guarantees that the complexity that scales only linearly in M. This result is also attractive because in practice M is unlikely to scale exponentially with respect to N and thus $M \ll 2^N$. Finally, in term of size, Theorem 2 guarantees that number of trees in the decision forest scales as $O(NM)$. We note that [52] established a similar size result for ranking-based models, showing that there exists a worst-case distribution over the set of all rankings that is consistent with the data and that has at most $K+1$ non-zero components, where K is the number of item-assortment pairs (see the proof of Theorem 1 of that paper); our result here about forest size can be viewed as a generalization of that result to the decision forest model.

In the case that the number of products N and the number of assortments M are both large, then the forests furnished by Theorem 2 will be very deep. A natural question is whether it is possible to do better than $\min\{N+1, M+1\}$ in this setting. To address model

complexity when both N and M are sufficiently large, we propose our second theoretical result, which is formalized below as Theorem 3. This theorem assumes a simple generative model of how historical assortments are chosen and establishes that, with high probability over the historical assortments, one can fit the data with a decision forest whose depth scales logarithmically in M.

Theorem 3 *Assume the M assortments $\mathcal{S} = \{S_1, S_2, S_3, \ldots, S_M\}$ of N products are drawn uniformly at random and independently from the set of all 2^N possible assortments. With probability at least $1 - O\left(M^2 \cdot 2^{-CN/\log_2 M}\right)$, there exists a distribution $\boldsymbol{\lambda}$ and a forest F of depth $O\left(\log_2 M\right)$ such that $\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$ for all $S \in \mathcal{S}$ and $o \in \mathcal{N}_+$, where $C > 0$ is a positive constant.*

Theorem 3 provides an asymptotic lower bound on the probability of the event that there exists a forest of depth logarithmic in M that can perfectly fit the training data, where the randomness is over the draw of M assortments from the set of all 2^N assortments. Note that the inequality $N \geq \log_2 M$ always holds, since one will have at most 2^N assortments for N products. On the other hand, in real-world data, M is unlikely to scale exponentially with respect to N; for example, a retailer offering 1000 products is unlikely to have offered $2^{1000} \approx 10^{300}$ subsets of those products in the past. Thus, when N is large and M does not scale exponentially with respect to N, the factor $N/\log_2 M$ makes the probability lower bound very close to 1. Stated differently, when N is large and M is not too large, most data sets – that is, most collections of assortments $\mathcal{S}$ of M assortments of the N products – will admit a forest representation that has depth $O(\log_2 M)$. We note that the result is completely independent of the choice probabilities: the result holds no matter what $(\mathbf{v}_S)_{S \in \mathcal{S}}$ is.

To prove Theorem 3, we prove an intermediate result, Theorem 18 (see Section A.1.4), which provides an explicit upper bound on the probability of not being able to find a forest of a specific choice of depth that is $O(\log_2 M)$ that fits the data. To give a sense of the scale

of the probability bound, for a retailer with $N = 10000$ products and $M = 2000$ historical assortments, the bound implies that the probability that the data set cannot be fit by a decision forest of depth at most 33 is no greater than 6.2×10^{-12}. In contrast, Theorem 1 and Theorem 2 yield decision forests of depths 10001 and 2001 respectively.

3.3 Estimating Decision Forest Model from Data

In this section, we describe two methods to estimate the decision forest model from data, based on column generation (Section 3.3.1) and randomized tree sampling (Section 3.3.2). In Section 3.3.3, we discuss two practical strategies for addressing overfitting. Lastly, in Section 3.3.4, we discuss how our methods can be extended to other forms of data and other types of objectives.

3.3.1 Method #1: Column Generation

Suppose for now that we select a large collection F of candidate trees. As discussed earlier, we wish to find a probability distribution $\boldsymbol{\lambda}$ over F that satisfies the constraint system (3.3) for F. If we specify the set of candidate trees F according to the depth or leaf complexity given in Theorem 2, then we are guaranteed the existence of a probability distribution $\boldsymbol{\lambda}$ that satisfies the constraint system (3.3). However, the collection of trees F may still be large enough that directly solving the feasibility problem (3.3) with F is computationally unwieldy. More importantly, if we specify F to consist of trees that are simpler (have a lower depth or fewer leaves) than those prescribed in Theorem 2, then it may not be possible to find a $\boldsymbol{\lambda}$ that exactly satisfies (3.3).

Thus, we will instead focus on finding a $\boldsymbol{\lambda}$ for which $\hat{\mathbf{v}}_S = \sum_{t \in F} \mathbf{A}_{t,S} \lambda_t$, the vector of predicted choice probabilities for the assortment S, is close to $\mathbf{v}_S$, the vector of actual choice probabilities for S, for all $S \in \mathcal{S}$. One approach to finding such a $\boldsymbol{\lambda}$ is to formulate an optimization problem where the objective is to minimize the average L_1 norm of the

prediction errors in the choice probabilities over all historical assortments:

$$\underset{\boldsymbol{\lambda},\hat{\mathbf{v}}}{\text{minimize}} \quad \frac{1}{|\mathcal{S}|} \cdot \sum_{S \in \mathcal{S}} \|\hat{\mathbf{v}}_S - \mathbf{v}_S\|_1 \tag{3.4a}$$

$$\text{subject to} \quad \hat{\mathbf{v}}_S = \sum_{t \in F} \mathbf{A}_{t,S} \lambda_t, \quad \forall\ S \in \mathcal{S}, \tag{3.4b}$$

$$\mathbf{1}^T \boldsymbol{\lambda} = 1, \tag{3.4c}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{3.4d}$$

By introducing additional variables $\boldsymbol{\epsilon}_S^+$ and $\boldsymbol{\epsilon}_S^-$ for each assortment $S \in \mathcal{S}$, we can reformulate problem (3.4) as a linear optimization problem. For a given data set $\mathcal{S}$ and forest F, we refer to this problem as EstLO, which we define below:

$$\text{EstLO}(\mathcal{S}, F) = \underset{\boldsymbol{\lambda},\boldsymbol{\epsilon}^+,\boldsymbol{\epsilon}^-}{\text{minimize}} \quad \frac{1}{|\mathcal{S}|} \cdot \left(\sum_{S \in \mathcal{S}} \mathbf{1}^T \boldsymbol{\epsilon}_S^+ + \sum_{S \in \mathcal{S}} \mathbf{1}^T \boldsymbol{\epsilon}_S^- \right) \tag{3.5a}$$

$$\text{subject to} \quad \sum_{t \in F} \mathbf{A}_{t,S} \lambda_t + \boldsymbol{\epsilon}_S^- - \boldsymbol{\epsilon}_S^+ = \mathbf{v}_S, \quad \forall\ S \in \mathcal{S}, \tag{3.5b}$$

$$\mathbf{1}^T \boldsymbol{\lambda} = 1, \tag{3.5c}$$

$$\boldsymbol{\epsilon}_S^+, \boldsymbol{\epsilon}_S^- \geq \mathbf{0}, \quad \forall\ S \in \mathcal{S}, \tag{3.5d}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{3.5e}$$

Before presenting our algorithm for solving this problem, we pause to comment on problem (3.5). Problem (3.5) is similar to the estimation problem that arises for ranking-based models. In particular, [119] study a maximum likelihood estimation problem, while [87] studies a similar L_1 estimation problem, both of which are formulated in a similar way to problem (3.5). Both [87] and [119] study solution methods for this general type of problem that are based on column generation, where one alternates between solving a master problem like (3.5) for a fixed set of rankings, and solving a subproblem to obtain the new ranking that should be added to the collection of rankings. In a different direction, the conditional gradient approach of [73] also involves iteratively adding rankings to a ranking-based model, which also involves solving a similar subproblem.

In the same way, one can also apply a column generation strategy to solve the decision forest estimation problem (3.5), which we now describe at a high level. For a fixed forest $\hat{F}$, we solve the problem $\textsc{EstLO}(\mathcal{S}, \hat{F})$ to obtain the primal solution $\boldsymbol{\lambda}$ and the dual solution $(\boldsymbol{\alpha}, \nu)$, where $\boldsymbol{\alpha} = (\alpha_{o,S})_{o \in \mathcal{N}^+, S \in \mathcal{S}}$ is the dual variable corresponding to constraint (3.5b) and ν is the dual variable corresponding to the unit sum constraint (3.5c). We then solve a subproblem to identify the tree in F with the lowest reduced cost:

$$\min_{t \in F} \left[-\sum_{S \in \mathcal{S}} \boldsymbol{\alpha}_S^T \mathbf{A}_{t,S} - \nu \right]. \tag{3.6}$$

If the lowest reduced cost is nonnegative, we terminate with $\boldsymbol{\lambda}$ as the optimal solution. (Note that $\boldsymbol{\lambda}$ is an optimal solution to $\textsc{EstLO}(\mathcal{S}, \hat{F})$; by setting $\lambda_t = 0$ for all $t \in F \setminus \hat{F}$, $\boldsymbol{\lambda}$ can be extended to be an optimal solution of $\textsc{EstLO}(\mathcal{S}, F)$.) If the reduced cost is negative, then we add the tree to $\hat{F}$, solve the problem again, and repeat the procedure until the reduced cost becomes nonnegative. The steps of this approach are summarized in Algorithm 1.

Algorithm 1 Column generation method for solving Problem (3.4).

procedure ColumnGeneration(F)
  Initialize $\hat{F} \leftarrow \emptyset$
  repeat
    Solve $\textsc{EstLO}(\mathcal{S}, \hat{F})$ to obtain $\boldsymbol{\lambda}$, $\boldsymbol{\alpha}$, ν
    Set $t^* \leftarrow \arg\min_{t \in F} \left[-\sum_{S \in \mathcal{S}} \boldsymbol{\alpha}_S^T \mathbf{A}_{t,S} - \nu \right]$
    Set $\hat{F} \leftarrow \hat{F} \cup \{t^*\}$
  until $-\sum_{S \in \mathcal{S}} \boldsymbol{\alpha}_S^T \mathbf{A}_{t^*,S} - \nu \geq 0$
  return $(\hat{F}, \hat{\boldsymbol{\lambda}})$

The key difference in the column generation approach for decision forests compared to column generation approaches for ranking-based models is the subproblem (3.6): rather than optimizing over the set of all rankings of the $N + 1$ options, one must optimize over a collection of trees. This subproblem can be formulated exactly as an integer optimization problem, with a structure that is different from the integer optimization problem that

arises in ranking-based models (as in [119]); we provide the details of the formulation in Section A.2.1. Although the resulting exact column generation approach is able to solve problem (3.5) to provable optimality, it is unfortunately not scalable; for example, for trees of depth $d = 4$, $N = 8$ products and $M = 50$ training assortments, the approach can require over 6 hours (see Section A.2.2 for detailed runtime results).

Motivated by the intractability of solving the subproblem (3.6) exactly, we consider an alternate strategy where we solve the subproblem heuristically. The heuristic procedure involves starting from a degenerate tree consisting of a single leaf, and then iteratively replacing each leaf with a split with two child leaf nodes. The leaf that is chosen for splitting, as well as the product that is placed on that leaf node and the purchase decisions for the two new leaves, are chosen in a greedy fashion, so as to result in the largest improvement in the reduced cost. The procedure terminates when the reduced cost can no longer be decreased. In addition, the procedure also grows each tree to a user-specified maximum depth of d; stated differently, a leaf cannot be considered for splitting when it reaches a depth of d.

We formally define our top-down induction heuristic as Algorithm 2. Within Algorithm 2, we use t_0 to denote a degenerate tree that consists of a single leaf node, whose purchase decision is the no-purchase option 0. We use $\textbf{leaves}(t, d)$ to denote the set of all leaves in the tree t that are at a depth up to (but not including) d. We define Z_{t,ℓ,p,o_1,o_2} as the reduced cost of the tree that is obtained by replacing leaf ℓ of tree t with a split, setting the product x_ℓ of that new split to the product p, and setting the left child leaf node's purchase decision to o_1 and the right child leaf node's purchase decision to o_2; we also use $\textsc{GrowTree}(t, \ell, p, o_1, o_2)$ to denote the tree that is obtained from growing tree t in this way. Lastly, we use $P(\ell)$ to denote the set of products that have appeared in the ancestral splits of leaf ℓ (i.e., the set of products p for which $x_s = p$ for some split s along the path from the root node to leaf ℓ). When choosing the product p to appear on the split at leaf ℓ, Algorithm 2 is restricted to using only those products that have not appeared in an ancestral split, i.e., those products in $\mathcal{N} \setminus P(\ell)$; this ensures that the trees generated by Algorithm 2 satisfy Requirement 3 in

Definition 6 (Section 3.1.1).

When we use the top-down induction heuristic (Algorithm 2) within the column generation method (Algorithm 1), we refer to the overall method as the *heuristic column generation* (HCG) method.

Algorithm 2 Top-down induction method for heuristically solving the subproblem (3.6).

procedure TopDownInduction($\boldsymbol{\alpha}$, ν, d)
  Initialize $t \leftarrow t_0$
  Initialize $Z_c \leftarrow \left[-\sum_{S \in \mathcal{S}} \boldsymbol{\alpha}_S^T \mathbf{A}_{t,S} - \nu\right]$
  while $|\mathbf{leaves}(t, d)| > 0$ **do**
    Compute Z_{t,ℓ,p,o_1,o_2} for all $\ell \in \mathbf{leaves}(t, d)$, $p \in \mathcal{N} \setminus P(\ell)$,
     $o_1 \in \{p, 0\} \cup \{x_s \mid s \in \mathbf{LS}(\ell)\}$, $o_2 \in \{0\} \cup \{x_s \mid s \in \mathbf{LS}(\ell)\}$
    Set $Z^* \leftarrow \min_{\ell,p,o_1,o_2} Z_{t,\ell,p,o_1,o_2}$
    Set $(\ell^*, p^*, o_1^*, o_2^*) \leftarrow \arg\min_{(\ell,p,o_1,o_2)} Z_{t,\ell,p,o_1,o_2}$
    if $Z^* < Z_c$ **then**
      Set $Z_c \leftarrow Z^*$
      Set $t \leftarrow$ GrowTree$(t, \ell^*, p^*, o_1^*, o_2^*)$
    else
      break
  return t, Z_c

We comment on three important aspects of our heuristic column generation method. First, our top-down induction procedure resembles greedy heuristics that are used for other tree models in the machine learning literature, such as CART [22], C4.5 [98] and ID3 [97]. In addition, such algorithms are also used in algorithms that build collections of trees. Within this literature, our heuristic column generation method most resembles boosting, wherein one adds trees (or other weak learners) iteratively to reduce the training error; see, for example, [31], [60], and [61].

Second, since our top-down induction heuristic considers trees of maximum depth d,

the overall column generation approach – Algorithm 1 combined with Algorithm 2 to solve the subproblem – effectively solves the problem $\textsc{EstLO}(\mathcal{S}, F_d)$, where F_d is the set of unbalanced trees of depth at most d. We note that the overall approach *heuristically* solves $\textsc{EstLO}(\mathcal{S}, F_d)$; it does not guarantee that the resulting solution is an optimal solution of $\textsc{EstLO}(\mathcal{S}, F_d)$. However, we find that the approach performs well in practice. In Section A.2.2 we numerically compare the heuristic column generation approach against the exact approach; we find that the heuristic approach obtains optimal or near-optimal training error in a fraction of the time required by the exact approach.

Third, the main complexity control in Algorithm 2 is the limit imposed on the depth of the tree. As discussed in Section 3.2.2, one could use the number of leaves instead of the depth to control the complexity of the trees. We can thus consider a variant of Algorithm 2 wherein one terminates the induction procedure upon reaching a user-specified limit on the total number of leaves. We formally define this alternate method in Section A.2.3.

3.3.2 Method #2: Randomized Tree Sampling

In this section, we present our second estimation method, which we refer to as the *randomized tree sampling* (RTS) approach. In this approach, instead of sequentially adding trees to a growing collection, we directly sample a large number of trees to serve as the forest $\hat{F}$, and then solve an optimization problem to find the corresponding probability distribution $\boldsymbol{\lambda}$.

The overall procedure requires three inputs. The first input K is the number of trees to be sampled. The second input F is a base collection of trees that the algorithm will sample from, while the third input $\boldsymbol{\xi}$ is a probability distribution over F according to which we will draw our sample of K trees. We formally define the method as Algorithm 3.

We theoretically characterize how the distribution $\boldsymbol{\xi}$ and the sample size K affect the performance of Algorithm 3 as follows. We first define the training error or *empirical risk* of

Algorithm 3 Randomized tree sampling method for solving Problem (3.4).

procedure RANDOMIZEDTREESAMPLING(K, F, $\boldsymbol{\xi}$)
  Draw K trees $t_1, t_2, \ldots, t_K$ from F according to distribution $\boldsymbol{\xi}$
  Set $\hat{F} \leftarrow \{t_1, t_2, \ldots, t_K\}$
  Solve ESTLO($\mathcal{S}, \hat{F}$) to obtain probability distribution $\hat{\boldsymbol{\lambda}}$
  return $(\hat{F}, \hat{\boldsymbol{\lambda}})$

a decision forest model $(F, \boldsymbol{\lambda})$ with respect to the data $\{(S, \mathbf{v}_S)\}_{S \in \mathcal{S}}$ as

$$\mathbf{R}(F, \boldsymbol{\lambda}) \equiv \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} \left\| \sum_{t \in F} \mathbf{A}_{t,S} \lambda_t - \mathbf{v}_S \right\|_1 . \tag{3.7}$$

Our main theoretical result (Theorem 4) states that with high probability, the empirical risk of the model returned by Algorithm 3 converges to the lowest risk attainable by any forest model in a set $\Lambda(C, \boldsymbol{\xi})$, which will be defined in Theorem 4, with rate $1/\sqrt{K}$.

Theorem 4 *Let F be any collection of trees, let $\boldsymbol{\xi}$ be a probability distribution over F such that $\xi_t > 0$ for all $t \in F$, and let $C > 1$ be a constant. Define the set*

$$\Lambda(C, \boldsymbol{\xi}) \equiv \left\{ \boldsymbol{\lambda} \in \mathbb{R}^{|F|} \mid \lambda_t \leq C \cdot \xi_t, \ \forall t \in F; \ \mathbf{1}^T \boldsymbol{\lambda} = 1; \ \boldsymbol{\lambda} \geq \mathbf{0} \right\} \tag{3.8}$$

as a collection of probability distributions over F. Then for any $\delta > 0$, Algorithm 3 returns a forest model $(\hat{F}, \hat{\boldsymbol{\lambda}})$ such that its empirical risk $\mathbf{R}(\hat{F}, \hat{\boldsymbol{\lambda}})$ satisfies

$$\mathbf{R}(\hat{F}, \hat{\boldsymbol{\lambda}}) \leq \min_{\boldsymbol{\lambda} \in \Lambda(C, \boldsymbol{\xi})} \mathbf{R}(F, \boldsymbol{\lambda}) + \frac{C}{\sqrt{K}} \cdot \left(\sqrt{N+1} + 3\sqrt{\log(4/\delta)} \right)$$

with probability at least $1 - \delta$ over the sample of trees $t_1, \ldots, t_K$ that comprise $\hat{F}$.

In words, the training error (i.e., the objective value of problem (3.4)) of the decision forest model $(\hat{F}, \hat{\boldsymbol{\lambda}})$ is bounded with high probability by the sum of two terms, where the first term measures the best possible training error over decision forest models $(F, \boldsymbol{\lambda})$ where $\boldsymbol{\lambda}$ is in $\Lambda(C, \boldsymbol{\xi})$, while the second term depends linearly on C. When C is large, the first

term will be small because the set $\Lambda(C, \boldsymbol{\xi})$ will be larger, but the second term will be large. Similarly, when C is small, the second term will be reduced, but the first term will become larger because the set $\Lambda(C, \boldsymbol{\xi})$ will shrink.

The set $\Lambda(C, \boldsymbol{\xi})$ reflects the "coverage" ability of the distribution $\boldsymbol{\xi}$. If the choice probabilities $(\mathbf{v}_S)_{S \in \mathcal{S}}$ can be generated by a decision forest model $(F, \boldsymbol{\lambda})$ for some $\boldsymbol{\lambda}$ from $\Lambda(C, \boldsymbol{\xi})$ corresponding to a small value C, then the number of trees that we need to sample in order to obtain a low training error $\mathbf{R}(\hat{F}, \hat{\boldsymbol{\lambda}})$ will be small. As an example, if $\boldsymbol{\xi}$ corresponds to the uniform distribution over F and if the optimal $\boldsymbol{\lambda}^*$ that fits the choice probabilities $(\mathbf{v}_S)_{S \in \mathcal{S}}$ is "close" to being uniform, then we only need to sample a small number of trees to achieve a low training error, because the implied value of C (i.e., the value of C needed for $\boldsymbol{\lambda}^*$ to be contained in $\Lambda(C, \boldsymbol{\xi})$) is small. As another example in contrast to the previous one, if the optimal $\boldsymbol{\lambda}^*$ is one where (for example) one tree t' has a disproportionately higher probability than the other trees, then we will need to sample many trees from $\boldsymbol{\xi}$ because the implied value of C is large; this makes sense intuitively because one has a low likelihood of sampling t' from $\boldsymbol{\xi}$ when $|F|$ is large. We also note that the effect of the structure of F in terms of the depth or the number of leaves of the trees is captured in the term $\min_{\boldsymbol{\lambda} \in \Lambda(C, \boldsymbol{\xi})} \mathbf{R}(F, \boldsymbol{\lambda})$. As F contains a richer collection of trees, this term will in general become smaller.

We note that Theorem 4 is inspired by the literature on randomization in machine learning – specifically, the idea of training weighted combinations of (nonlinear) features by randomly sampling the features [91, 99, 100]. Indeed, our proof of Theorem 4 adapts the technique in [100], which considers the problem of learning arbitrary weighted sums of feature functions, to the problem of learning a probability distribution (in the setup of [100], the weights need not add up to one). In choice modeling, random sampling was previously used in [52]. In that paper one formulates the problem of finding the worst-case probability distribution over a collection of rankings, which is a linear optimization problem of a similar form to our estimation problem $\textsc{EstLO}(\mathcal{S}, F)$. To solve this worst-case problem, one formulates the dual and randomly samples a collection of constraints (i.e., rankings).

The paper of [52] justifies this by appealing to the paper of [26], which shows that with $O((1/\epsilon)(NM\ln(1/\epsilon)+\ln(1/\delta))$ constraints being sampled, at most an ϵ fraction of the constraints will be violated, with probability at least $1-\delta$ over the sampling. However, as noted in [52], the theory of [26] does not govern how far the optimal objective of the sampled problem will be from the complete problem, which is the focus of our result here.

3.3.3 Addressing Overfitting

Given the richness of the decision forest model, an important concern is overfitting. In this section, we describe two practical strategies for addressing overfitting in the decision forest model.

k-fold cross-validation: As in other machine learning methods, one can use k-fold cross validation to tune the hyperparameters for the decision forest model. In this approach, we divide the training set into a collection of k smaller subsets or *folds*. For a fixed value of a hyperparameter, we use the $k-1$ folds as training data to estimate the model with that hyperparameter value, and evaluate the model's performance on the remaining hold-out fold; we repeat this k times, with each of the k folds serving as the hold-out fold, and average over the k folds. We then repeat this for each value of interest for the hyperparameter, and choose the best value. This approach can be used to set the depth limit d for the top-down induction method (Algorithm 2) within HCG. This approach can also be used to select an appropriate collection of trees F and probability distribution $\boldsymbol{\xi}$ for the randomized tree sampling method; a simple implementation of this idea is to specify F as the set of all balanced trees of depth d that satisfy Requirement 1-3 in Definition 6 (Section 3.1.1), specify $\boldsymbol{\xi}$ as the uniform distribution over F, and use k-fold cross-validation to determine the optimal depth d. In our numerical experiments in Section 3.4.2, we use k-fold cross-validation to tune the depth d for the HCG and RTS approaches.

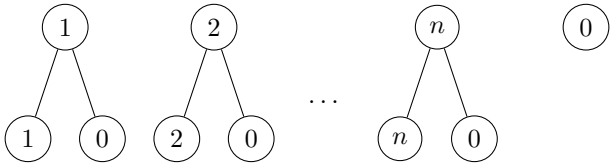


Figure 3.8: Collection of trees F_0 corresponding to an independent demand model.

Warm-starts: Both the heuristic column generation method and the randomized tree sampling method build the collection of trees $\hat{F}$ from scratch, without any set of trees explicitly provided by the user. However, they can be easily modified to take an initial set of trees F_0 as an input: in Algorithm 1, we can modify line 2 so that we initialize $\hat{F} \leftarrow F_0$, while in Algorithm 3, we can modify line 3 to set $\hat{F} \leftarrow \{t_1, \ldots, t_K\} \cup F_0$. With regard to F_0, the simplest choice is the independent demand model, which corresponds to the forest shown in Figure 3.8. Another natural choice for F_0 is the set of trees that correspond to a ranking-based model learned by another method (such as [87] or [119]). By warm-starting either Algorithm 1 or Algorithm 3 in this way, one can bias the estimation so that the resulting decision forest model is close to the best-fitting ranking-based model, and reduce the possibility of overfitting in cases where the customer choice behavior is close to a rational model.

3.3.4 Estimating Decision Forests with Log-Likelihood Objective

So far, we have assumed that the choice probabilities $\mathbf{v} = (v_{o,S})_{o \in \mathcal{N}_+, S \in \mathcal{S}}$ for a set of historical assortments $\mathcal{S}$ is known. Our goal has thus been to minimize the error between $\mathbf{v}$ and $\hat{\mathbf{v}}$, the choice probabilities predicted by a decision forest model, and we have measured error using the L_1 norm. In practice, when the number of transactions is sufficiently large for each assortment S, then the frequency of each observed option o given assortment S can serve as an ideal value for $v_{o,S}$.

In other real-world settings, transaction records may be abundant for some assortments but scarce for others. A more common objective function for this finite sample setting

is log-likelihood. Let $c(o, S)$ be the number of transactions in which o was chosen given assortment S. The maximum likelihood problem can be represented as the following concave optimization problem:

$$\underset{\boldsymbol{\lambda}, \hat{\mathbf{v}}}{\text{maximize}} \quad \sum_{S \in \mathcal{S}} \sum_{o \in \mathcal{N}_+} c(o, S) \cdot \log \hat{v}_{o,S} \tag{3.9a}$$

$$\text{subject to} \quad \hat{\mathbf{v}}_S = \sum_{t \in F} \mathbf{A}_{t,S} \lambda_t, \quad \forall\ S \in \mathcal{S}, \tag{3.9b}$$

$$\mathbf{1}^T \boldsymbol{\lambda} = 1, \tag{3.9c}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \tag{3.9d}$$

where the objective is the log-likelihood of the transaction records and $\hat{v}_{o,S}$ is the choice probability of the forest model for option o given assortment S. Problem (3.9) only differs from problem (3.4) in the objective function. Note that when each column $(\mathbf{A}_{t,S})_{S \in \mathcal{S}}$ corresponds to a ranking, then problem (3.9) coincides with the maximum likelihood problem that is solved in [119]. The paper of [119] solves this problem using column generation, and shows how one can obtain the dual variables for constraints (3.9b) and (3.9c) in closed form. In addition, the paper of [121] proposes a specialized expectation maximization (EM) method for solving the ranking-based maximum likelihood problem, without invoking a nonlinear optimization solver. It turns out that for the forest maximum likelihood problem (3.9), the dual variables can be obtained in the same way as in [119] and the problem itself can be solved with the same EM algorithm from [121]. We thus adapt the heuristic column generation and randomized tree sampling methods as follows. For the heuristic column generation, we solve the restricted master problem at each iteration using the EM algorithm of [121] and solve the subproblem using our top-down induction method, with the dual variables obtained as in [119]. For the randomized tree sampling algorithm, instead of solving $\textsc{EstLO}(\mathcal{S}, \hat{F})$ with a sampled collection of trees $\hat{F}$, we solve problem (3.9) with $\hat{F}$ using the EM algorithm of [121].

3.4 Decision Forest Model on a Real-World Dataset

In this section, we apply our decision forest model to the IRI Academic Dataset [23] and evaluate its predictive performance.

3.4.1 Background

The IRI dataset is comprised of real-world transaction records of store sales and consumer panels for thirty product categories, and includes sales information for products collected from 47 U.S. markets. The purpose of these experiments is to show how the decision forest model can lead to better predictions of real-world customer choices. We note that the same data set was used in [73] to empirically demonstrate the loss of rationality in real customer purchase data.

To pre-process the data, we follow the same pre-processing steps as in [73]. In the dataset, each item is labeled with its respective universal product code (UPC). By aggregating the items with the same vendor code (denoted by digits four through eight of the UPC) as a product, we can identify products from the raw transactions; we note that this is a common pre-processing technique (see [24, 92]). By selecting the top nine purchased products and combining the remaining products as the no-purchase option, we create transaction records for the model setup. Due to the large number of transactions, we follow [73] by only focusing on data from the first two weeks of calendar year 2007.

After pre-processing the data, we convert the sales transactions for each product category into assortment-choice pairs $\{(S_t, o_t)\}_{t\in\mathcal{T}}$, where $\mathcal{T}$ is a collection of transactions, as follows. Each transaction t contains the following information: the week of the purchase (w_t), the store ID where the purchase was recorded (z_t), the UPC of the purchased product (p_t). Let $\mathcal{W}$ and $\mathcal{Z}$ be the non-repeated collection of $\{w_t\}_{t\in\mathcal{T}}$ and $\{z_t\}_{t\in\mathcal{T}}$, respectively. With week $w \in \mathcal{W}$ and store $z \in \mathcal{Z}$, we define the offer set $S_{w,z} = \left\{\bigcup_{t\in\mathcal{T}}\{p_t \mid w_t = w, z_t = z\}\right\} \cup \{0\}$, as the collection of the products as well as the no-purchase option, purchased at least once

at store z in week w. As in Section 3.3.4, we define $c(o, S)$ as the purchase count for option o given assortment S, i.e., $c(o, S) = \sum_{t \in \mathcal{T}} \mathbb{I}\{S_t = S, o_t = o\}$.

To quantify the out-of-sample performance of each predictive model on testing transaction set $\mathcal{T}_{\text{test}}$, we use Kullback-Leibler (KL) divergence per transaction, which is defined as

$$\text{KL}(\mathcal{T}_{\text{test}}) = -\frac{1}{|\mathcal{T}_{\text{test}}|} \sum_{S \in \mathcal{S}(\mathcal{T}_{\text{test}})} \sum_{o \in \mathcal{N}_+} c(o, S, \mathcal{T}_{\text{test}}) \log \left(p_{o,S} / v_{o,S}(\mathcal{T}_{\text{test}}) \right),$$

where $\mathcal{S}(\mathcal{T}_{\text{test}})$ is the set of assortments found in $\mathcal{T}_{\text{test}}$, $c(o, S, \mathcal{T}_{\text{test}})$ is the number of purchases of option o given assortment S observed in $\mathcal{T}_{\text{test}}$, $p_{o,S}$ is the predicted choice probability for option o given assortment S, and $v_{o,S}(\mathcal{T}_{\text{test}})$ is the empirical choice probability for option o given assortment S derived from the transaction set $\mathcal{T}_{\text{test}}$. Specifically, $v_{o,S}(\mathcal{T}_{\text{test}}) = c(o, S, \mathcal{T}_{\text{test}}) / \sum_{o' \in \mathcal{N}_+} c(o', S, \mathcal{T}_{\text{test}})$. We remark that [73] also used KL divergence as a measure of goodness of fit. While their work focused on the *in-sample* information loss from fitting any RUM model, our numerical experiments here emphasize *out-of-sample* predictive ability.

3.4.2 Experiment #1: Assortment Splitting

In our first experiment, we test the out-of-sample predictive ability of our models using five-fold cross validation, where the splitting is done with respect to assortments. We divide the set of assortments $\mathcal{S}$ into five (approximately) equally-sized subsets $\mathcal{S}_1, \ldots, \mathcal{S}_5$, and for each $i \in \{1, \ldots, 5\}$, we use the transaction data for assortments $\mathcal{S}_1, \ldots, \mathcal{S}_{i-1}, \mathcal{S}_{i+1}, \ldots, \mathcal{S}_5$ to build each predictive model and the remaining fold $\mathcal{S}_i$ is used for testing. We note that this is a more stringent test of the predictive performance of the models than the standard cross validation based on splitting the transactions, as each model is used to make predictions on assortments that are different from the assortments used to train the models.

In addition to the decision forest model, we test four other models: the ordinary (single-class) MNL model, the latent-class MNL (LC-MNL) model, the ranking-based model and the HALO-MNL model [82]. For both the MNL and the HALO-MNL models, we fit the

parameters using maximum likelihood estimation.

For the LC-MNL model, we implement the EM algorithm of [115]. We tune the number of classes K within the set $\{2, 3, 5, 10, 15\}$ using k-fold cross validation with $k = 4$, using the previously-defined folds $\mathcal{S}_1, \ldots, \mathcal{S}_5$. We emphasize here that this "inner" cross-validation, which involves four folds and is used for tuning the number of classes K, is distinct from the "outer" cross-validation, which involves five folds and is for the purpose of obtaining a reliable estimate of the out-of-sample KL divergence.

For the ranking-based model, we estimate the model using the column generation method of [119], where the master problem is solved using the EM algorithm in [121]. We define the parameter τ for this model as the maximum allowable consideration set size; in other words, any ranking must be such that there are no more than τ products that are more preferred to the no-purchase option. We tune the parameter τ within the set $\{2, 3, 4, 5, 6, 7, 8, 9\}$ using k-fold cross validation with $k = 4$, using the folds $\mathcal{S}_1, \ldots, \mathcal{S}_5$. We note that ranking-based models with constrained consideration sets have been considered in previous research on the ranking-based model (see [54]).

For the decision forest model, we estimate the model in two different ways. The first involves using the heuristic column generation method in Section 3.3.1 with log-likelihood as the objective function (as in Section 3.3.4). We solve the master problem using the same EM algorithm from [121]. We warm start the model by setting the initial set of trees F_0 to be the set of trees corresponding to the rankings estimated for the ranking-based model with $\tau = 9$ (note that since the number of products $N = 10$, this value corresponds to estimating ranking-based model without a constraint on the consideration set size). In the same way that we tune K for the LC-MNL model, we also tune the value of d, the maximum depth parameter of the top-down induction method (Algorithm 2). We tune d within the set $\{3, 4, 5, 6, 7\}$ using k-fold cross validation with $k = 4$, again using the folds $\mathcal{S}_1, \ldots, \mathcal{S}_5$ defined earlier.

The second approach for the decision forest model that we consider is the randomized

tree sampling method in Section 3.3.2, again with log-likelihood as the objective function. We find the optimal $\boldsymbol{\lambda}$ using the EM algorithm from [121], and as with the HCG method, we warm start the model by setting the initial set of trees F_0 to be the set of trees corresponding to the rankings estimated for the ranking-based model with $\tau = 9$. We set the base collection of trees F to be sampled as the set of all balanced trees of depth d, and the distribution $\boldsymbol{\xi}$ as the uniform distribution over F. We tune d within the set $\{3, 4, 5, 6, 7\}$ using k-fold cross-validation with $k = 4$. We fix the number of sampled trees K to 2000; for simplicity, we do not tune the value of K.

Table 3.4.2 summarizes the out-of-sample performance of each predictive model over the thirty product categories. The first three columns under "Datasets" show the product category, and the number of historical assortments and transactions in that category. The remaining columns report the average out-of-sample KL divergence over five folds. The best performing method in each category is indicated in bold.

Out of 30 product categories, the MNL model attains the lowest KL divergence in 1 category, the LC-MNL model attains the lowest in 4 categories, the HALO-MNL model in 8 categories, the ranking-based model in 1 category, and the decision forest model (using either HCG or RTS) in 16 categories. Comparing the decision forest using HCG to the three RUM models (the MNL, LC-MNL and ranking-based models), we find that the decision forest model leads to a lower out-of-sample KL divergence in 22 out of 30 categories. Similarly, the decision forest model using HCG also outperforms the HALO-MNL model in 22 out of 30 categories. In addition, the decision forest model (using either HCG or RTS) achieves lower average, median and maximum KL divergences over the thirty product categories than the other benchmark models. These results suggest the potential of the decision forest model to provide accurate predictions of choice probabilities on new, unseen assortments. For space consideration, we relegate the runtime and complexity results of each predictive model in Section A.3.1.

Product Category	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{T}\rvert$	MNL	LC-MNL	HALO-MNL	RM	DF (HCG)	DF (RTS)
Beer	55	380,932	6.43	5.68	**0.79**	5.49	0.88	1.56
Blades	57	92,404	0.48	**0.40**	0.52	1.44	0.41	1.02
Carbonated Beverages	31	721,506	2.85	2.54	**0.95**	2.65	1.56	1.56
Cigarettes	68	249,668	1.91	1.67	**0.91**	1.65	0.98	0.96
Coffee	47	372,536	2.99	2.03	2.11	2.03	1.96	**1.64**
Cold Cereal	15	577,236	1.73	1.79	**0.58**	2.10	0.90	0.69
Deodorant	45	271,286	0.61	0.73	0.82	0.83	**0.42**	0.68
Diapers	18	143,055	3.34	1.54	58.56	7.13	**1.07**	1.51
Facial Tissue	43	73,806	1.39	1.09	1.47	1.21	**0.77**	1.32
Frozen Dinners	30	979,936	1.44	0.95	3.84	**0.94**	2.40	1.98
Frozen Pizza	61	292,878	2.76	2.13	**1.04**	2.10	1.10	1.13
Hotdogs	100	101,624	3.52	3.22	**2.81**	3.17	2.97	2.92
Household Cleaners	19	282,981	0.94	0.93	1.61	0.96	0.68	**0.51**
Laundry Detergent	56	238,163	2.37	2.30	2.29	2.39	**2.13**	2.33
Margarine/Butter	18	140,969	2.21	2.06	1.68	2.04	1.19	**0.74**
Mayonnaise	48	97,282	1.33	0.94	0.93	0.90	**0.84**	0.89
Milk	49	240,691	4.22	3.63	1.59	2.78	**1.29**	1.45
Mustard/Ketchup	44	134,800	1.32	1.06	0.78	1.10	**0.74**	0.80
Paper Towels	40	82,636	1.21	**1.09**	1.42	1.17	1.09	1.10
Peanut Butter	51	108,770	2.05	1.52	1.86	1.66	**1.49**	1.51
Photo	80	17,047	0.84	**0.76**	4.66	3.33	1.31	1.28
Salty Snacks	39	736,148	1.87	1.74	2.09	1.79	1.77	**1.70**
Shampoo	66	290,429	1.17	1.37	**0.86**	1.34	0.95	1.21
Soup	24	905,541	1.19	1.17	2.86	1.04	**0.95**	1.63
Spaghetti/Italian Sauce	38	276,860	3.38	3.26	4.44	2.89	3.37	**2.88**
Sugar Substitutes	64	53,834	0.83	**0.76**	0.79	0.93	0.77	0.88
Toilet Tissue	27	112,788	**1.42**	1.49	2.09	1.79	1.47	1.86
Toothbrush	114	197,676	1.53	1.28	**0.60**	1.23	0.99	1.19
Toothpaste	42	238,271	0.53	0.55	0.37	0.64	**0.35**	0.39
Yogurt	43	499,203	4.71	4.38	4.07	3.16	2.80	**1.58**
(Mean)	–	–	2.09	1.80	3.65	2.06	**1.32**	1.36
(Median)	–	–	1.63	1.50	1.53	1.72	**1.08**	1.30
(Maximum)	–	–	6.43	5.68	58.56	7.13	3.37	**2.92**

Table 3.2: Out-of-sample KL divergence (in units of 10^{-2}) for each model over the thirty product categories in the IRI data set, under assortment-based splitting.

3.4.3 Experiment #2: Temporal Splitting

In addition to the assortment-based splitting schemes, we consider an additional splitting scheme that we term *temporal splitting*. In this experimental approach, we use the first two weeks of transactions in 2007 in the IRI data set as training data, and then use the following four weeks as test data; this approach emulates how one would use the predictive models to make predictions prospectively (i.e., for transactions occurring in the future). We note that this type of splitting approach has been used previously in the literature; see, for example, [3] and [6].

We compare the decision forest model (estimated using both the HCG and RTS approaches), the single-class MNL model, the LC-MNL model, the ranking-based model and the HALO-MNL model. We use five-fold cross-validation to tune the models, where the folds correspond to the five assortment folds used in Section 3.4.2. We use this cross-validation to tune the values of the following hyperparameters: the depth limit d for the HCG approach for the decision forest model; the depth d of the base forest for the RTS approach for the decision forest model; the number of classes K for the latent-class MNL model; and the consideration set size τ for the ranking-based model. We estimate all of the models using the same methods as in Section 3.4.2. As in the earlier experiments, we set the number of sampled trees K for the RTS method to 2000. Note that unlike the transaction-splitting experiment in Section 3.4.2, there is only one form of cross-validation done in this experiment, which is to tune the hyperparameters; the out-of-sample performance of the final model of each class is then evaluated using the testing data for weeks 3 to 6, without any further cross-validation.

Table 3.3 shows the out-of-sample KL divergence for each of the five models, on each of the 30 product categories. In addition, the table also summarizes the number of transactions and unique assortments in the training data ($|\mathcal{T}_{1:2}|$ and $|\mathcal{S}_{1:2}|$, respectively, where the subscript 1:2 indicates weeks 1 to 2); the number of transactions and unique assortments in the test

data ($|\mathcal{T}_{3:6}|$ and $|\mathcal{S}_{3:6}|$, respectively, where the subscript 3:6 indicates weeks 3 to 6); and lastly, how many new assortments exist in the test data (i.e., how many assortments in the test data are not present in the training data; this is indicated by $|\mathcal{S}_{3:6} \setminus \mathcal{S}_{1:2}|$).

From this table, we can see that the decision forest model and the HALO-MNL model are essentially tied for the best performance. The decision forest model, using either the HCG or RTS methods, delivers the lowest KL divergence out of all of the models on 14 out of 30 product categories, while the HALO-MNL provides the lowest KL divergence out of all the models on 16 out of 30 categories. At the same time, when comparing the average, median and maximum out-of-sample KL divergence over the 30 product categories, we can see that the decision forest model achieves the best performance, with the HALO-MNL model being very slightly higher. Overall, this experiment indicates the potential of the decision forest model to be used for making predictions prospectively.

3.4.4 Extracting Substitution and Complementarity Behavior

In addition to obtaining predictions, choice modeling is useful for obtaining insights on the relationship between products, i.e., how the presence of one product will affect the choice probability of another product. For parametric choice models, such as the LC-MNL model, such insights can be easily obtained by examining the estimated utility parameters. In contrast, for nonparametric models such as the ranking-based model or the decision forest model, it is less straightforward to obtain a simple picture of the relationship between products.

In this section, we propose a simple method for extracting substitution and complementarity effects between products for a given choice model, and use it to analyze the decision forest model for a single product category in the IRI dataset. We note that our procedure is not specific to the decision forest model and can be used for other choice models (such as the ranking-based model, which we also analyze), and thus may be of independent interest.

Category	$\lvert\mathcal{S}_{1:2}\rvert$	$\lvert\mathcal{T}_{1:2}\rvert$	$\lvert\mathcal{S}_{3:6}\rvert$	$\lvert\mathcal{T}_{3:6}\rvert$	$\lvert\mathcal{S}_{3:6} \setminus \mathcal{S}_{1:2}\rvert$	MNL	HALO-MNL	LC-MNL	RM	DF (HCG)	DF (RTS)
Beer	55	380,932	57	772,629	11	0.83	0.10	0.66	0.66	**0.09**	0.21
Blades	57	92,404	74	183,906	23	0.42	**0.18**	0.26	0.34	0.29	0.35
Carbonated Beverages	31	721,506	32	1,452,900	1	1.57	0.25	1.25	1.18	0.29	**0.24**
Cigarettes	68	249,668	87	499,296	25	1.62	0.39	1.26	1.08	**0.31**	0.42
Coffee	47	372,536	50	759,294	10	2.22	0.54	1.23	1.14	**0.26**	0.45
Cold Cereal	15	577,236	17	1,163,581	2	1.41	0.09	1.34	1.18	**0.09**	0.20
Deodorant	45	271,286	66	543,687	36	0.37	**0.31**	0.36	0.35	0.33	0.37
Diapers	18	143,055	19	284,854	3	0.15	**0.06**	0.07	0.09	0.09	0.19
Facial Tissue	43	73,806	46	146,165	4	1.03	**0.29**	0.64	0.64	0.36	0.40
Frozen Dinners	30	979,936	29	1,972,097	3	0.46	**0.14**	0.36	0.32	0.18	0.26
Frozen Pizza	61	292,878	70	584,388	14	2.50	0.44	1.62	1.68	**0.35**	0.67
Hotdogs	100	101,624	112	203,573	23	2.83	1.32	2.13	1.89	**0.33**	0.84
Household Cleaners	19	282,981	28	561,774	13	0.07	**0.05**	0.07	0.06	0.05	0.06
Laundry Detergent	56	238,163	73	459,055	23	1.78	**0.66**	1.46	0.97	0.71	0.79
Margarine/Butter	18	140,969	19	283,612	5	1.18	**0.08**	0.70	0.70	0.08	0.20
Mayonnaise	48	97,282	49	194,607	7	1.19	0.33	0.59	0.56	**0.28**	0.44
Milk	49	240,691	56	472,009	9	3.38	**0.67**	1.87	1.86	0.71	0.72
Mustard/Ketchup	44	134,800	57	265,669	16	1.13	0.23	0.71	0.65	**0.12**	0.28
Paper Towels	40	82,636	49	164,366	15	0.84	**0.33**	0.41	0.46	0.36	0.48
Peanut Butter	51	108,770	53	220,186	7	1.90	**0.56**	1.23	1.17	0.81	0.81
Photo	80	17,047	91	33,769	18	0.74	0.90	0.45	0.54	**0.40**	0.43
Salty Snacks	39	736,148	39	1,491,938	3	1.40	0.50	0.85	0.83	**0.09**	0.33
Shampoo	66	290,429	82	568,503	28	0.44	**0.16**	0.38	0.36	0.16	0.26
Soup	24	905,541	29	1,819,666	5	0.55	**0.10**	0.40	0.29	0.23	0.27
Spaghetti/Italian Sauce	38	276,860	40	550,463	7	2.51	0.73	1.83	1.80	**0.22**	0.50
Sugar Substitutes	64	53,834	69	108,422	8	0.53	0.32	0.36	0.38	**0.15**	0.28
Toilet Tissue	27	112,788	34	226,410	8	1.01	**0.30**	0.70	0.64	0.45	0.46
Toothbrush	114	197,676	143	390,089	52	0.58	0.24	0.37	0.35	**0.22**	0.37
Toothpaste	42	238,271	48	474,519	8	0.42	**0.15**	0.39	0.39	0.18	0.25
Yogurt	43	499,203	49	1,028,179	9	3.98	**0.79**	3.12	2.33	1.40	0.88
(Mean)	–	–	–	–	–	1.30	0.37	0.90	0.83	**0.32**	0.41
(Median)	–	–	–	–	–	1.08	0.31	0.68	0.65	**0.27**	0.37
(Maximum)	–	–	–	–	–	3.98	1.32	3.12	2.33	1.40	**0.88**

Table 3.3: Comparison of out-of-sample KL divergence (in units of 10^{-2}) for the temporal splitting experiment.

We first define a function $\Delta(j, k, S)$ of a product j, a product $k \neq j$, and an assortment S that does not include product j and k, as

$$\Delta(j, k, S) = \frac{\mathbf{P}(j \mid S \cup \{j, k\}) - \mathbf{P}(j \mid S \cup \{j\})}{\mathbf{P}(j \mid S \cup \{j\})}, \tag{3.10}$$

which measures the relative change in the choice probability of product j when product k is introduced to assortment S. For convenience, we define $\Delta(j, k, S) \equiv 0$ if $j = k$. We say that product k complements product j under assortment S when $\Delta(j, k, S) > 0$. Similarly, we say product k substitutes product j under S when $\Delta(j, k, S) < 0$.

The substitution and complementarity relation depends on the existence of other products, i.e., on the assortment S. To quantify the overall impact of product k toward product j, we consider the averaged version of $\Delta(j, k, S)$. That is, we consider $\Delta^{\text{avg}}_{j,k} = (1/|\mathcal{S}^{\setminus jk}|) \cdot \sum_{S \in \mathcal{S}^{\setminus jk}} \Delta(j, k, S)$, where $\mathcal{S}^{\setminus jk}$ is the set of all assortments that do not include product j and k. Similarly, we can also define $\Delta^{\text{avg}}_{0,k}$ to measure the average impact of the addition of product k on the no-purchase option. We use $\mathbf{\Delta}^{\text{avg}} = [\Delta^{\text{avg}}_{j,k}]_{j \in \mathcal{N}_+, k \in \mathcal{N}}$ to denote the matrix of all such values.

Figure 3.9 illustrates two $\mathbf{\Delta}^{\text{avg}}$ matrices, one corresponding to the decision forest model with depth $d = 3$ (top matrix) and the other corresponding to the ranking-based model (down matrix), for the coffee product category. For each matrix, starting from the top-left corner that corresponds to $\Delta^{\text{avg}}_{1,1}$, the first 9-by-9 submatrix corresponds to $[\Delta^{\text{avg}}_{j,k}]_{j,k=1,\ldots,9}$ and the tenth row represents to $[\Delta^{\text{avg}}_{0,k}]_{k=1,\ldots,9}$, which captures the effects of the presence of each brand on the choice probability of the no-purchase option. Each cell corresponds to the effect of adding the brand on the corresponding column towards the brand on the corresponding row. The color level of each cell in each matrix represents the numeric value of $\mathbf{\Delta}^{\text{avg}}$ in accordance with the color bar on the right hand side of the figure: green corresponds to positive values and shows complementarity behavior, while red corresponds to negative values and shows substitution behavior.

Figure 3.9 shows that the decision forest model and the ranking-based model capture

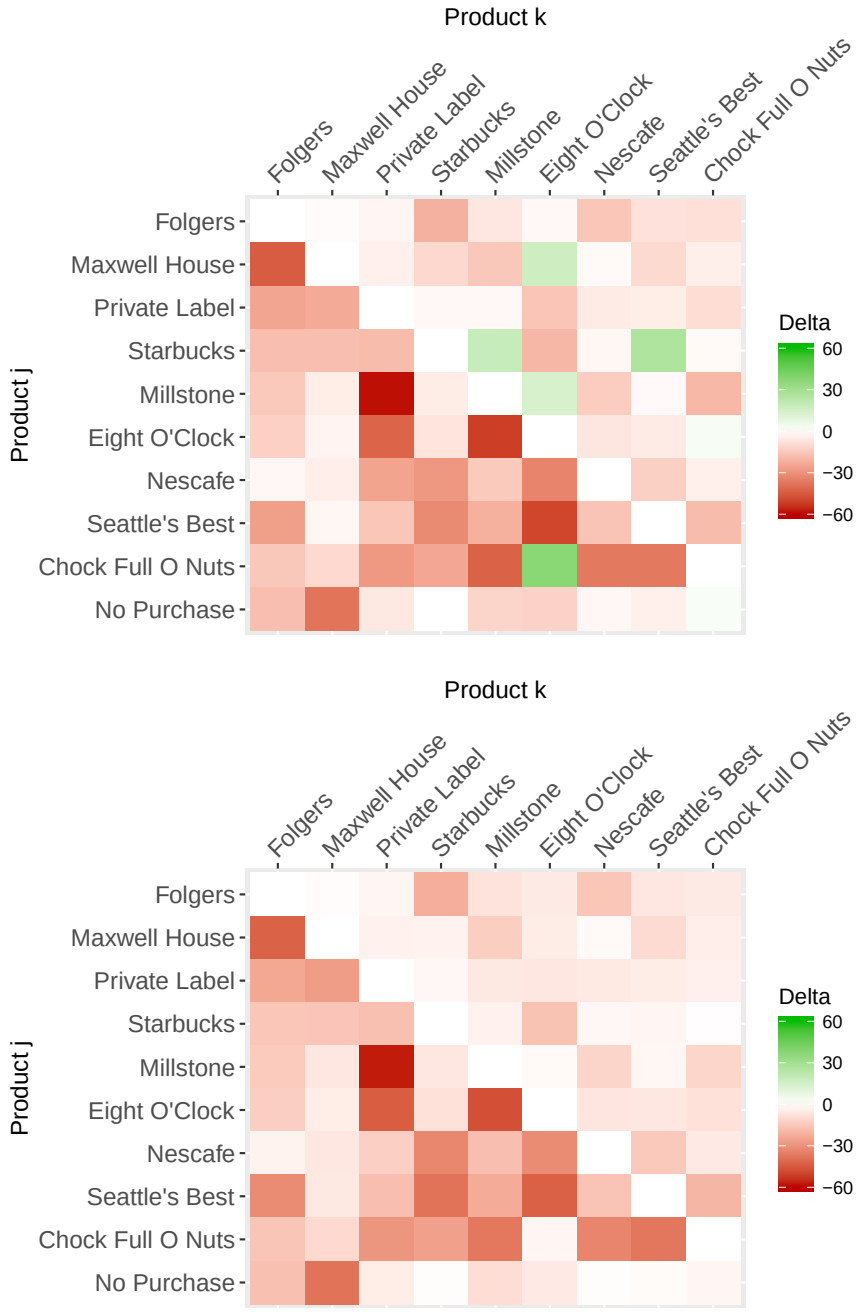


Figure 3.9: Illustration of the substitution/complementarity matrices $\mathbf{\Delta}^{\text{avg}}$ on coffee brands, corresponding to the decision forest (top) and the ranking-based model (down).

similar substitution patterns. For example, both models show that, on average, the choice probabilities of *Millstone* and *Eight O'Clock* decrease by about 60% and 40%, respectively, when *Private Label* is added, as shown in elements (5,3) and (6,3) from the top-left corner. However, since the ranking-based model satisfies the regularity property, all elements in $\mathbf{\Delta}^{\text{avg}}$ are forced to be non-positive. Thus, with the ranking-based model we are restricted to understanding only the substitution behavior between products, and we cannot use it to identify any complementarity behavior.

In contrast, the decision forest model is not constrained by the regularity property, and thus we can use it to identify interesting complementarities between certain brands. For example, Figure 3.9 shows that adding *Seattle's Best* to the assortment increases the choice probability of *Starbucks* by about 25% on average; the addition of *Eight O'Clock* provides a similar boost of about 16% to the choice probability of *Maxwell House*.

Another way to identify substitution and complementarity effects between products is to directly inspect the decision forest model. Figure 3.10 visualizes the top three trees by λ_t value of the decision forest model used in the top matrix of Figure 3.9. From left to right, the probability weights (λ_t values) of each tree are 4.1%, 2.5%, and 2.3%, respectively. The second tree exhibits the decoy effect that is described in Section 3.1.4. This customer type behaves in the following way: when *Eight O'Clock* exists in the assortment, the customer will purchase *Maxwell House* if it is available; otherwise, if *Eight O'Clock* does not exist in the assortment, the customer will purchase *Private Label* if it is available. This matches the complementarity effect shown in Figure 3.9 (element (2,6) in the left-hand matrix). The decision forest model is also capable of capturing effects that do not fit into well-studied customer behaviors in the marketing literature. For example, the first tree in Figure 3.10 corresponds to customers who purchase *Millstone* only if *Eight O'Clock* is observed in the assortment; otherwise, they do not make a purchase. Similarly, the third tree represents a decoy-like effect, where a customer checks for the existence of *Starbucks*: if it exists, the customer will purchase *Starbucks* if *Seattle's Best* is available; if not, the customer will pur-

chase *Maxwell House* if it is available. This highlights another benefit of our nonparametric approach: since we do not impose any assumptions on how the data is generated, we are able to discover interesting customer behaviors that fall outside of well-studied irrational behaviors.

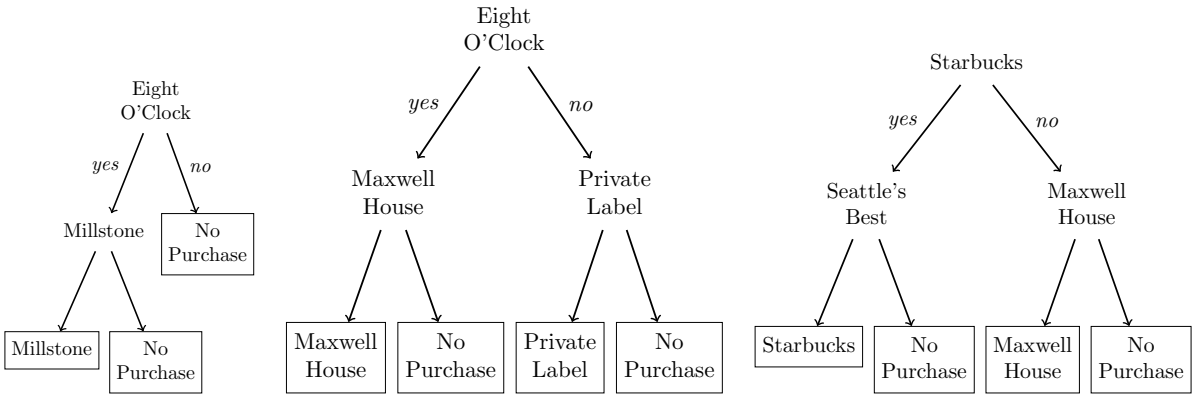


Figure 3.10: Top three decision trees on coffee brands with highest probability weights in the decision forest model learned from data.

CHAPTER 4

Assortment Optimization Under the Decision Forest Model

In this chapter[1], we study the problem of finding the assortment that maximizes expected revenue under the decision forest model. We approach this problem from a mixed-integer optimization perspective and propose three different formulations (Section 4.1) and theoretically compare these formulations in strength. We propose a methodology for solving these problems at a large-scale based on Benders decomposition (Section 4.2). Using synthetically generated instances, we demonstrate the tractability of our proposed approach, and their edge over heuristic approaches (Section 4.3). All proofs in this chapter are relegated to Section B.2 in Appendix.

4.1 Optimization model

In this section, we define the decision forest assortment optimization problem (Section 4.1.1) and subsequently develop our three formulations, LeafMIO (Section 4.1.2), SplitMIO (Section 4.1.2) and ProductMIO (Section 4.1.4).

[1]This chapter is based on my doctoral research work "Assortment Optimization Under the Decision Forest Model" [33].

4.1.1 Problem definition

For completeness, let us first briefly re-introduce[2] the decision forest model and then formally state the corresponding assortment optimization problem. We assume that there are N products, indexed from 1 to N, and let $\mathcal{N} = \{1, \ldots, N\}$ denote the set of all products. An assortment S is a subset of $\mathcal{N}$. When offered S, a customer may choose to purchase one of the products in S, or to not purchase anything at all; we use the index 0 to denote the latter possibility, which we will also refer to as the no-purchase option. We denote $S_+ \equiv S \cup \{0\}$ and $\mathcal{N}_+ \equiv \mathcal{N} \cup \{0\}$.

Recall that the basic building block of the decision forest model is a purchase decision tree. A purchase decision tree is a directed binary tree, with each leaf node corresponding to an option in $\mathcal{N}_+$, and each non-leaf (or *split*) node corresponding to a product in $\mathcal{N}$. We use $\mathbf{splits}(t)$ to denote the set of split nodes of tree t, and $\mathbf{leaves}(t)$ to denote the set of leaf nodes. We use $c(t, \ell)$ to denote the purchase decision of leaf ℓ of tree t, i.e., the option chosen by tree t if the assortment is mapped to leaf ℓ. We use $v(t, s)$ to denote the product that is checked at split node s in tree t.

Each tree represents the purchasing behavior of one type of customer. Specifically, for an assortment S, the customer behaves as follows: the customer starts at the root of the tree. The customer checks whether the product corresponding to the root node is contained in S; if it is, he proceeds to the left child, and if not, he proceeds to the right child. He then checks again with the product at the new node, and the process repeats, until the customer reaches a leaf; the option that is at the leaf represents the choice of that customer. Figure 4.1 shows an example of a purchase decision tree being used to map an assortment to a purchase decision. In the figure, leaf nodes are enclosed in squares, while split nodes are not enclosed. The number on each node corresponds either to $v(t, s)$ for splits, or $c(t, \ell)$ for leaves. The path highlighted in red indicates how a customer following this tree maps the assortment

[2]The notation and figures in this chapter are slightly different with the ones in Chapter 3, due to the need to incorporate the additional optimization procedure.

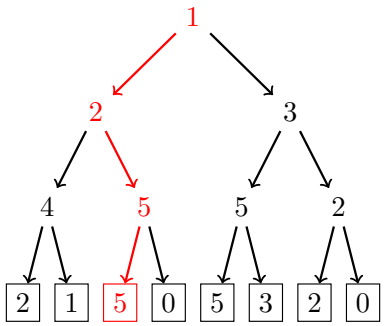


Figure 4.1: Example of a purchase decision tree for $n = 5$ products.

$S = \{1, 3, 4, 5\}$ to a leaf. For this assortment, the customer's decision is to purchase product 5.

The decision forest model assumes that the customer population is represented by a collection or *forest* F. Each tree $t \in F$ corresponds to a different customer type. We use λ_t to denote the probability associated with customer type/tree t, and $\boldsymbol{\lambda} = (\lambda_t)_{t \in F}$ to denote the probability distribution over the forest F. For each tree t, we use $\hat{A}(t, S)$ to denote the choice that a customer type following tree t will make when given the assortment S. For a given assortment $S \subseteq \mathcal{N}$ and a given choice $j \in S_+$, we use $\mathbf{P}^{(F,\boldsymbol{\lambda})}(j \mid S)$ to denote the choice probability, i.e., the probability of a random customer customer choosing j when offered the assortment S. It is defined as

$$\mathbf{P}^{(F,\boldsymbol{\lambda})}(j \mid S) = \sum_{t \in F} \lambda_t \cdot \mathbb{I}\{\hat{A}(t, S) = j\}. \tag{4.1}$$

We now define the assortment optimization problem. We use $\bar{r}_i$ to denote the marginal revenue of product i; for convenience, we use $\bar{r}_0 = 0$ to denote the revenue of the no-purchase option. The assortment optimization problem that we wish to solve is

$$\underset{S \subseteq \mathcal{N}}{\text{maximize}} \sum_{i \in S} \bar{r}_i \cdot \mathbf{P}^{(F,\boldsymbol{\lambda})}(i \mid S). \tag{4.2}$$

This is a challenging problem because of the general nature of the choice model $\mathbf{P}^{(F,\boldsymbol{\lambda})}(\cdot \mid \cdot)$. It turns out that problem (4.2) is theoretically intractable.

Proposition 2 *The decision forest assortment optimization problem* (4.2) *is NP-Hard.*

The proof of this proposition follows by a reduction from the MAX 3SAT problem; see see Section B.2.1. In the next three sections, we present different mixed-integer optimization (MIO) formulations of this problem.

4.1.2 Formulation 1: LeafMIO

We now present our first formulation of the assortment optimization problem (4.2) as a mixed-integer optimization (MIO) problem. To formulate the problem, we introduce some additional notation. For notational convenience we let $r_{t,\ell} = \bar{r}_{c(t,\ell)}$ be the revenue of the purchase option of leaf ℓ of tree t. We let $\mathbf{left}(s)$ denote the set of leaf nodes that are to the left of split s (i.e., can only be reached by taking the left branch at split s), and similarly, we let $\mathbf{right}(s)$ denote the leaf nodes that are to the right of s.

We introduce two sets of decision variables. For each $i \in \mathcal{N}$, we let x_i be a binary decision variable that is 1 if product i is included in the assortment, and 0 otherwise. For each tree $t \in F$ and leaf $\ell \in \mathbf{leaves}(t)$, we let $y_{t,\ell}$ be a binary decision variable that is 1 if the assortment encoded by $\mathbf{x}$ is mapped to leaf ℓ of tree t, and 0 otherwise.

With these definitions, our first formulation, LeafMIO, is given below.

$$\begin{aligned}
\text{LeafMIO}: \quad \underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{t\in F} \lambda_t \cdot \left[\sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} y_{t,\ell} \right] && \text{(4.3a)}\\
\text{subject to} \quad & \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, \quad \forall\, t \in F, && \text{(4.3b)}\\
& y_{t,\ell} \le x_{v(t,s)}, \quad \forall\, t \in F,\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{left}(s), && \text{(4.3c)}\\
& y_{t,\ell} \le 1 - x_{v(t,s)}, \quad \forall\, t \in F,\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{right}(s), && \text{(4.3d)}\\
& x_i \in \{0,1\}, \quad \forall\, i \in \mathcal{N}, && \text{(4.3e)}\\
& y_{t,\ell} \ge 0, \quad \forall\, t \in F,\ \ell \in \mathbf{leaves}(t). && \text{(4.3f)}
\end{aligned}$$

In order of appearance, the constraints in this formulation have the following meaning. Constraint (4.3b) requires that for each customer type t, the assortment encoded by $\mathbf{x}$ is mapped to exactly one leaf. Constraint (4.3c) requires that for any split s and any leaf ℓ that is to the left of split s, the assortment can be mapped to leaf ℓ only if the assortment includes the product $v(t,s)$ (i.e., if product $v(t,s)$ is not included in the assortment, then $y_{t,\ell}$ is forced to zero). Similarly, constraint (4.3d) requires the same for each split s and each leaf ℓ that is to the right of split s. The last two constraints require that $\mathbf{x}$ is binary and $\mathbf{y}$ is nonnegative. Note that it is not necessary to require $\mathbf{y}$ to be binary, as the constraints ensure that each $y_{t,\ell}$ automatically takes the correct value whenever $\mathbf{x}$ is binary. Finally, the objective function corresponds to the expected per-customer revenue of the assortment.

To motivate our main result, let $\mathcal{F}_{\text{LeafMIO}}$ denote the feasible region of the linear optimization relaxation of problem (4.3). Our main result is that, even in the simple case when the forest F consists of a single tree, $\mathcal{F}_{\text{LeafMIO}}$ may fail to be integral. We leave the proof to Section B.2.2.

Proposition 3 *There exists a decision forest model $(F, \boldsymbol{\lambda})$ with $|F| = 1$ such that $\mathcal{F}_{\text{LeafMIO}}$ is not integral, that is, there exists an extreme point $(\mathbf{x}, \mathbf{y}) \in \mathcal{F}_{\text{LeafMIO}}$ such that $\mathbf{x} \notin \{0,1\}^n$.*

The instance that we use to prove Proposition 3 is constructed so that each split corresponds to a distinct product, i.e., each product appears at most once in the splits of the tree. The dichotomy between decision forests where a product appears at most once in the splits of a given tree, and decision forests where a product may appear in two or more splits of a given tree, is important: the next formulation that we will consider, SplitMIO, is guaranteed to be integral in the former case when $|F| = 1$, but can be non-integral in the latter case.

4.1.3 Formulation 2: SplitMIO

While problem LeafMIO is one formulation of problem (4.2), it is not the strongest possible formulation. In particular, for a fixed split s, the constraints (4.3c) and (4.3d) can be aggregated over all leaves in $\mathbf{left}(s)$ and $\mathbf{right}(s)$, respectively, for a fixed split s. This leads to our second formulation, SplitMIO, which is defined below.

$$
\begin{aligned}
\text{SplitMIO}: \quad \underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{t \in F} \lambda_t \cdot \left[\sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} y_{t,\ell} \right] && \text{(4.4a)} \\
\text{subject to} \quad & \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, \quad \forall\, t \in F, && \text{(4.4b)} \\
& \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq x_{v(t,s)}, \quad \forall\, t \in F,\ s \in \mathbf{splits}(t), && \text{(4.4c)} \\
& \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - x_{v(t,s)}, \quad \forall\, t \in F,\ s \in \mathbf{splits}(t), && \text{(4.4d)} \\
& x_i \in \{0,1\}, \quad \forall\, i \in \mathcal{N}, && \text{(4.4e)} \\
& y_{t,\ell} \geq 0, \quad \forall\, t \in F,\ \ell \in \mathbf{leaves}(t). && \text{(4.4f)}
\end{aligned}
$$

Constraints (4.4b), (4.4e) and (4.4f) and the objective function are the same as in formulation LeafMIO. Constraint (4.4c) is an aggregated version of constraint (4.3c): for a split s in tree t, if product $v(t,s)$ is not in the assortment, then the assortment cannot be mapped to any of the leaves that are to the left of split s in tree t. Similarly, constraint (4.4d) is an aggregated version of constraint (4.3d), requiring that if $v(t,s)$ is included in the assortment, then the assortment cannot be mapped to any leaf to the right of split s in tree t. As in LeafMIO, $\mathbf{y}$ is modeled as nonnegative without affecting the validity of the formulation.

The above formulation we present here is related to two existing MIO formulations in the literature. The formulation here can be viewed as a specialized case of the MIO formulation in [89]. In that paper, the author develops a formulation for tree ensemble optimization, i.e.,

the problem of setting the independent variables in a tree ensemble model (e.g., a random forest or a gradient boosted tree model) to maximize the value predicted by that ensemble. Since the decision forest model is a type of tree ensemble model, where the "independent variables" are binary (i.e., product i is in the assortment or not), the formulation in [89] naturally applies here, leading to problem (4.4).

In addition to [89], problem (4.4) also relates to another MIO formulation, specifically that of [13]. In that paper, the authors develop a formulation for the product line design problem under the ranking-based model. As discussed in Section 3.1.3, if we restrict each tree in the forest to be a ranking, then the decision forest becomes a ranking-based choice model. Therefore, one can be verified that the formulation (4.4) actually coincides with the MIO formulation for product line design under ranking-based models presented in [13].

Before continuing to our other formulations, we establish a couple of important properties of problem (4.4). Let $\mathcal{F}_{\text{SplitMIO}}$ denote the feasible region of the linear optimization relaxation of problem (4.4). Our first result, alluded to above, is that SplitMIO is at least as strong as LeafMIO.

Proposition 4 *For any decision forest model* $(F, \boldsymbol{\lambda})$, $\mathcal{F}_{\text{SplitMIO}} \subseteq \mathcal{F}_{\text{LeafMIO}}$.

This result follows straightforwardly from the definition of SplitMIO; we thus omit the proof. Our second result concerns the behavior of SplitMIO when $|F| = 1$. When $|F| = 1$, we can show that $\mathcal{F}_{\text{SplitMIO}}$ is integral in a particular special case. (Note that in the statement of the proposition below, we drop the index t to simplify notation.)

Proposition 5 *Let* $(F, \boldsymbol{\lambda})$ *be a decision forest model consisting of a single tree, i.e.,* $|F| = 1$. *In addition, assume that for every* $i \in \mathcal{N}$, $v(s) = i$ *for at most one* $s \in \textbf{splits}$. *Then* $\mathcal{F}_{\text{SplitMIO}}$ *is integral, i.e., every extreme point* $(\mathbf{x}, \mathbf{y})$ *of the polyhedron* $\mathcal{F}_{\text{SplitMIO}}$ *satisfies* $\mathbf{x} \in \{0, 1\}^N$.

The proof of this result (see Section B.2.3) follows by showing that the constraint matrix defining $\mathcal{F}_{\text{SplitMIO}}$ is totally unimodular. In addition to Proposition 5, we also have the following proposition that sheds light on when SplitMIO is not integral.

Proposition 6 *There exists a decision forest model* $(F, \boldsymbol{\lambda})$ *with* $|F| = 1$ *and for which* $v(s_1) = v(s_2) = i$ *for at least two* $s_1, s_2 \in \mathbf{splits}$*,* $s_1 \neq s_2$ *and at least one* $i \in \mathcal{N}$*, such that* $\mathcal{F}_{\text{SplitMIO}}$ *is not integral.*

The proof of this result is given in Section B.2.4. Proposition 6 is significant because it implies that for $|F| = 1$, the distinction between trees where each product appears at most once in any split and trees where a product may appear two or more times as a split is sharp. This insight provides the motivation for our third formulation, ProductMIO, which we present next.

4.1.4 Formulation 3: ProductMIO

The third formulation of problem (4.2) that we will present is motivated by the behavior of SplitMIO when a product participates in two or more splits. In particular, observe that in a given purchase decision tree, a product i may participate in two different splits s_1 and s_2 in the same tree. In this case, constraint (4.4c) in problem (4.4) will result in two constraints:

$$\sum_{\ell \in \mathbf{left}(s_1)} y_{t,\ell} \leq x_i, \tag{4.5}$$

$$\sum_{\ell \in \mathbf{left}(s_2)} y_{t,\ell} \leq x_i. \tag{4.6}$$

In the above two constraints, observe that $\mathbf{left}(s_1)$ and $\mathbf{left}(s_2)$ are disjoint (this is a straightforward consequence of Requirement 3 in Definition 6 in Section 3.1.1). Given this and constraint (4.4b) that requires the $y_{t,\ell}$ variables to sum to 1, we can come up with a constraint that strengthens constraints (4.5) and (4.6) by combining them:

$$\sum_{\ell \in \mathbf{left}(s_1)} y_{t,\ell} + \sum_{\ell \in \mathbf{left}(s_2)} y_{t,\ell} \leq x_i. \tag{4.7}$$

In general, one can aggregate all the $y_{t,\ell}$ variables that are to the left of all splits involving a product i to produce a single left split constraint for product i. The same can also be done

for the right split constraints. Generalizing this principle leads to the following alternate formulation, which we refer to as ProductMIO:

$$\text{ProductMIO}: \quad \underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{t\in F} \lambda_t \cdot \left[\sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} y_{t,\ell} \right] \tag{4.8a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, \quad \forall\, t \in F, \tag{4.8b}$$

$$\sum_{\substack{s\in \mathbf{splits}(t):\\ v(t,s)=i}} \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq x_i, \quad \forall\, t \in F,\ i \in \mathcal{N}, \tag{4.8c}$$

$$\sum_{\substack{s\in \mathbf{splits}(t):\\ v(t,s)=i}} \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - x_i, \quad \forall\, t \in F,\ i \in \mathcal{N}, \tag{4.8d}$$

$$x_i \in \{0,1\}, \quad \forall\, i \in \mathcal{N}, \tag{4.8e}$$

$$y_{t,\ell} \geq 0, \quad \forall\, t \in F,\ \ell \in \mathbf{leaves}(t). \tag{4.8f}$$

Relative to SplitMIO, ProductMIO differs in several ways. First, note that while both formulations have the same number of variables, formulation ProductMIO has a smaller number of constraints. In particular, problem SplitMIO has one left and one right split constraints for each split in each tree, whereas ProductMIO has one left and one right split constraint for each product. When the trees involve a large number of splits, this can lead to a sizable reduction in the number of constraints. Note also that when a product does not appear in any splits of a tree, we can also safely omit constraints (4.8c) and (4.8d) for that product.

The second difference with formulation SplitMIO, as we have already mentioned, is in formulation strength. Let $\mathcal{F}_{\text{ProductMIO}}$ be the feasible region of the LO relaxation of ProductMIO. The following proposition formalizes the fact that formulation ProductMIO is at least as strong as formulation SplitMIO.

Proposition 7 *For any decision forest model* $(F, \boldsymbol{\lambda})$*,* $\mathcal{F}_{\text{ProductMIO}} \subseteq \mathcal{F}_{\text{SplitMIO}}$*.*

The proof follows straightforwardly using the logic given above; we thus omit the proof.

The last major difference is in how ProductMIO behaves when $|F| = 1$. We saw that a sufficient condition for $\mathcal{F}_{\text{SplitMIO}}$ to be integral when $|F| = 1$ is that each product appears in at most one split in the tree. In contrast, formulation ProductMIO is *always* integral when $|F| = 1$.

Proposition 8 *For any decision forest model* $(F, \boldsymbol{\lambda})$ *with* $|F| = 1$, $\mathcal{F}_{\text{ProductMIO}}$ *is integral.*

The proof of this proposition, given in Section B.2.5, follows by recognizing the connection between ProductMIO and another type of formulation in the literature. In particular, a stream of papers in the mixed-integer optimization community [71, 122, 123] has considered a general approach for deriving small and strong formulations of disjunctive constraints using independent branching schemes; we briefly review the most general such approach from [71]. In this approach, one has a finite ground set J, and is interested in optimizing over a particular subset of the $(|J| - 1)-$dimensional unit simplex over J, $\Delta^J = \{\boldsymbol{\lambda} \in \mathbb{R}^J \mid \sum_{j \in J} \lambda_j = 1; \boldsymbol{\lambda} \geq \mathbf{0}\}$. The specific subset that we are interested in is called a *combinatorial disjunctive constraint* (CDC), and is given by

$$\text{CDC}(\mathcal{S}) = \bigcup_{S \in \mathcal{S}} Q(S), \tag{4.9}$$

where $\mathcal{S}$ is a finite collection of subsets of J and $Q(S) = \{\boldsymbol{\lambda} \in \Delta \mid \lambda_j \leq 0 \text{ for } j \in J \setminus S\}$ for any $S \subseteq J$. This approach is very general: for example, by associating each j with a point $\mathbf{x}^j$ in $\mathbb{R}^n$, one can use $\text{CDC}(\mathcal{S})$ to model an optimization problem over a union of polyhedra, where each polyhedron is the convex hull of a collection of vertices in $S \in \mathcal{S}$.

A *k-way independent branching scheme of depth t* is a representation of $\text{CDC}(\mathcal{S})$ as a sequence of t choices between k alternatives:

$$\text{CDC}(\mathcal{S}) = \bigcap_{m=1}^{t} \bigcup_{i=1}^{k} Q(L_i^m), \tag{4.10}$$

where $L_i^m \subseteq J$. In the special case that $k = 2$, we can write $\text{CDC}(\mathcal{S}) = \cap_{m=1}^{t}(L_m \cup R_m)$ where $L_m, R_m \subseteq J$. This representation is known as a *pairwise independent branching scheme* and

the constraints of the corresponding MIO can be written simply as

$$\sum_{j \in L_m} \lambda_j \leq z_m, \quad \forall\, m \in \{1, \ldots, k\}, \tag{4.11a}$$

$$\sum_{j \in R_m} \lambda_j \leq 1 - z_m, \quad \forall\, m \in \{1, \ldots, k\}, \tag{4.11b}$$

$$z_m \in \{0, 1\}, \quad \forall\, m \in \{1, \ldots, k\}, \tag{4.11c}$$

$$\sum_{j \in J} \lambda_j = 1, \tag{4.11d}$$

$$\lambda_j \geq 0, \quad \forall\, j \in J. \tag{4.11e}$$

This particular special case is important because it is always integral (see Theorem 1 of [122]). Moreover, we can see that ProductMIO bears a strong resemblance to formulation (4.11). Constraints (4.11a) and (4.11a) correspond to constraints (4.8c) and (4.8d), respectively. In terms of variables, the λ_j and z_m variables in formulation (4.11) correspond to the $y_{t,\ell}$ and x_i variables in ProductMIO, respectively.

One notable difference is that in practice, one would use formulation (4.11) in a modular way; specifically, one would be faced with a problem where the feasible region can be written as $\text{CDC}(\mathcal{S}_1) \cap \text{CDC}(\mathcal{S}_2) \cap \cdots \cap \text{CDC}(\mathcal{S}_G)$, where each $\mathcal{S}_g$ is a collection of subsets of J. To model this feasible region, one would introduce a set of $z_{g,m}$ variables for the gth CDC, enforce constraints (4.11a) - (4.11c) for the gth CDC, and use only one set of λ_j variables for the whole formulation. Thus, the λ_j variables are the "global" variables, while the $z_{g,m}$ variables would be "local" and specific to each CDC. In contrast, in ProductMIO, the x_i variables (the analogues of z_m) are the "global" variables, while the $y_{t,\ell}$ variables (the analogues of λ_j) are the "local" variables.

4.2 Solution methodology based on Benders decomposition

While the formulations in Section 4.1 bring the assortment optimization problem under the decision forest choice model closer to being solvable in practice, the effectiveness of these

formulations can be limited in large-scale problems. In particular, consider the case where there is a large number of trees in the decision forest model and each tree consists of a large number of splits and leaves. In this setting, all three formulations – LeafMIO, SplitMIO and ProductMIO– will have a large number of $y_{t,\ell}$ variables and a large number of constraints to link those variables with the x_i variables, and may require significant computation time.

At the same time, LeafMIO, SplitMIO and ProductMIO share a common problem structure. In particular, all three formulations have two sets of variables: the $\mathbf{x}$ variables, which determine the products that are to be included, and the $(\mathbf{y}_t)_{t\in F}$ variables, which model the choice of each customer type. In addition, for any two trees t, t' such that $t \neq t'$, the $\mathbf{y}_t$ variables and $\mathbf{y}_{t'}$ variables do not appear together in any constraints. Thus, one can view each of the three formulations as a two-stage stochastic program, where each tree t corresponds to a scenario; the variable $\mathbf{x}$ corresponds to the first-stage decision; and the variable $\mathbf{y}_t$ corresponds to the second-stage decision under scenario t, which is appropriately constrained by the first-stage decision $\mathbf{x}$.

Thus, we can apply Benders decomposition to solve the problem. At a high level, Benders decomposition involves using linear optimization duality to represent the optimal value of the second-stage problem for each tree t as a piecewise-linear concave function of $\mathbf{x}$, and to eliminate the $(\mathbf{y}_t)_{t\in F}$ variables. One can then re-write the optimization problem in epigraph form, resulting in an optimization problem in terms of the $\mathbf{x}$ variable and an auxiliary epigraph variable θ_t for each tree t, and a large family of constraints linking $\mathbf{x}$ and θ_t for each tree t. Although the family of constraints for each tree t is too large to be enumerated, one can solve the problem through constraint generation.

The main message of this section of the paper is that, in most cases, the primal and the dual forms of the second-stage problem can be solved either in closed form (when $\mathbf{x}$ is binary) or via a greedy algorithm (when $\mathbf{x}$ is fractional), thus allowing one to identify violated constraints for either the relaxation or the integer problem in a computationally efficient manner.

In the remaining sections, we carefully analyze the second-stage problem for each of the three formulations. For LeafMIO, we show that the second-stage problem can be solved by a greedy algorithm when $\mathbf{x}$ is fractional (Section 4.2.1). For SplitMIO, we similarly show that the second-stage problem can be solved by a slightly different greedy algorithm when $\mathbf{x}$ is fractional (Section 4.2.2). For ProductMIO, we show that the same greedy approach does not solve the second-stage problem in the fractional case (Section 4.2.3). For all three formulations, when $\mathbf{x}$ is binary, we characterize the primal and dual solutions in closed form; due to space considerations, we relegate these results to the appendix (LeafMIO in Section B.1.1, SplitMIO in Section B.1.2 and ProductMIO in Section B.1.3). Lastly, in Section 4.2.4, we briefly describe our overall algorithmic approach to solving the assortment optimization problem, which involves solving the Benders reformulation of the relaxed problem, followed by the Benders reformulation of the integer problem.

4.2.1 Benders reformulation of the LeafMIO relaxation

The Benders reformulation of the LO relaxation of LeafMIO can be written as

$$\underset{\mathbf{x},\boldsymbol{\theta}}{\text{maximize}} \quad \sum_{t \in F} \lambda_t \theta_t \tag{4.12a}$$

$$\text{subject to} \quad \theta_t \leq G_t(\mathbf{x}), \quad \forall\ t \in F, \tag{4.12b}$$

$$\mathbf{x} \in [0,1]^n, \tag{4.12c}$$

where the function $G_t(\mathbf{x})$ is defined as the optimal value of the following subproblem corresponding to tree t:

$$
\begin{aligned}
G_t(\mathbf{x}) \;=\; & \underset{\mathbf{y}_t}{\text{maximize}} && \sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} \cdot y_{t,\ell} && \text{(4.13a)}\\
& \text{subject to} && \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, && \text{(4.13b)}\\
& && y_{t,\ell} \le x_{v(t,s)}, \quad \forall\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{left}(s), && \text{(4.13c)}\\
& && y_{t,\ell} \le 1 - x_{v(t,s)}, \quad \forall\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{right}(s), && \text{(4.13d)}\\
& && y_{t,\ell} \ge 0, \quad \forall\ \ell \in \mathbf{leaves}(t). && \text{(4.13e)}
\end{aligned}
$$

We now present a greedy algorithm for solving problem (4.13), which is presented below as Algorithm 4. The algorithm requires a bijection $\tau : \{1, \ldots, |\mathbf{leaves}(t)|\} \to \mathbf{leaves}(t)$ such that $r_{t,\tau(1)} \ge r_{t,\tau(2)} \ge \cdots \ge r_{t,\tau(|\mathbf{leaves}(t)|}$, i.e., an ordering of leaves in nondecreasing revenue. In addition, in the definition of Algorithm 4, we use $\mathbf{LS}(\ell)$ and $\mathbf{RS}(\ell)$ to denote the set of left and right splits, respectively, of ℓ, which are defined as

$$
\begin{aligned}
\mathbf{LS}(\ell) &= \{s \in \mathbf{splits}(t) \mid \ell \in \mathbf{left}(s)\},\\
\mathbf{RS}(\ell) &= \{s \in \mathbf{splits}(t) \mid \ell \in \mathbf{right}(s)\},
\end{aligned}
$$

In words, $\mathbf{LS}(\ell)$ is the set of splits for which we must proceed to the left in order to be able to reach ℓ, and $\mathbf{RS}(\ell)$ is the set of splits for which we must proceed to the right to reach ℓ. A split $s \in \mathbf{LS}(\ell)$ if and only if $\ell \in \mathbf{left}(s)$, and similarly, $s \in \mathbf{RS}(\ell)$ if and only if $\ell \in \mathbf{right}(s)$.

Intuitively, this algorithm progresses through the leaves in order of their revenue $r_{t,\ell}$, and sets the $y_{t,\ell}$ variable of each leaf ℓ to the highest it can be set to without violating constraints (4.13c) and (4.13d), while also ensuring that $\sum_\ell y_{t,\ell} \le 1$. At each stage of the algorithm, the algorithm keeps track of which constraints become tight through the event set $\mathcal{E}$. If the constraint (4.13c) becomes tight for a particular split-leaf pair (s, ℓ), we say that an $A_{s,\ell}$ event has occurred, and we add $A_{s,\ell}$ to $\mathcal{E}$. Similarly, if constraint (4.13d) becomes tight for (s, ℓ), we say that a $B_{s,\ell}$ event has occurred and add $B_{s,\ell}$ to $\mathcal{E}$. (In the case of a

tie, that is, when there is more than one split s which attains the minimum on line 13 or 17, we choose the split arbitrarily.) If the constraint (4.13b) holds, then we say that a C event has occurred, and we terminate the algorithm, as all the remaining $y_{t,\ell}$ variables cannot be set to anything other than zero. In addition to the events in $\mathcal{E}$, we also keep track of which $y_{t,\ell}$ variable was being modified when each event in $\mathcal{E}$ occurred; this is done through the function f. We note that both $\mathcal{E}$ and f are not essential for the primal algorithm, but they become important for the dual algorithm (to be defined as Algorithm 5 below), in order to determine the corresponding dual solution.

It turns out that Algorithm 4 returns a feasible solution that is an extreme point of the polyhedron defined in problem (4.13), which we establish as Theorem 5 below.

Theorem 5 *Fix $t \in F$. Let $\mathbf{y}_t$ be a solution to problem* (4.13) *produced by Algorithm 4. Then:*

a) $\mathbf{y}_t$ is a feasible solution to problem (4.13).

b) $\mathbf{y}_t$ is an extreme point of the feasible region of problem (4.13).

By Theorem 5, problem (4.13) is feasible; since the feasible region is additionally bounded, it follows that problem (4.13) has a finite optimal value. Therefore, by strong duality, the optimal objective value of problem (4.13) is equal to the optimal value of its dual. The dual of problem (4.13) can be written as:

$$\underset{\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t}{\text{minimize}} \quad \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} x_{v(t,s)} + \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{right}(s)} \beta_{t,s,\ell}(1 - x_{v(t,s)}) + \gamma_t \tag{4.14a}$$

$$\text{subject to} \quad \sum_{s : \ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} + \sum_{s : \ell \in \mathbf{right}(s)} \beta_{t,s,\ell} + \gamma_t \geq r_{t,\ell}, \quad \forall\ \ell \in \mathbf{leaves}(t), \tag{4.14b}$$

$$\alpha_{t,s,\ell} \geq 0, \quad \forall\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{left}(s), \tag{4.14c}$$

$$\beta_{t,s,\ell} \geq 0, \quad \forall\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{right}(s). \tag{4.14d}$$

Algorithm 4 Primal greedy algorithm for LeafMIO.

Require: Bijection $\tau : \{1, \ldots, |\mathbf{leaves}(t)|\} \to \mathbf{leaves}(t)$ such that $r_{t,\tau(1)} \geq r_{t,\tau(2)} \geq \cdots \geq r_{t,\tau(|\mathbf{leaves}(t)|)}$

Initialize $y_{t,\ell} \leftarrow 0$ for all $\ell \in \mathbf{leaves}(t)$
for $i = 1, \ldots, |\mathbf{leaves}(t)|$ **do**
    Set $q_A \leftarrow \min\{x_{v(t,s)} \mid s \in \mathbf{LS}(\tau(i))\}$
    Set $q_B \leftarrow \min\{1 - x_{v(t,s)} \mid s \in \mathbf{RS}(\tau(i))\}$
    Set $q_C \leftarrow 1 - \sum_{j=1}^{i-1} y_{t,\tau(j)}$
    Set $q^* \leftarrow \min\{q_A, q_B, q_C\}$
    Set $y_{t,\tau(i)} \leftarrow q^*$
    if $q^* = q_C$ **then**
        Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{C\}$
        Set $f(C) = \tau(j)$
        break
    else if $q^* = q_A$ **then**
        Select $s^* \in \arg\min_{s \in \mathbf{LS}(\tau(i))} x_{v(t,s)}$
        Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{A_{s^*,\tau(i)}\}$
        Set $f(A_{s^*,\tau(i)}) = \tau(i)$
    else
        Select $s^* \in \arg\min_{s \in \mathbf{RS}(\tau(i))} [1 - x_{v(t,s)}]$
        Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{B_{s^*,\tau(i)}\}$
        Set $f(B_{s^*,\tau(i)}) = \tau(i)$

Letting $\mathcal{D}_{t,\text{LEAFMIO}}$ denote the set of feasible solutions $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ to the dual subproblem (4.14), we can re-write the master problem (4.12) as

$$\begin{aligned}
\underset{\mathbf{x},\boldsymbol{\theta}}{\text{maximize}} \quad & \sum_{t \in F} \lambda_t \theta_t && \text{(4.15a)} \\
\text{subject to} \quad & \theta_t \leq \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} x_{v(t,s)} + \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{right}(s)} \beta_{t,s,\ell}(1 - x_{v(t,s)}) + \gamma_t, \\
& \qquad \forall\ (\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t) \in \mathcal{D}_{t,\text{LEAFMIO}}, && \text{(4.15b)} \\
& \mathbf{x} \in [0,1]^n. && \text{(4.15c)}
\end{aligned}$$

The value of this formulation, relative to the original formulation, is that we have replaced the $(\mathbf{y}_t)_{t \in F}$ variables and the constraints that link them to the $\mathbf{x}$ variables, with a large family of constraints in terms of $\mathbf{x}$. Although this new formulation is still challenging, the advantage of this formulation is that it is suited to constraint generation.

The constraint generation approach to solving problem (4.15) involves starting the problem with no constraints and then, for each $t \in F$, checking whether constraint (4.15b) is violated. If constraint (4.15b) is not violated for any $t \in F$, then we conclude that the current solution $\mathbf{x}$ is optimal. Otherwise, for any $t \in F$ such that constraint (4.15b) is violated, we add the constraint corresponding to the $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ solution at which the violation occurred, and solve the problem again to obtain a new $\mathbf{x}$. The procedure then repeats at the new $\mathbf{x}$ solution until no more violated constraints have been found.

The critical step in the constraint generation approach is the separation procedure for constraint (4.15b): that is, for a fixed $t \in F$, either asserting that the current solution $\mathbf{x}$ satisfies constraint (4.15b) or identifying a $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ at which constraint (4.15b). This amounts to solving the dual subproblem (4.14) and comparing its objective value to θ_t.

Fortunately, it turns out that we can solve the dual subproblem (4.14) using a specialized algorithm, in the same way that we can solve the primal subproblem (4.13) using Algorithm 4. Using the event set $\mathcal{E}$ and the mapping f produced by Algorithm 4, we can now consider a separate algorithm for solving the dual problem (4.14), which we present

below as Algorithm 5.

Algorithm 5 Dual greedy algorithm for LeafMIO.

Initialize $\alpha_{t,s,\ell} \leftarrow 0$, $\beta_{t,s,\ell} \leftarrow 0$ for all $s \in \textbf{splits}(t)$, $\ell \in \textbf{leaves}(t)$, $\gamma_t \leftarrow 0$.

Set $\gamma_t \leftarrow r_{t,f(C)}$

for $\ell \in \textbf{leaves}(t)$ **do**

    Set $\alpha_{t,s,\ell} = r_{t,f(A_{s,\ell})} - \gamma_t$ for any s such that $A_{s,\ell} \in \mathcal{E}$

    Set $\beta_{t,s,\ell} = r_{t,f(B_{s,\ell})} - \gamma_t$ for any s such that $B_{s,\ell} \in \mathcal{E}$

As with Algorithm 4, we can show that the dual solution produced by Algorithm 5 is a feasible extreme point solution of problem (4.14).

Theorem 6 *Fix $t \in F$. Let $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ be a solution to problem* (4.14) *produced by Algorithm 5. Then:*

a) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to problem (4.14).

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is an extreme point of the feasible region of problem (4.14).

Lastly, given the two solutions $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$, we now show that these solutions are optimal for their respective problems.

Theorem 7 *Fix $t \in F$. Let $\mathbf{y}_t$ be a solution to problem* (4.13) *produced by Algorithm 4 and let $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ be a solution to problem* (4.14) *produced by Algorithm 5. Then:*

a) $\mathbf{y}_t$ is an optimal solution to problem (4.13).

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is an optimal solution to problem (4.14).

Before continuing, we pause to make two important remarks on Theorem 7 and our results in this section. First, the value of Theorem 7 is that it allows us to use Algorithms 4 and

5 to solve the primal and dual subproblems (4.13) and (4.14). Thus, rather than invoking a linear optimization solver, such as Gurobi, to solve problem (4.14), we can simply run Algorithms 4 and 5.

Second, we note that the existence of a greedy algorithm is perhaps not too surprising, because of the connection between problem (4.13) and the 0-1 knapsack problem. In particular, consider the following problem:

$$\underset{\tilde{\mathbf{y}}_t}{\text{maximize}} \quad \sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} \cdot w_{t,\ell} \cdot \tilde{y}_{t,\ell} \tag{4.16a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}(t)} w_{t,\ell} \cdot \tilde{y}_{t,\ell} \leq 1, \tag{4.16b}$$

$$0 \leq \tilde{y}_{t,\ell} \leq 1, \quad \forall\ \ell \in \mathbf{leaves}(t). \tag{4.16c}$$

where the coefficient $w_{t,\ell}$ is defined as

$$w_{t,\ell} = \min \left\{ \min_{s \in \mathbf{LS}(\ell)} x_{v(t,s)}, \min_{s \in \mathbf{RS}(\ell)} (1 - x_{v(t,s)}) \right\},$$

and $\tilde{y}_{t,\ell}$ is a new decision variable defined for each $\ell \in \mathbf{leaves}(t)$. Note that this problem is equivalent to problem (4.13) with the constraint $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1$ relaxed to $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} \leq 1$. The coefficient $w_{t,\ell}$ has the interpretation of the tightest upper bound on $y_{t,\ell}$ in problem (4.13). The variable $\tilde{y}_{t,\ell}$ can therefore be viewed as a re-scaling of $y_{t,\ell}$ relative to this bound; in other words, we can recover $y_{t,\ell}$ from a solution by setting it as $y_{t,\ell} = w_{t,\ell} \cdot \tilde{y}_{t,\ell}$. Problem (4.16) is special because it is exactly the linear optimization relaxation of a 0-1 knapsack problem: each leaf ℓ correspond to an item; each $w_{t,\ell}$ value corresponds to item ℓ's weight; and the coefficient $r_{t,\ell} \cdot w_{t,\ell}$ corresponds to the profit of item ℓ. It is well-known that the optimal solution to the relaxation of a 0-1 knapsack problem can be obtained via a greedy heuristic that sets the fractional amount of each item to the highest it can be, in order of decreasing profit-to-weight ratio [83]. For problem (4.16) above, the profit-to-weight ratio is exactly $r_{t,\ell} \cdot w_{t,\ell} / w_{t,\ell} = r_{t,\ell}$, so the greedy algorithm coincides with our greedy algorithm (Algorithm 4).

4.2.2 Benders reformulation of the SplitMIO relaxation

We now turn our attention to the SplitMIO formulation. We can reformulate the relaxation of SplitMIO in the same way as LeafMIO; in particular, we have the same master problem (4.12), where the function $G_t(\mathbf{x})$ is now defined as the optimal value of the tree t subproblem in SplitMIO:

$$
\begin{aligned}
G_t(\mathbf{x}) = \quad & \underset{\mathbf{y}_t}{\text{maximize}} && \sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} \cdot y_{t,\ell} && \text{(4.17a)} \\
& \text{subject to} && \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, && \text{(4.17b)} \\
& && \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq x_{v(t,s)}, \quad \forall\ s \in \mathbf{splits}(t), && \text{(4.17c)} \\
& && \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - x_{v(t,s)}, \quad \forall\ s \in \mathbf{splits}(t), && \text{(4.17d)} \\
& && y_{t,\ell} \geq 0, \quad \forall\ \ell \in \mathbf{leaves}(t). && \text{(4.17e)}
\end{aligned}
$$

As with LeafMIO, it turns out that the primal subproblem (4.17) can be solved using a greedy algorithm, which we present below as Algorithm 6. As with Algorithm 4, this algorithm requires as input an ordering τ of the leaves in nondecreasing revenue. Like Algorithm 4, this algorithm also progresses through the leaves from highest to lowest revenue, and sets the $y_{t,\ell}$ variable of each leaf ℓ to the highest value it can be set to without violating the left and right split constraints (4.17c) and (4.17d) and without violating the constraint $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} \leq 1$. At each iteration, the algorithm additionally keeps track of which constraint became tight through the event set $\mathcal{E}$. An A_s event indicates that the left split constraint (4.17c) for split s became tight; a B_s event indicates that the right split constraint (4.17d) for split s became tight; and a C event indicates that the constraint $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} \leq 1$ became tight. When a C event is not triggered, Algorithm 6 looks for the split which has the least remaining capacity (line 17). In the case that the arg min is not unique and there are two or more splits that are tied, we break ties by choosing the split s

with the lowest depth $d(s)$ (i.e., the split closest to the root node of the tree).

The function f keeps track of which leaf ℓ was being checked when an A_s / B_s / C event occurred. As with LeafMIO, $\mathcal{E}$ and f are not needed to find the primal solution, but they are essential to determining the dual solution in the dual procedure (Algorithm 7, which we will define shortly).

The following result establishes that Algorithm 6 produces a feasible, extreme point solution of problem (4.17).

Theorem 8 *Fix $t \in F$. Let $\mathbf{y}_t$ be a solution to problem* (4.17) *produced by Algorithm 6. Then:*

a) $\mathbf{y}_t$ is a feasible solution to problem (4.17)*; and*

b) $\mathbf{y}_t$ is an extreme point of the feasible region of problem (4.17)*.*

As in our analysis of LeafMIO, a consequence of Theorem 8 is that problem (4.17) is feasible, and since the problem is bounded, it has a finite optimal value. By strong duality, the optimal value of problem (4.18) is equal to the optimal value of its dual:

$$\underset{\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t}{\text{minimize}} \quad \sum_{s \in \mathbf{splits}(t)} x_{v(t,s)} \cdot \alpha_{t,s} + \sum_{s \in \mathbf{splits}(t)} (1 - x_{v(t,s)}) \beta_{t,s} + \gamma_t \tag{4.18a}$$

$$\text{subject to} \quad \sum_{s: \ell \in \mathbf{left}(s)} \alpha_{t,s} + \sum_{s: \ell \in \mathbf{right}(s)} \beta_{t,s} + \gamma_t \geq r_{t,\ell}, \quad \forall\ \ell \in \mathbf{leaves}(t), \tag{4.18b}$$

$$\alpha_{t,s} \geq 0, \quad \forall\ s \in \mathbf{splits}(t), \tag{4.18c}$$

$$\beta_{t,s} \geq 0, \quad \forall\ s \in \mathbf{splits}(t). \tag{4.18d}$$

Letting $\mathcal{D}_{t,\text{SplitMIO}}$ denote the set of feasible solutions to the dual subproblem (4.18), we can

Algorithm 6 Primal greedy algorithm for SplitMIO.

Require: Bijection $\tau : \{1, \ldots, |\mathbf{leaves}(t)|\} \to \mathbf{leaves}(t)$ s.t. $r_{t,\tau(1)} \geq \cdots \geq r_{t,\tau(|\mathbf{leaves}(t)|)}$

Initialize $y_{t,\ell} \leftarrow 0$ for each $\ell \in \mathbf{leaves}(t)$.
for $i = 1, \ldots, |\mathbf{leaves}(t)|$ **do**
    Set $q_C \leftarrow 1 - \sum_{j=1}^{i-1} y_{t,\tau(j)}$.
    for $s \in \mathbf{LS}(\tau(i))$ **do**
        Set $q_s \leftarrow x_{v(t,s)} - \sum_{\substack{j=1: \\ \tau(j) \in \mathbf{left}(s)}}^{i-1} y_{t,\tau(j)}$
    for $s \in \mathbf{RS}(\tau(i))$ **do**
        Set $q_s \leftarrow 1 - x_{v(t,s)} - \sum_{\substack{j=1: \\ \tau(j) \in \mathbf{right}(s)}}^{i-1} y_{t,\tau(j)}$
    Set $q_{A,B} \leftarrow \min_{s \in \mathbf{LS}(\tau(i)) \cup \mathbf{RS}(\tau(i))} q_s$
    Set $q^* \leftarrow \min\{q_C, q_{A,B}\}$
    Set $y_{t,\tau(i)} \leftarrow q^*$
    if $q^* = q_C$ **then**
        Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{C\}$.
        Set $f(C) = \tau(i)$.
    else
        Set $s^* \leftarrow \arg\min_{s \in \mathbf{LS}(\tau(i)) \cup \mathbf{RS}(\tau(i))} q_s$
        if $s^* \in \mathbf{LS}(\tau(i))$ **then**
            Set $e = A_s$
        else
            Set $e = B_s$
        if $e \notin \mathcal{E}$ **then**
            Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$.
            Set $f(e) = \tau(i)$.

formulate the master problem (4.12) as

$$\underset{\mathbf{x},\boldsymbol{\theta}}{\text{maximize}} \quad \sum_{t \in F} \lambda_t \theta_t \tag{4.19a}$$

$$\text{subject to} \quad \theta_t \leq \sum_{s \in \mathbf{splits}(t)} x_{v(t,s)} \cdot \alpha_{t,s} + \sum_{s \in \mathbf{splits}(t)} (1 - x_{v(t,s)}) \beta_{t,s} + \gamma_t,$$

$$\forall\ (\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t) \in \mathcal{D}_{t,\textsc{SplitMIO}}, \tag{4.19b}$$

$$\mathbf{x} \in [0,1]^n. \tag{4.19c}$$

As with the Benders approach to LeafMIO, the crucial step to solving this problem is being able to solve the dual subproblem (4.18). Similarly to problem (4.17), we can also obtain a solution to the dual problem (4.18) via an algorithm that is formalized as Algorithm 7 below. Algorithm 7 uses auxiliary information obtained during the execution of Algorithm 6. In the definition of Algorithm 7, we use $d(s)$ to denote the depth of an arbitrary split, where the root split corresponds to a depth of 1, and $d_{\max} = \max_{s \in \mathbf{splits}(t)} d(s)$ is the depth of the deepest split in the tree. In addition, we use $\mathbf{splits}(t, d) = \{s \in \mathbf{splits}(t) \mid d(s) = d\}$ to denote the set of all splits at a particular depth d.

Algorithm 7 Dual greedy algorithm for SplitMIO.

Initialize $\alpha_{t,s} \leftarrow 0, \beta_{t,s} \leftarrow 0$ for all $s \in \mathbf{splits}(t)$, $\gamma_t \leftarrow 0$
Set $\gamma \leftarrow r_{f(C)}$
for $d = 1, \ldots, d_{\max}$ **do**
    for $s \in \mathbf{splits}(t, d)$ **do**
        if $A_s \in \mathcal{E}$ **then**
            Set $\alpha_{t,s} \leftarrow r_{t,f(A_s)} - \gamma_t - \sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ A_{s'} \in \mathcal{E}, \\ d(s') < d}} \alpha_{t,s'} - \sum_{\substack{s' \in \mathbf{RS}(f(A_s)): \\ B_{s'} \in \mathcal{E}, \\ d(s') < d}} \beta_{t,s'}$
        if $B_s \in \mathcal{E}$ **then**
            Set $\beta_{t,s} \leftarrow r_{t,f(B_s)} - \gamma_t - \sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ A_{s'} \in \mathcal{E}, \\ d(s') < d}} \alpha_{t,s'} - \sum_{\substack{s' \in \mathbf{RS}(f(A_s)): \\ B_{s'} \in \mathcal{E}, \\ d(s') < d}} \beta_{t,s'}$

We provide a worked example of the execution of both Algorithms 6 and 7 in Sec-

tion B.3.1.

Our next result, Theorem 9, establishes that Algorithm 7 returns a feasible, extreme point solution of the dual subproblem (4.18).

Theorem 9 *Fix $t \in F$. Let $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ be a solution to problem* (4.18) *produced by Algorithm 7. Then:*

a) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to problem (4.18)*; and*

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is an extreme point of the feasible region of problem (4.18)*.*

Lastly, and most importantly, we show that the solutions produced by Algorithms 6 and 7 are optimal for their respective problems. Thus, Algorithm 7 is a valid procedure for identifying values of $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ at which constraint (4.19b) is violated.

Theorem 10 *Fix $t \in F$. Let $\mathbf{y}_t$ be a solution to problem* (4.17) *produced by Algorithm 6 and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ be a solution to problem* (4.18) *produced by Algorithm 7. Then:*

a) $\mathbf{y}_t$ is an optimal solution to problem (4.17)*; and*

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is an optimal solution to problem (4.18)*.*

The proof of this result follows by verifying that the two solutions satisfy complementary slackness.

Before continuing, we note that Algorithms 6 and 7 can be viewed as the generalization of the algorithms arising in the Benders decomposition approach to the ranking-based assortment optimization problem in [13] (see Section 4 of that paper). The results of that paper show that the primal subproblem of the MIO formulation in [13] can be solved via a greedy algorithm (analogous to Algorithm 6) and the dual subproblem can be solved via an algorithm that uses information from the primal algorithm (analogous to Algorithm 7). This

generalization is not straightforward. The main challenge in this generalization is redesigning the sequence of updates in the greedy algorithm according to the tree topology. For the ranking-based assortment problem, one only needs to calculate the "capacities" (the q_s values in Algorithm 6) by subtracting the y values of the preceding products in the rank. In contrast, in Algorithm 6, one considers all left/right splits and the y values of their left/right leaves when constructing the lowest upper bound of y_ℓ for each leaf node ℓ. Also, as shown in Algorithm 7, the dual variables $\alpha_{t,s}$ and $\beta_{t,s}$ have to be updated according to the tree topology and the events $A_{s'}$ and $B_{s'}$ of the split s' with smaller depth. For these reasons, the primal and dual Benders subproblems for the decision forest assortment problem are more challenging than that of the ranking-based assortment problem.

4.2.3 Benders reformulation of the ProductMIO relaxation

Lastly, we can consider a Benders reformulation of the relaxation of ProductMIO. The Benders master problem is given by formulation (4.12) where the function $G_t(\mathbf{x})$ is defined as the optimal value of the ProductMIO subproblem for tree t. To aid in the definition of the subproblem, let $P(t)$ denote the set of products that appear in the splits of tree t:

$$P(t) = \{i \in \mathcal{N} \mid i = v(t,s) \text{ for some } s \in \mathbf{splits}(t)\}.$$

With a slight abuse of notation, let $\mathbf{left}(i)$ denote the set of leaves for which product i must be included in the assortment for those leaves to be reached, and similarly, let $\mathbf{right}(i)$ denote the set of leaves for which product i must be excluded from the assortment for those leaves to be reached; formally,

$$\mathbf{left}(i) = \bigcup_{\substack{s \in \mathbf{splits}(t): \\ v(t,s)=i}} \mathbf{left}(s),$$

$$\mathbf{right}(i) = \bigcup_{\substack{s \in \mathbf{splits}(t): \\ v(t,s)=i}} \mathbf{right}(s).$$

With these definitions, we can write down the ProductMIO subproblem as follows:

$$
\begin{aligned}
G_t(\mathbf{x}) = \quad & \underset{\mathbf{y}_t}{\text{maximize}} && \sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} \cdot y_{t,\ell} && \text{(4.20a)} \\
& \text{subject to} && \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, && \text{(4.20b)} \\
& && \sum_{\ell \in \mathbf{left}(i)} y_{t,\ell} \leq x_i, \quad \forall\, i \in P(t), && \text{(4.20c)} \\
& && \sum_{\ell \in \mathbf{right}(i)} y_{t,\ell} \leq 1 - x_i, \quad \forall\, i \in P(t), && \text{(4.20d)} \\
& && y_{t,\ell} \geq 0, \quad \forall\, \ell \in \mathbf{leaves}(t). && \text{(4.20e)}
\end{aligned}
$$

In the same way as LeafMIO and SplitMIO, one can consider solving problem (4.20) using a greedy approach, where one iterates through the leaves from highest to lowest revenue, and sets each leaf's $y_{t,\ell}$ variable to the highest possible value without violating any of the constraints. Unlike LeafMIO and SplitMIO, it unfortunately turns out that this greedy approach is not always optimal, which is formalized in the following proposition.

Proposition 9 *There exists an $\mathbf{x} \in [0,1]^n$, a tree t and revenues $\bar{r}_1, \ldots, \bar{r}_n$ for which the greedy solution to problem* (4.20) *is not optimal.*

The proof of Proposition 9 involves an instance where a product appears in more than one split. (Recall that ProductMIO and SplitMIO are equivalent when a product appears at most once in each tree.)

4.2.4 Overall Benders algorithm

We conclude Section 4.2 by summarizing how the results are used. In our overall algorithmic approach below, we focus on LeafMIO and SplitMIO, as the subproblem can be solved for these two formulations when $\mathbf{x}$ is either fractional or binary (whereas for ProductMIO, the subproblem can only be solved when $\mathbf{x}$ is binary).

1. *Relaxation phase.* We first solve the relaxed problem (problem (4.15) for LeafMIO or problem (4.19) for SplitMIO) using ordinary constraint generation. Given a solution $\mathbf{x} \in [0,1]^n$, we generate Benders cuts by running the primal-dual procedure (either Algorithm 4 followed by Algorithm 5 for LeafMIO, or Algorithm 6 followed by Algorithm 7 for SplitMIO).

2. *Integer phase.* In the integer phase, we add all of the Benders cuts generated in the relaxation phase to the integer version of problem (4.15) (if solving LeafMIO) or problem (4.19) (if solving SplitMIO). We then solve the problem as an integer optimization problem, where we generate Benders cuts for integer solutions using the closed form expressions in Section B.1 (Theorem 19 in Section B.1.1 if solving LeafMIO, or Theorem 20 in Section B.1.2 if solving SplitMIO). In either case, we add these cuts using *lazy constraint generation*. That is, we solve the master problem using a single branch-and-bound tree, and we check whether the main constraint of the Benders formulation (either constraint (4.15b) for LeafMIO or constraint (4.19b) for SplitMIO) is violated at every integer solution generated in the branch-and-bound tree.

4.3 Numerical Experiments with Synthetic Data

In this section, we present the results from our numerical experiments involving synthetically-generated problem instances. Section 4.3.1 describes how the instances were generated. Section 4.3.2 presents results on the tightness of the LO relaxation of the three formulations. Section 4.3.3 presents results on the tractability of the integer version of each formulation. Finally, Section 4.3.4 compares the Benders approach for SplitMIO with the direct solution approach and with a simple local search heuristic on a collection of large-scale instances. Our experiments were implemented in the Julia programming language, version 0.6.2 [17] and executed on Amazon Elastic Compute Cloud (EC2) using a single instance of type `r4.4xlarge` (Intel Xeon E5-2686 v4 processor with 16 virtual CPUs and 122 GB memory). All mixed-

integer optimization formulations were solved using Gurobi version 8.1 and modeled using the `JuMP` package [80].

We remark that our experiments here use synthetically generated decision forest models. We focus on synthetically generated instances as we were not able to obtain a suitable real transaction data set for estimating the decision forest that would lead to sufficiently large instances of the assortment problem. The evaluation of our optimization methodology on real decision forest instances is an important direction for future research.

4.3.1 Background

To test our method, we generate three different families of synthetic decision forest instances, which differ in the topology of the trees and the products that appear in the splits:

1. **T1 forest**. A T1 forest consists of balanced trees of depth d (i.e., trees where all leaves are at depth $d+1$). For each tree, we sample d products $i_1, \ldots, i_d$ uniformly without replacement from $\mathcal{N}$, the set of all products. Then, for every depth $d' \in \{1, \ldots, d\}$, we set the split product $v(t,s)$ as $v(t,s) = i_{d'}$ for every split s that is at depth d'.

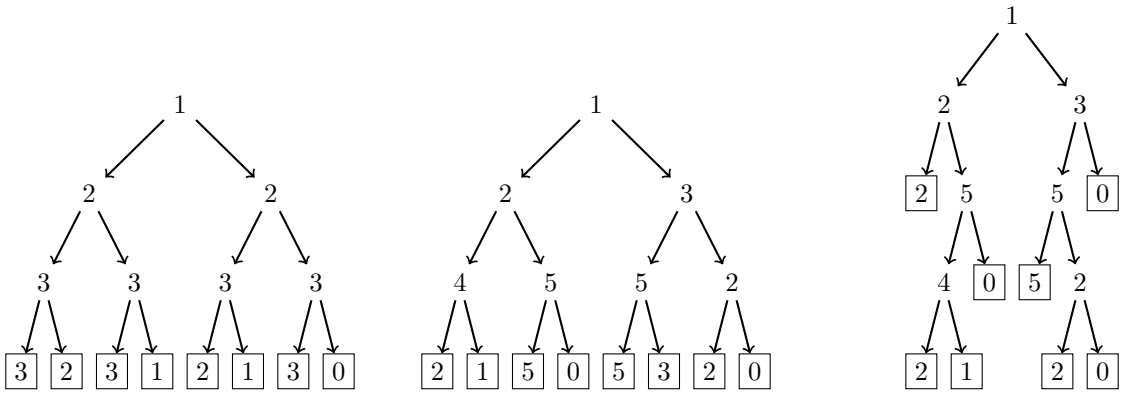
2. **T2 instances**. A T2 forest consists of balanced trees of depth d. For each tree, we set the split products at each split iteratively, starting at the root, in the following manner:

   (a) Initialize $d' = 1$.

   (b) For all splits s at depth d', set $v(s,t) = i_s$ where i_s is drawn uniformly at random from the set $\mathcal{N} \setminus \cup_{s' \in A(s)} \{v(t,s')\}$, where $A(s)$ is the set of ancestor splits to split s (i.e., all splits appearing on the path from the root node to split s).

   (c) Increment $d' \leftarrow d' + 1$.

   (d) If $d' > d$, stop; otherwise, return to Step (b).



(a) T1 tree ($d = 3$). (b) T2 tree ($d = 3$). (c) T3 tree ($L = 8$).

Figure 4.2: Examples of T1, T2 and T3 trees.

3. **T3 instances**. A T3 forest consists of unbalanced trees with L leaves. Each tree is generated according to the following iterative procedure:

   (a) Initialize t to a tree consisting of a single leaf.

   (b) Select a leaf ℓ uniformly at random from $\mathbf{leaves}(t)$, and replace it with a split s and two child leaves ℓ_1, ℓ_2. For split s, set $v(s,t) = i_s$ where i_s is drawn uniformly at random from $\mathcal{N} \setminus \cup_{s' \in A(s)} \{v(t,s')\}$.

   (c) If $|\mathbf{leaves}(t)| = L$, terminate; otherwise, return to Step (b).

For all three types of forests, we generate the purchase decision $c(t,\ell)$ for each leaf ℓ in each tree t in the following way: for each leaf ℓ, we uniformly at random choose a product $i \in \cup_{s \in \mathbf{LS}(\ell)} \{v(t,s)\} \cup \{0\}$. In words, the purchase decision is chosen to be consistent with the products that are known to be in the assortment if leaf ℓ is reached. Figure 4.2 shows an example of each type of tree (T1, T2, and T3). Given a forest of any of the three types above, we generate the customer type probability vector $\boldsymbol{\lambda} = (\lambda_t)_{t \in F}$ by drawing it uniformly from the $(|F| - 1)-$dimensional unit simplex.

In our experiments, we fix the number of products $n = 100$ and vary the number of

trees $|F| \in \{50, 100, 200, 500\}$, and the number of leaves $|\mathbf{leaves}(t)| \in \{8, 16, 32, 64\}$. (Note that the chosen values for $|\mathbf{leaves}(t)|$ correspond to depths of $\{3, 4, 5, 6\}$ for the T1 and T2 instances.) For each combination of n, $|F|$ and $|\mathbf{leaves}(t)|$ and each type of instance (T1, T2 and T3) we randomly generate 20 problem instances, where a problem instance consists of a decision forest model and the product marginal revenues $\bar{r}_1, \ldots, \bar{r}_n$. For each instance, the decision forest model is generated according to the process described above and the product revenues are sampled uniformly with replacement from the set $\{1, \ldots, 100\}$.

4.3.2 Experiment #1: Formulation Strength

Our first experiment is to simply understand how the three formulations – LeafMIO, SplitMIO and ProductMIO– compare in terms of formulation strength. Recall from Propositions 4 and 7 that SplitMIO is at least as strong as LeafMIO, and ProductMIO is at least as strong as SplitMIO. For a given instance and a given formulation $\mathcal{M}$ (one of LeafMIO, SplitMIO and ProductMIO), we define the integrality gap G_F as

$$G_{\mathcal{M}} = 100\% \times \frac{Z_{\mathcal{M}} - Z^*}{Z^*},$$

where Z^* is the optimal objective value of the integer problem. We consider the T1, T2 and T3 instances with $n = 100$, $|F| \in \{50, 100, 200, 500\}$ and $|\mathbf{leaves}(t)| = 8$. We restrict our focus to instances with $n = 100$ and $|\mathbf{leaves}(t)| = 8$, as the optimal value Z^* of the integer problem could be computed within one hour for these instances.

Table 4.1 displays the average integrality gap of each of the three formulations for each combination of n and $|F|$ and each instance type. From this table, we can see that in general there is an appreciable difference in the integrality gap between LeafMIO, SplitMIO and ProductMIO. In particular, the integrality gap of LeafMIO is in general about 2 to 44%; for SplitMIO, it ranges from 0.2 to 17%; and for ProductMIO, it ranges from 0 to 17%. Note that the difference between ProductMIO and SplitMIO is most pronounced for the T1 instances, as the decision forests in these instances exhibit the highest degree of

Type	$\|F\|$	G_{LeafMIO}	G_{SplitMIO}	$G_{\text{ProductMIO}}$
T1	50	2.4	0.9	0.0
T1	100	6.3	2.5	0.1
T1	200	13.0	5.6	0.2
T1	500	26.7	15.8	3.3
T2	50	2.1	0.2	0.2
T2	100	5.7	1.0	1.0
T2	200	14.8	5.4	5.3
T2	500	31.4	16.7	16.4
T3	50	5.4	0.2	0.2
T3	100	12.3	0.5	0.5
T3	200	23.8	4.1	3.9
T3	500	43.8	14.2	14.0

Table 4.1: Average integrality gap of LeafMIO, SplitMIO and ProductMIO for T1, T2 and T3 instances with $n = 100$, $|\mathbf{leaves}(t)| = 8$.

repetition of products within the splits of a tree. In contrast, the difference is smaller for the T2 and T3 instances, where the trees are balanced but there is less repetition of products within the splits of the tree (as the trees are not forced to have the same product appear on all of the splits at a particular depth).

4.3.3 Experiment #2: Tractability

In our second experiment, we seek to understand the tractability of LeafMIO, SplitMIO and ProductMIO when they are solved as integer problems (i.e., not as relaxations). For a given instance and a given formulation $\mathcal{M}$, we solve the integer version of formulation $\mathcal{M}$. Due to the large size of some of the problem instances, we impose a computation time limit of 1 hour for each formulation. We record $T_{\mathcal{M}}$, the computation time required for formulation

$\mathcal{M}$, and we record $\tilde{G}_{\mathcal{M}}$ which is the final optimality gap, and is defined as

$$\tilde{G}_{\mathcal{M}} = 100\% \times \frac{Z_{UB,\mathcal{M}} - Z_{LB,\mathcal{M}}}{Z_{UB,\mathcal{M}}}$$

where $Z_{UB,\mathcal{M}}$ and $Z_{LB,\mathcal{M}}$ are the best upper and lower bounds, respectively, obtained at the termination of formulation $\mathcal{M}$ for the instance. We test all of the T1, T2 and T3 instances with $n = 100$, $|F| \in \{50, 100, 200, 500\}$ and $|\mathbf{leaves}(t)| \in \{8, 16, 32, 64\}$.

Table 4.2 displays the average computation time and average optimality gap of each formulation for each combination of n, $|F|$ and $|\mathbf{leaves}(t)|$. Due to space considerations, we focus on the T3 instances; results for the T1 and T2 instances are provided in Section B.3.2. From this table, we can see that for the smaller instances, LeafMIO requires significantly more time to solve than SplitMIO, which itself requires more time to solve than ProductMIO. For larger instances, where the computation time limit is exhausted, the average gap obtained by ProductMIO tends to be lower than that of SplitMIO, which is lower than that of LeafMIO.

4.3.4 Experiment #3: Benders Decomposition for Large-Scale Problems

In this final experiment, we report on the performance of our Benders decomposition approach for solving large scale instances of SplitMIO. We focus on the SplitMIO formulation, as this formulation is stronger than the LeafMIO formulation, but unlike the ProductMIO formulation, we are able to efficiently generate Benders cuts for both fractional and integral values of $\mathbf{x}$.

We deviate from our previous experiments by generating a collection of T3 instances with $n \in \{200, 500, 1000, 2000, 3000\}$, $|F| = 500$ trees and $|\mathbf{leaves}(t)| = 512$ leaves. As before, the marginal revenue $\bar{r}_i$ of each product i is chosen uniformly at random from $\{1, \ldots, 100\}$. For each value of n, we generate 5 instances. For each instance, we solve the SplitMIO problem subject to the constraint $\sum_{i=1}^{n} x_i = b$, where b is set as $b = \rho n$ and we vary $\rho \in \{0.02, 0.04, 0.06, 0.08\}$.

Type	$\lvert F\rvert$	$\lvert\mathbf{leaves}(t)\rvert$	$\tilde{G}_{\text{LeafMIO}}$	$\tilde{G}_{\text{SplitMIO}}$	$\tilde{G}_{\text{ProductMIO}}$	T_{LeafMIO}	T_{SplitMIO}	$T_{\text{ProductMIO}}$
T3	50	8	0.0	0.0	0.0	0.1	0.0	0.0
T3	50	16	0.0	0.0	0.0	0.9	0.2	0.2
T3	50	32	0.0	0.0	0.0	13.3	0.8	0.8
T3	50	64	0.0	0.0	0.0	339.9	14.4	12.5
T3	100	8	0.0	0.0	0.0	0.4	0.1	0.1
T3	100	16	0.0	0.0	0.0	20.9	1.3	1.3
T3	100	32	0.0	0.0	0.0	1351.3	87.8	79.7
T3	100	64	8.2	4.2	3.5	3600.2	3512.8	3474.1
T3	200	8	0.0	0.0	0.0	2.8	0.5	0.5
T3	200	16	0.7	0.0	0.0	2031.9	210.5	184.8
T3	200	32	12.9	9.1	8.7	3600.2	3600.1	3600.1
T3	200	64	20.1	16.0	15.6	3600.3	3600.3	3600.1
T3	500	8	0.3	0.0	0.0	1834.0	307.5	245.0
T3	500	16	16.9	14.2	13.8	3600.2	3600.2	3600.1
T3	500	32	27.6	23.2	23.0	3600.6	3600.1	3600.1
T3	500	64	35.3	31.1	30.8	3600.8	3600.1	3600.1

Table 4.2: Comparison of final optimality gaps and computation times for LeafMIO, SplitMIO and ProductMIO, for T3 instances.

We compare three different methods: the two-phase Benders method described in Section 4.2.4, using the SplitMIO cut results (Section 4.2.2 and Section B.1.2); the divide-and-conquer (D&C) heuristic; and the direct solution approach, where we attempt to directly solve the full SplitMIO formulation using Gurobi. The D&C heuristic is a type of local search heuristic proposed in the product line design literature (see [67]; see also [8]). In this heuristic, one iterates through the b products currently in the assortment, and replaces a single product with the product outside of the assortment that leads to the highest improvement in the expected revenue; this process repeats until the assortment can no longer be improved. We choose the initial assortment uniformly at random from the collection of assortments of size b. For each instance, we repeat the D&C heuristic 10 times, and retain

the best solution. We do not impose a time limit on the D&C heuristic. For the Benders approach, we do not impose a time limit on the LO phase, and impose a time limit of one hour on the integer phase. For the direct solution approach, we impose a time limit of two hours; this time limit was chosen as it exceeded the total solution time used by the Benders approach across all of the instances.

Table 4.3 reports the performance of the three methods – the Benders approach, the D&C heuristic and direct solution of SplitMIO– across all combinations of n and ρ. In this table, $Z_{B,LO}$ indicates the objective value of the LO relaxation obtained after the Benders relaxation phase; $Z_{B,UB}$ and $Z_{B,LB}$ indicate the best upper and lower bounds obtained from Gurobi after the Benders integer phase; G_B indicates the final optimality gap of the Benders integer phase, defined as $G_B = (Z_{B,UB} - Z_{B,LB})/Z_{B,UB} \times 100\%$; $Z_{D\&C}$ indicates the objective value of the D&C heuristic; $RI_{D\&C}$ indicates the relative improvement of the final Benders solution over the D&C solution, defined as $RI_{D\&C} = (Z_{B,LB} - Z_{D\&C})/Z_{D\&C} \times 100\%$; Z_{Direct} indicates the best lower bound obtained from directly solving SplitMIO; and RI_{Direct} indicates the relative improvement of the final Benders solution over the final solution obtained from directly solving SplitMIO. The value reported of each metric is the average over the five replications corresponding to the particular (n, ρ) combination.

In addition to the comparison of the objective values obtained by the three methods, it is also useful to compare the methods by computation time. Table 4.4 displays the computation time required for all three methods. In this table, $T_{B,LO}$ indicates the time required by the LO relaxation phase of the Benders approach; $T_{B,IO}$ indicates the time required by the integer phase of the Benders approach; $T_{B,Total}$ indicates the total time of the Benders approach (i.e., $T_{B,LO} + T_{B,IO}$); $T_{D\&C}$ indicates the time required for the D&C heuristic; and T_{Direct} indicates the time required for the direct solution approach, i.e., solving SplitMIO directly using Gurobi. For all of these metrics, we report the average over the five replications for each combination of n and ρ. In addition, the table also reports the metric NU_{Direct}, which is the number of instances for which the direct solution approach terminated without an

n	ρ	b	$Z_{B,LO}$	$Z_{B,UB}$	$Z_{B,LB}$	G_B	$Z_{D\&C}$	$RI_{D\&C}$	Z_{Direct}	RI_{Direct}
200	0.02	4	13.10	12.69	12.69	0.00	12.69	0.00	12.69	0.00
200	0.04	8	24.98	21.95	21.95	0.00	21.95	0.00	21.95	0.00
200	0.06	12	36.71	32.43	29.27	9.83	29.23	0.13	29.27	0.00
200	0.08	16	48.00	43.67	35.90	17.85	35.87	0.10	35.82	0.22
500	0.02	10	16.55	16.38	16.38	0.00	16.35	0.18	16.38	0.00
500	0.04	20	29.61	28.37	28.11	0.93	28.07	0.15	28.11	0.00
500	0.06	30	42.00	40.90	37.46	8.42	37.24	0.58	37.32	0.38
500	0.08	40	53.61	52.65	45.03	14.46	44.67	0.81	44.56	1.06
1000	0.02	20	21.97	21.91	21.91	0.00	21.85	0.25	21.91	0.00
1000	0.04	40	37.43	37.03	36.46	1.55	35.94	1.45	36.44	0.05
1000	0.06	60	51.42	51.01	47.76	6.37	46.61	2.47	32.39	176.88
1000	0.08	80	63.60	63.28	56.61	10.55	55.32	2.33	29.82	208.25
2000	0.02	40	30.60	30.55	30.55	0.00	30.16	1.28	30.55	0.00
2000	0.04	80	48.74	48.45	48.31	0.30	46.29	4.34	48.31	0.00
2000	0.06	120	62.68	62.59	60.93	2.65	58.51	4.16	39.65	199.46
2000	0.08	160	73.81	73.77	69.76	5.43	67.05	4.05	34.66	294.00
3000	0.02	60	36.73	36.73	36.73	0.00	36.10	1.79	36.73	0.00
3000	0.04	120	57.13	56.98	56.88	0.18	54.52	4.35	56.88	0.00
3000	0.06	180	71.32	71.27	69.69	2.22	66.24	5.22	43.39	372.56
3000	0.08	240	81.46	81.44	74.76	8.20	74.43	0.43	10.52	620.54

Table 4.3: Comparison of the Benders decomposition approach, the D&C heuristic and direct solution of SplitMIO in terms of solution quality.

upper bound (in other words, the LO relaxation of SplitMIO was not solved within the two hour time limit).

Comparing the performance of the Benders approach with the D&C heuristic, we can see that in general, the Benders approach is able to find better solutions than the D&C heuristic. The performance gap, as indicated by the $RI_{D\&C}$ metric, can be substantial: with $n = 3000$ and $\rho = 0.06$, the Benders solution achieves an objective value that is on average more than 5% higher than that of the D&C heuristic's solution. In addition, from a computation time standpoint, the Benders approach compares quite favorably to the D&C heuristic. While the D&C heuristic is faster for small problems with low n and/or low ρ, it can require a significant amount of time for $n = 2000$ or $n = 3000$. In addition to this comparison against the D&C heuristic, in Section B.3.3 we also provide a comparison of the MIO solutions for the smaller T1, T2 and T3 instances used in the previous two sections against heuristic solutions; in those instances, we similarly find that solutions obtained from our MIO formulations can be significantly better than heuristic solutions.

Comparing the performance of the Benders approach with the direct solution approach, our results indicate two types of behavior. The first type of behavior corresponds to "easy" instances. These are instances with $\rho \in \{0.02, 0.04\}$ for which it is sometimes possible to directly solve SplitMIO to optimality within the two hour time limit. For example, with $n = 2000$ and $\rho = 0.04$, all five instances are solved to optimality by the direct approach. For those instances, the Benders approach is either able to prove optimality (for example, for $n = 200$ and $\rho = 0.04$, $G_B = 0\%$) or terminate with a low optimality gap (for example, for $n = 3000$ and $\rho = 0.04$, $G_B = 0.18\%$); among all instances with $\rho \in \{0.02, 0.04\}$, the average optimality gap is no more than about 1.6%. More importantly, the solution obtained by the Benders approach is at least as good as the solution obtained after two hours of direct solution of SplitMIO.

The second type of behavior corresponds to "hard" instances, which are the instances with $\rho \in \{0.06, 0.08\}$. For these instances, when one applies the direct approach, Gurobi is

n	ρ	b	$T_{B,LO}$	$T_{B,IO}$	$T_{B,Total}$	$T_{D\&C}$	T_{Direct}	NU_{Direct}
200	0.02	4	25.42	3.77	29.19	2.60	4901.79	1
200	0.04	8	41.19	371.58	412.77	5.64	7200.49	5
200	0.06	12	44.92	3600.02	3644.95	12.09	7200.37	5
200	0.08	16	45.82	3600.03	3645.85	23.63	7200.35	5
500	0.02	10	24.50	9.21	33.71	20.43	460.86	0
500	0.04	20	62.17	3126.73	3188.89	71.65	7200.36	5
500	0.06	30	66.95	3600.03	3666.98	161.35	7200.91	5
500	0.08	40	65.81	3600.03	3665.84	256.14	7200.35	5
1000	0.02	20	28.01	17.44	45.46	134.31	184.89	0
1000	0.04	40	89.80	3600.04	3689.84	507.19	7200.26	5
1000	0.06	60	106.99	3600.04	3707.02	1016.84	7200.99	5
1000	0.08	80	118.63	3600.03	3718.66	1552.29	7200.20	5
2000	0.02	40	26.13	3.60	29.73	878.12	63.35	0
2000	0.04	80	67.69	2242.50	2310.19	2558.43	2614.88	0
2000	0.06	120	247.57	3600.03	3847.60	5310.77	7200.40	5
2000	0.08	160	445.68	3600.04	4045.72	9911.23	7201.26	5
3000	0.02	60	26.32	1.21	27.53	2616.82	39.38	0
3000	0.04	120	170.58	3392.88	3563.46	7890.45	2830.19	0
3000	0.06	180	675.41	3600.04	4275.45	15567.13	7200.52	5
3000	0.08	240	1518.77	3600.04	5118.81	28186.04	7200.44	5

Table 4.4: Comparison of the Benders decomposition approach, the D&C heuristic and direct solution of SplitMIO in terms of solution time.

not able to solve the LO relaxation of SplitMIO within the two hour time limit for any instance (see the NU_{Direct} column of Table 4.4). In those instances, the integer solution returned by Gurobi is obtained from applying heuristics before solving the root node of the branch-and-bound tree, which is often quite suboptimal. In contrast, the Benders approach delivers significantly better solutions. In particular, as indicated by the RI_{Direct} column, for $n \in \{1000, 2000, 3000\}$, the Benders solution can achieve an objective value that is anywhere from 177% to 621% better, on average, than the solution obtained by Gurobi. It is also interesting to note that while Gurobi is not able to solve the LO relaxation within the two hour time limit, our Benders method solves it quickly; in the largest case, the solution time for the relaxation is no more than about 1500 seconds, or roughly 25 minutes. This highlights another benefit of our Benders approach, which is that it is capable of solving problems that are simply too large to be directly solved using a solver like Gurobi.

These results suggest that our Benders approach can solve large-scale instances of the assortment optimization problem in a reasonable computational timeframe and return high quality solutions that are at least as good, and often significantly better, than those obtained by the D&C heuristic or those obtained by directly solving the problem using Gurobi.

CHAPTER 5

Column-Randomized Linear Programs

In this chapter[1], we generalize a model estimation procedure of decision forest model (Section 3.3.2) as a solution method for large-scale linear optimization problems. Our contribution is two-fold. First, we propose a computationally-efficient algorithm that is easy to implement and can have a wide range of applications. Second, we provide a novel theoretical result that upper bounds the optimality gap of column/constraint sampling method in linear programming without structure assumptions.

5.1 Large-scale Linear Programs

We consider solving a linear program (LP) in standard form:

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad \mathbf{c}^T\mathbf{x} \tag{5.1a}$$

$$\text{such that} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.1b}$$

$$\mathbf{x} \geq \mathbf{0}, \tag{5.1c}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, and $\mathbf{b} \in \mathbb{R}^m$. In various applications of linear programming, such as the cutting-stock problem [66] and the vehicle routing problem [46], it is often the case that the number of variables n is much larger than the number of constraints m. Given that there are many more columns than constraints and enumerating all of the columns is impossible in most cases, a standard solution method is column generation (CG), which

[1]This chapter is based on my doctoral research work "Column-Randomized Linear Programs: Performance Guarantees and Applications" [32]

works as follows: (i) start with an initial set of columns from $\mathbf{A}$; (ii) solve the corresponding restricted linear program to optimality; (iii) solve a subproblem to find the column with the lowest reduced cost; (iv) add the new column to the current set of columns; (v) go back to step (i) until problem (5.1) is solved to optimality (i.e., the minimum reduced cost in step (iii) is nonnegative). The subproblem that ones solves to introduce a new column is often computationally challenging. For example, in the cutting-stock problem, the subproblem is a knapsack problem, which is NP-hard [64]. In practice, the subproblem is often formulated as an integer program, and can be difficult to solve at a large scale. In addition, CG is a *sequential* method, that is, the subproblem that one solves to introduce the ith column depends on the computational results of the previous $i-1$ iterations. Such a structure prohibits one from applying parallel computing techniques to implement the column generation method.

Instead of searching for columns by a subproblem that is potentially NP-hard, we propose a randomized method, called *column randomization*. In this method, one first samples a collection of columns according to a user-specified randomization scheme, and then solves the corresponding restricted linear program. We refer to this restricted linear program that consists of sampled columns as the *column-randomized linear program*. This approach is attractive because computationally, it is often significantly easier to randomly sample columns than it is to optimize over columns (as is the case in CG). In addition, while CG operates sequentially, the sampling step in column randomization is well-suited to parallelization.

We note that similar sampling-based methods for large-scale LPs have been previously considered in the operations research literature. In particular, there is a significant literature on solving problems with large numbers of constraints by randomly sampling constraints [39, 26]. By strong duality of linear programs, sampling the columns of problem (5.1) is equivalent to sampling the constraints of its dual problem. However, the behavior of the sampled LP in terms of its optimality gap – the difference in objective value between the sampled problem and the complete problem – has received scarce attention in the literature. In this paper,

our main goal is to answer the following question: *Given a user-specified randomization scheme for sampling columns from a linear program, is it possible to probabilistically bound the optimality gap of the column-randomized linear program?*

We provide theoretical results to answer this question and demonstrate how these results can be applied to common applications of large-scale linear programming. We make the following specific contributions:

1. **Theoretical Guarantees**. We show that with high probability over the sample of columns, the optimality gap of the column-randomized linear program is bounded by the sum of two terms: the optimality gap of a linear program related to the sampling distribution and a term that is of order $1/\sqrt{K}$, where K is the number of sampled columns. To best of our knowledge, this is the first theoretical result that addresses the behavior of the optimality gap of the column/constraint sampling technique for general linear programs without structural assumptions.

2. **Problem-Specific Bounds**. We apply the proposed method to several applications of large-scale linear programming and derive problem-specific upper bounds for the optimality gap. The problems include LPs with totally unimodular constraints, Markov decision processes (MDP), covering problems and packing problems. We also extend our approach to the portfolio optimization problem, in which the objective function is only assumed to be Lipschitz continuous (and is not necessarily linear or convex).

3. **Generalization to Non-I.I.D. Samples**. While the literature has mainly focus on independent and identically distributed (i.i.d.) samples, we generalize the randomization scheme to the case where the sampled columns may be statistically dependent, and develop a theoretical guarantee for this case. We apply our guarantee to a simple non-independent randomization scheme, where one samples n_r columns from each of $n_{\mathcal{G}}$ groups of columns, which applies to many LPs with columns that have a natural group structure (such as MDPs).

4. **Numerical Results**. We numerically demonstrate the effectiveness of the proposed method on two optimization problems that are commonly solved by CG: the cutting-stock problem, which is a classical application of linear programming; and the non-parametric choice model estimation problem, which is a modern application of linear programming. We compare the performance of the column randomization method to that of the CG method and show that for a fixed optimality gap, the column randomization method can attain that optimality gap within a fraction of the time required by CG. Thus, for some problems, the column randomization method can be a viable alternative to CG or can otherwise be used to provide a good warm start solution for CG.

We organize this chapter as follows. In Section 5.2, we state our theoretical results and discuss their implications. In Section 5.3, we apply our method to several applications of large-scale LP and derive problem-specific guarantees. In Section 5.4, we generalize our approach to sampling non-i.i.d. columns. In Section 5.5, we present our numerical results.

Related Works

Before we present our main results, we first review three streams of literature.

Column Generation (CG). CG has been widely used to solve optimization problems that have a huge number of columns compared to the number of constraints [37, 45, 59]. Applications include vehicle routing [46, 53], facility location problems [76], and choice model estimation [88, 120]; we refer readers to [43] for a comprehensive review. By strong duality of linear programs, CG is equivalent to constraint generation that solves linear programs with a large number of constraints [14]. A key component of both methods is the subproblem that one solves to iteratively introduce columns or constraints. Usually, this subproblem is computationally challenging and is often solved by integer programming. For example, in the cutting-stock problem, the CG subproblem is a knapsack problem, which is NP-hard

[64, 66].

Sampling Columns/Constraints. Another approach to solving LPs with huge numbers of columns (or equivalently, with huge numbers of constraints), is by sampling [26, 27, 28, 29, 39]. Specifically, one first samples a set of columns (or constraints) according to a given distribution then solves a linear program that consists of the sampled columns (or constraints). The seminal paper of [39] proposed the constraint sampling method for linear programs that arise in approximate dynamic programming (ADP). Given a distribution for sampling the constraints, the paper showed that with high probability over the sampled set of constraints, any feasible solution of the sampled problem is nearly feasible for the complete problem (that is, there is a high probability of satisfying a new random constraint, sampled according to the same distribution). Under the additional assumption that the constraint sampling distribution is a Lyapunov function, the paper also develops a specific guarantee on the error between the optimal value function and the approximate value function that is obtained by solving the sampled problem, but does not relate the objective value of the sampled and complete problems. In contrast, the results of our paper pertain specifically to the objective value of the sampled problem, are free from any assumptions on the sampling distribution and are applicable to general linear programs beyond those arising in ADP. Around the same period, [26, 27] pioneered the sampling approach to robust convex optimization. With a different perspective from [39], [26, 27] also characterized the sample complexity needed for the optimal solution (as opposed to an arbitrary feasible solution) of the sampled problem to be nearly feasible for the original problem. However, the performance of the sampled problem in terms of the objective value, and its dependence on the number of samples, was not addressed.

Since the works of [26] and [39], there has been some work that has quantified the dependence of the objective value on the number of sampled constraints. In particular, the paper of [90] considers a convex program where the decision variable $\mathbf{x}$ satisfies a family of convex constraints, which are later sampled, and is also constrained to lie in an ambient set

$\mathbb{X}$. The paper develops a probabilistic bound on the difference in objective value between the complete problem and its sampled counterpart in terms of a uniform level-set bound (ULB), which is a quantile function of the worst-case probability over all feasible solutions in set $\mathbb{X}$. Our work differs significantly from [90] in two aspects. First, in terms of the problem setting, [90] assumes that even before any constraints are sampled, the decision variable is already constrained in the convex compact (and thus bounded) set $\mathbb{X}$, and the associated performance guarantees also rely on properties of $\mathbb{X}$. In our setting, this corresponds to the dual solutions of problem (5.1) being bounded, which need not be the case in general. Moreover, we do not assume that the linear program is initialized with a specific set of variables (or equivalently, a set of constraints in the dual) before we sample columns. Consequently, the result of [90] is not directly applicable to the research question discussed in this paper. Second, as noted earlier, the performance bound in [90] relies on the ULB function of the sampling distribution. While sufficient conditions for the existence of a ULB are provided in the paper, in general a ULB cannot be represented explicitly and thus the resulting performance guarantee is less interpretable. In contrast, our theoretical results do not require a ULB or other related functions, and have a more interpretable dependence on the sampling distribution (via the distributional counterpart; see problem (5.7) in Theorem 11). In addition, we also believe our results are more straightforward technically: one only needs McDiarmid's inequality and standard linear programming results to prove them. As we will show in Section 5.3, our theoretical results and proof technique can be applied to many common types of LPs to derive application-specific guarantees.

Randomized Methods, Stochastic Optimization and Online Linear Programming. Besides column/constraint sampling, many other randomized methods have been proposed to solve large-scale optimization problems, including methods based on random walks [15] and random projection [95, 124]. In addition to these randomized methods, there is also a separate literature on optimization problems where stochasticity is part of the problem definition; some examples include stochastic programming [18, 109], contextual op-

timization [51], and online optimization [107]. Within this literature, the problem setting of online linear programming, where columns of a linear program are revealed sequentially to a decision maker, bears a resemblance to ours; some examples of papers in this area include [1, 50, 78]. Despite this similarity, this problem setting differs significantly from ours in that a decision maker is making irrevocable decisions in an online fashion: the decision maker must decide how much to use of a variable/column at the time that it is revealed, and cannot revise this decision in the future.

5.2 Theoretical Results

In this section, we first describe the basic notations and definitions that will be used throughout the paper (Section 5.2.1). Then we formally define the column randomization method and investigate its theoretical properties (Section 5.2.2). We end this section by discussing implications and interpretations of the theoretical results (Section 5.2.3). Proofs of the results are relegated to Section C.1.1.

5.2.1 Notation and Definitions

For any positive integer n, let $[n] \equiv \{1, 2, \ldots, n\}$. Let $\mathbf{e}_i$ be the ith standard basis vector for $\mathbb{R}^n$; that is, $\mathbf{e}_i = (e_{ij})$ where $e_{i,j} = 1$ if $j = i$ and $e_{i,j} = 0$ if $j \neq i$. Thus, for any $\mathbf{x} \in \mathbb{R}^n$, we can represent it as $\mathbf{x} = \sum_{i \in [n]} x_i \mathbf{e}_i$.

We consider a linear program in standard form:

$$P: \quad \min\{\mathbf{c}^T \mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\}, \tag{5.2}$$

where $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{c} \in \mathbb{R}^n$. We will refer to the problem P as the *complete problem* throughout the paper, as it contains all of the columns of $\mathbf{A}$.

We define the dual problem of problem (5.2) as

$$D: \quad \max\{\mathbf{p}^T \mathbf{b} \mid \mathbf{p}^T \mathbf{A} \leq \mathbf{c}^T\}. \tag{5.3}$$

For any optimization problem P', we denote its optimal objective value by $v(P')$ and its feasible region by $\mathcal{F}(P')$. By LP strong duality, we have $v(P) = v(D)$. Furthermore, for any optimization problem P'' that shares same objective function as the complete problem P and satisfies $\mathcal{F}(P'') \subseteq \mathcal{F}(P)$, we define $\Delta v(P'') \equiv v(P'') - v(P)$, which is nonnegative and can be interpreted as the optimality gap of solving P'' instead of P.

We make two assumptions on problem P. First, we assume that problem P is feasible and bounded; this assumption is not too restrictive, since the cases where the complete problem P is either unbounded or infeasible are not interesting to consider. The second assumption we make is that $\text{rank}(\mathbf{A}) = m$, i.e., the rows of $\mathbf{A}$ are linearly independent. This is also not too restrictive, as one can remove any rows of $\mathbf{A}$ that are linear combinations of the other rows without changing the problem.

For each $i \in [m]$ and $j \in [n]$, we use $\mathbf{A}^i$ and $\mathbf{A}_j$ to denote the ith row and jth column of matrix $\mathbf{A}$, respectively. For any collection of indices $J \subseteq [n]$, we let $\mathbf{A}_J$ represent the submatrix of $\mathbf{A}$ that consists of columns whose indices belong to J. In this paper, instead of solving either the complete problem P or its dual D, we consider solving a linear program whose columns are randomly selected. We call such a linear program a *column-randomized linear program*, which we formally define below.

Definition 7 (Column-Randomized Linear Program) *Let J be a finite collection of random indices, i.e., $J \equiv \{j_1, j_2, \ldots, j_K\}$ for an integer K, where $j_k \in [n]$ is a random variable for $k = 1, 2, \ldots, K$. Then the problem*

$$P_J : \quad \min\{\mathbf{c}_J^T \tilde{\mathbf{x}} \mid \mathbf{A}_J \tilde{\mathbf{x}} = \mathbf{b},\ \tilde{\mathbf{x}} \geq \mathbf{0}\} \tag{5.4}$$

is called a column-randomized linear program.

Clearly, P_J is equivalent to $\min\left\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0},\ x_j = 0\ \forall j \notin J.\right\}$. With this reformulation, any feasible solution of P_J can be represented as an element in $\mathcal{F}$. We can thus define $\Delta v(P_J)$ for the column-randomized LP P_J. We sample random indices in J by a

randomization scheme ρ, which is a computational procedure that randomly selects indices from $[n]$, or equivalently, randomly generates columns from $\mathbf{A}$. Let $\boldsymbol{\xi}$ be the probability distribution over $[n]$ that corresponds to ρ; that is, the jth component of $\boldsymbol{\xi}$, denoted by ξ_j, is the probability that index j is selected by ρ. Throughout this section, we assume ρ samples each index independently and identically according to $\boldsymbol{\xi}$. We will relax this assumption in Section 5.4.

We use D_J to denote the dual of P_J, which is defined as

$$D_J : \quad \max\{\mathbf{p}^T\mathbf{b} \mid \mathbf{p}^T\mathbf{A}_J \leq \mathbf{c}_J^T\}. \tag{5.5}$$

We will also require the notions of a basis, basic solutions and reduced costs in our theoretical results. A collection of indices $B \subseteq [n]$ of size m is called a *basis* if the matrix $\mathbf{A}_B$ is nonsingular, i.e., the collection of m columns $\{\mathbf{A}_j\}_{j\in B}$ is linearly independent. A *basic solution* $\mathbf{x}$ of the primal problem P corresponding to the basis B is the solution $\mathbf{x}$ obtained by setting $\mathbf{x}_B = \mathbf{A}_B^{-1}\mathbf{b}$, where $\mathbf{x}_B$ is the subvector corresponding to the columns in B, and $\mathbf{x}_N = \mathbf{0}$, where $\mathbf{x}_N$ is the subvector corresponding to the columns in $[n] \setminus B$. A solution $\mathbf{x}$ is called a *basic feasible solution* of P if it is a basic solution for some basis B and satisfies $\mathbf{x} \geq \mathbf{0}$. For the dual problem, a basic solution $\mathbf{p}$ corresponding to the basis B is the solution $\mathbf{p}$ defined by setting $\mathbf{p}^T = \mathbf{c}_B^T\mathbf{A}_B^{-1}$; if it additionally satisfies $\mathbf{p}^T\mathbf{A} \leq \mathbf{c}^T$, then it is also a basic feasible solution. Given a basis B, we define the reduced cost vector $\bar{\mathbf{c}}$ for that basis as $\bar{\mathbf{c}} \equiv \mathbf{c}^T - \mathbf{c}_B^T\mathbf{A}_B^{-1}\mathbf{A}$.

Finally, we use $\|\cdot\|$ to denote norms. For a vector $\mathbf{v} \in \mathbb{R}^n$, we let $\|\mathbf{v}\|_1 = \sum_{j=1}^n |v_j|$ be its ℓ_1 norm, $\|\mathbf{v}\|_2 = \sqrt{\sum_{j=1}^n v_j^2}$ be its Euclidean or ℓ_2 norm, and $\|\mathbf{v}\|_\infty = \max_{j=1,\ldots,n} |v_j|$ be its ℓ_∞ norm. For a matrix $\mathbf{A}$, we let $\|\mathbf{A}\|_{\max} = \max_{i,j} |A_{i,j}|$. Without loss of generality, we assume that the cost vector $\mathbf{c}$ has unit Euclidean norm, i.e., $\|\mathbf{c}\|_2 = 1$. This is not a restrictive assumption, because by normalizing the cost vector $\mathbf{c}$ to have unit Euclidean norm, the objectives of the complete problem P and the column-randomized problem P_J are both scaled by $1/\|\mathbf{c}\|_2$. Thus, the relative performance of problem P_J to the complete

problem P, which is the main focus of our paper, remains the same.

5.2.2 Main Theoretical Results

We propose the *column randomization method* in Algorithm 8. We first sample K indices, $j_1, j_2, \ldots, j_K$, by a randomization scheme ρ and let $J = \{j_1, \ldots, j_K\}$. We then collect the corresponding columns of $\mathbf{A}$ as matrix $\mathbf{A}_J$ and the corresponding components of $\mathbf{c}$ as vector $\mathbf{c}_J$. After forming $\mathbf{A}_J$ and $\mathbf{c}_J$, we solve the LP (5.6) and return its optimal value $v(P_J)$ and optimal solution.

Algorithm 8 The Column Randomization Method

1: Sample K indices as $J \equiv \{j_1, \ldots, j_K\}$ by a randomization scheme ρ.

2: Define $\mathbf{A}_J = [A_{j_1}, \ldots, A_{j_K}]$ and $\mathbf{c}_J = [c_{j_1}, \ldots, c_{j_K}]$.

3: Solve the column-randomized linear program, which only has K columns:

$$P_J : \quad \min \left\{ \mathbf{c}_J^T \tilde{\mathbf{x}} \mid \mathbf{A}_J \tilde{\mathbf{x}} = \mathbf{b},\ \tilde{\mathbf{x}} \geq \mathbf{0} \right\}. \tag{5.6}$$

return optimal objective value $v(P_J)$ and an optimal solution $\tilde{\mathbf{x}}^*$.

Notice that an optimal solution $\tilde{\mathbf{x}}^*$ of problem P_J can be immediately converted to a feasible solution for the complete problem P by enlarging $\tilde{\mathbf{x}}^*$ to length n and setting $\tilde{x}_j^* = 0$ for $j \in [n] \setminus J$.

We now present two theorems that bound the optimality gap $\Delta v(P_J) \equiv v(P_J) - v(P)$ of problem P_J; we defer our discussion of these two theorems to Section 5.2.3 and relegate the proofs of the theorems to Section C.1.1.

Theorem 11 *Let C be a positive constant and define the linear program P_{distr} as*

$$P_{distr} \equiv \quad \underset{\mathbf{x} \in \mathbb{R}^n}{\textit{minimize}} \quad \mathbf{c}^T \mathbf{x} \tag{5.7a}$$

$$\textit{such that} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.7b}$$

$$\mathbf{0} \leq \mathbf{x} \leq C \cdot \boldsymbol{\xi}. \tag{5.7c}$$

Let P_J be the column-randomized LP solved by Algorithm 8, and $\mathbf{A}_J$ be the corresponding constraint matrix. For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sample J, the following holds: if P_J is feasible and $\operatorname{rank}(\mathbf{A}_J) = m$*, then*

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + \frac{C\,(1 + m\gamma\|\mathbf{A}\|_{\max})}{\sqrt{K}} \left(1 + \sqrt{2\log\frac{2}{\delta}}\right), \tag{5.8}$$

where γ is an upper bound on $\|\mathbf{p}\|_\infty$ for every basic solution $\mathbf{p}$ of the dual problem D and $\|\mathbf{A}\|_{\max} = \max_{ij} |A_{ij}|$.

Theorem 11 shows that, with probability at least $1-\delta$, the optimality gap $\Delta v(P_J)$ of the column-randomized LP P_J is upper bounded by the sum of two terms. The first term is the optimality gap $\Delta v(P_{\text{distr}})$ of the problem P_{distr}, which we refer to as the *distributional counterpart*. The second term involves $\|\mathbf{A}\|_{\max}$, the largest absolute value of elements in the constraint matrix; γ, the upper bound of the ℓ_∞ norm of any basic solution of the dual problem; δ, the confidence parameter; and K, the number of sampled columns. Most importantly, the second term converges to zero with a rate $1/\sqrt{K}$. In Section 5.3, we will see how γ and $\|\mathbf{A}\|_{\max}$ can be further bounded for certain special cases.

We now present our second theorem, which relates the optimality gap to the reduced costs of the complete problem.

Theorem 12 *Define C, P_{distr}, P_J and $\mathbf{A}_J$ as in Theorem 11. For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sample J, the following holds: if P_J is feasible and* $\operatorname{rank}(\mathbf{A}_J) = m$*, then*

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + \frac{C}{\sqrt{K}} \cdot \chi \cdot \left(1 + \sqrt{2\log\frac{1}{\delta}}\right) \tag{5.9}$$

where χ is an upper bound on $\|\bar{\mathbf{c}}\|_2$ for every basic solution of the complete problem P.

Theorem 12 has a similar structure to Theorem 11. Compared to Theorem 11, the upper bound in Theorem 12 does not involve γ and $\|\mathbf{A}\|_{\max}$, but instead requires a bound on the norm of the reduced cost vector for all the bases of P.

5.2.3 Discussion on main theorems

Both Theorem 11 and 12 provide bounds on the optimality gap $\Delta v(P_J)$ of the following form:

$$\Delta v(P_J) \leq \Delta v(P_{\text{distr}}) + \frac{C \cdot C_P \cdot C_\delta}{\sqrt{K}}, \tag{5.10}$$

where C_P only depends on properties of the complete problem P and C_δ only depends on the confidence parameter δ. In Theorem 11, $C_P = 1 + m\gamma\|\mathbf{A}\|_{\max}$ and $C_\delta = 1 + \sqrt{2\log(2/\delta)}$; in Theorem 12, $C_P = \chi$ and $C_\delta = 1 + \sqrt{2\log(1/\delta)}$. In the following discussion, we first focus on the general structure of the upper bounds given in (5.10), and subsequently we address the differences between Theorem 11 and Theorem 12.

Role of Problem P_{distr}:

The distributional counterpart P_{distr} is the restricted version of the complete problem P, which includes the additional constraint $\mathbf{x} \leq C\boldsymbol{\xi}$. Thus, $\Delta v(P_{\text{distr}}) \geq 0$. If there exists an optimal solution $\mathbf{x}^*$ of the complete problem P such that $\mathbf{0} \leq \mathbf{x}^* \leq C\boldsymbol{\xi}$, then $\Delta v(P_{\text{distr}}) = 0$. Notice that neither Theorem 11 nor 12 implies that the optimality gap $\Delta v(P_J)$ of the column-randomized linear program P_J can be arbitrarily small with large K. Indeed, if $\boldsymbol{\xi}$ is not "comprehensive" enough – that is, its support is small, and does not include the complete set of columns of any optimal basis for P – then one would not expect the column-randomized program P_J to perform closely to the complete problem P, even if K is sufficiently large. In other words, problem P_{distr} reflects the "coverage" ability of the distribution $\boldsymbol{\xi}$, or equivalently, its randomization scheme ρ.

Role of Constant C:

Given a randomization scheme ρ and its corresponding distribution $\boldsymbol{\xi}$, as the constant C increases, the optimality gap $\Delta v(P_{\text{distr}})$ of problem P_{distr} decreases since its feasible set $\mathcal{F}(P_{\text{distr}})$

is enlarged. On the other hand, the second term on the RHS of bound (5.10) increases since it is proportional to C. To interpret this phenomenon, we can view bound (5.10) as a type of bias-complexity/bias-variance tradeoff, which is common in statistical learning theory [108]:

$$\Delta v(P_J) \leq \underbrace{\Delta v(P_{\text{distr}})}_{\text{Approximation Error}} + \underbrace{\frac{C \cdot C_P \cdot C_\delta}{\sqrt{K}}}_{\text{Sampling Error}}. \tag{5.11}$$

When the constant C increases, the feasible set $\mathcal{F}(P_{\text{distr}})$ gradually becomes a better approximation of the feasible set $\mathcal{F}(P)$, as more feasible solutions in $\mathcal{F}(P)$ are included in $\mathcal{F}(P_{\text{distr}})$. The optimality gap $\Delta v(P_{\text{distr}})$, which can be viewed as the approximation error, is thus narrowed. On the other hand, as the set $\mathcal{F}(P_{\text{distr}})$ expands, one needs more samples to ensure that the sampled feasible set $\mathcal{F}(P_J)$ can approximate $\mathcal{F}(P_{\text{distr}})$. In that sense, as we increase C, the second term of the right-hand side of (5.11) also increases.

Feasibility of P_J:

We make several important remarks regarding the feasibility of P_J and how feasibility is incorporated in our guarantee. First, note that in general, the sampled problem P_J need not be feasible. As a simple example, consider the following complete problem:

$$P = P_{\mathbf{I}} \equiv \min\{\mathbf{1}^T\mathbf{x} \mid \mathbf{I}\mathbf{x} = \mathbf{1}, \mathbf{x} \geq \mathbf{0}\},$$

where $\mathbf{I}$ is the n-by-n identity matrix and $m = n$. In this problem, the only way that the sampled problem P_J can be feasible is if the collection $j_1, \ldots, j_K$ includes every index in $[n]$; if any column $j \in [n]$ is not part of the sample J, then the sampled problem P_J is automatically infeasible. Thus, when $K < n$, P_J is infeasible almost surely. When $K \geq n$, it is still possible that $j_1, \ldots, j_K$ does not include all indices in $[n]$, and thus P_J is infeasible with positive probability.

For this reason, our guarantee on the optimality gap is stated as a *conditional* guarantee: with high probability over the sample $j_1, \ldots, j_K$, the optimality gap of P_J obeys a

particular bound *if the column-randomized LP is feasible*. We note that this is distinct from probabilistically conditioning on $j_1, \ldots, j_K$, i.e., our guarantee is *not* the same as

$$\mathbf{Pr}\left[\Delta v(P_J) \leq \Delta v(P_{\text{distr}}) + \frac{C}{\sqrt{K}} \cdot C_P \cdot C_\delta \;\middle|\; P_J \text{ is feasible}\right] \geq 1 - \delta,$$

because upon conditioning on the feasibility of P_J, the random variables $j_1, \ldots, j_K$ are in general no longer an i.i.d. sample. As an example of this, consider again problem $P_{\mathbf{I}}$ above, with $K = n$ and a randomization scheme ρ corresponding to the uniform distribution $\boldsymbol{\xi} = (1/n, \ldots, 1/n)$ over $[n]$. By conditioning on the event that P_J is feasible, the sample $J = \{j_1, \ldots, j_K\}$ must then be exactly equal to $[n]$, and we obtain that $\mathbf{Pr}[j_k = t, j_{k'} = t] = 0 \neq \mathbf{Pr}[j_k = t] \cdot \mathbf{Pr}[j_{k'} = t]$ for any $k, k' \in [K]$ with $k \neq k'$ and $t \in [n]$. In this example, the indices $j_1, \ldots, j_K$ are thus not independent.

With regard to the feasibility of column-randomized LPs, it appears to be difficult to guarantee feasibility in general. However, one can use similar techniques as in the proofs of our main results to characterize the near-feasibility of a column-randomized LP. Consider the following complete problem, and its sampled and distributional counterparts:

$$\begin{aligned}
P^{\text{feas}} &= \min\{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 \mid \mathbf{x} \geq \mathbf{0}\}, \\
P_J^{\text{feas}} &= \min\{\|\mathbf{A}_J\tilde{\mathbf{x}} - \mathbf{b}\|_1 \mid \tilde{\mathbf{x}} \geq \mathbf{0}\}, \\
P_{\text{distr}}^{\text{feas}} &= \min\{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 \mid \mathbf{0} \leq \mathbf{x} \leq C\boldsymbol{\xi}\}.
\end{aligned}$$

The objective function in each problem measures how close $\mathbf{A}\mathbf{x}$ is to $\mathbf{b}$ for a given nonnegative solution $\mathbf{x}$, and the optimal value measures the minimum total infeasibility, as measured by the lowest attainable ℓ_1 distance between $\mathbf{A}\mathbf{x}$ and $\mathbf{b}$. Note that an optimal value of zero for a given problem implies that the feasible region contains a solution $\mathbf{x}$ that satisfies $\mathbf{A}\mathbf{x} = \mathbf{b}$. With a slight abuse of notation, let us use $v(P^{\text{feas}})$, $v(P_J^{\text{feas}})$ and $v(P_{\text{distr}}^{\text{feas}})$ to denote the optimal objective value of each problem. We then have the following result.

Proposition 10 *Let C be a nonnegative constant. For any $\delta \in (0, 1)$, with probability at*

least $1-\delta$ *over the sample* J*,*

$$v(P_J^{feas}) \leq v(P_{distr}^{feas}) + \frac{C}{\sqrt{K}} \cdot m \cdot \|\mathbf{A}\|_{\max} \cdot \left(1 + \sqrt{2 \log \frac{1}{\delta}}\right).$$

The proof of Proposition 10 (see Section C.1.2) follows using a similar but simpler procedure than those used in the proofs of Theorems 11 and 12. The guarantee in Proposition 10 has a similar interpretation to Theorems 11 and 12: the magnitude of the total infeasibility of the columns J is bounded with high probability by the minimum infeasibility of the distributional counterpart $P_{\text{distr}}^{\text{feas}}$ plus a $O(1/\sqrt{K})$ term.

Interpretation of γ and χ:

We first note that the technique of bounding the objective value of a linear program using the ℓ_∞ norm of basic feasible solutions has been applied previously in the literature [128, 75]. The presence of γ and χ in Theorem 11 and 12, respectively, arises due to the use of sensitivity analysis results from linear programming with respect to the right-hand side vector $\mathbf{b}$. As shown in Section C.1.1, we show that any optimal solution $\mathbf{x}^{*0}$ of problem P_{distr} has a sparse counterpart $\mathbf{x}'$ in the space $\mathcal{S}_J \equiv \{\mathbf{x} \mid x_j = 0 \ \forall j \notin J\}$ such that it is in the vicinity of $\mathbf{x}^{*0}$ in terms of Euclidean distance. However, $\mathbf{x}'$ does not necessarily belong to the feasible set $\mathcal{F}(P_J)$ of the column-randomized linear program P_J, since $\mathcal{F}(P_J)$ is a subset of $\mathcal{S}_J$. To relate the optimal objective value $v(P_J)$ of problem P_J to $\mathbf{c}^T\mathbf{x}'$, which is close to $\mathbf{c}^T\mathbf{x}^{*0}$, we use sensitivity analysis arguments which involve either γ or χ.

Comparison of Theorems 11 and 12:

While both Theorem 11 and 12 provide valid bounds for the optimality gap $\Delta v(P_J)$, Theorem 11 is in general easier to apply; indeed, in Section 5.3 we discuss two notable examples where γ can be easily computed (specifically, LPs with totally unimodular constraint matrices $\mathbf{A}$ and infinite horizon discounted Markov decision processes). For problems that are not standard form LPs, neither guarantee directly applies, but we can obtain specialized guarantees by

carefully modifying a result (Proposition 17 in Section C.1.1) that leads to Theorem 11 and designing bounds for the ℓ_∞ norm of feasible or optimal solutions of D_J (as opposed to basic solutions of D). We will later showcase two examples of such guarantees, for covering LPs (Section 5.3.3) and packing LPs (Section 5.3.4).

With regard to Theorem 12, we expect for most problems that Theorem 12 will be difficult to apply, as it requires a universal bound for the norm of the reduced cost vector for every basis, feasible or not, of problem P. Nevertheless, Theorem 12 is interesting because it involves reduced costs, which are also of importance in column generation. For a basic feasible solution, the reduced cost of a non-basic variable j can be thought of as the rate at which the objective changes as one increases x_j to move from the current basic feasible solution to an adjacent/neighboring basic feasible solution in which j is part of the basis. With this perspective of reduced costs, one can informally interpret the result in the following way: if χ is small, then the rate at which the objective changes between adjacent basic feasible solutions is small. In such a setting, it is reasonable to expect that there will be many basic feasible solutions that are close to being optimal and that solving the sampled problem P_J should return a solution that performs well. On the other hand, if there exist non-optimal basic feasible solutions where the reduced cost vector has a very large magnitude (which would imply a large χ), then this would suggest that the objective changes by a large amount between certain adjacent basic feasible solutions, and that there are certain "good" columns that are more important than others for achieving a low objective value. In this setting, we would expect the sampled problem objective $v(P_J)$ to only be close to $v(P)$ if J includes the "good" columns, which would be unlikely to happen in general.

Design of Randomization Scheme ρ:

The quantity ξ_j, which is the probability that the jth column is drawn by the randomization scheme ρ, can be interpreted as the relative importance of x_j compared to other components of $\mathbf{x} \in \mathbb{R}^n$ in the complete problem P; indeed, when the corresponding column is randomly

chosen, x_j is allowed to be nonzero, and can thus be utilized to solve the optimization problem. For example, in a network flow optimization problem, x_j represents the amount of flow over edge j; a nonzero ξ_j can thus be interpreted as the belief that edge j should be used for flow. As another example, consider the LP formulation of an MDP, where each component of $\mathbf{x}$ corresponds to a state-action pair (s, a) (i.e., $x_{(s,a)}$ is the expected discounted frequency of the system being in state s and action a being taken). In this setting, a nonzero $\xi_{(s,a)}$ can be interpreted as the relative importance of (s, a) to other state-action pairs.

One can design the randomization scheme based on prior knowledge of the problem. For example, one could use a heuristic solution to a network flow problem to design a randomization scheme ρ resulting in a distribution $\boldsymbol{\xi}$ that is biased towards this heuristic solution. Similarly, if one has access to a good heuristic policy for an MDP, one can design a distribution $\boldsymbol{\xi}$ that is biased towards state-action pairs (s, a) that occur frequently for this policy. If such prior knowledge is not available, a uniform or nearly-uniform distribution over $[n]$ is adequate. We provide two concrete examples on how to design randomization schemes in our numerical experiments in Section 5.5. Finally, we note that the indices in J have been assumed to be i.i.d. It turns out that this assumption can actually be relaxed: in Section 5.4, we derive an upper bound on the optimality gap $\Delta v(P_J)$ for the case when the indices are sampled non-independently.

Minor Remarks on the Upper Bound:

We remark on two other interesting properties of the bound (5.10). First, the second term in (5.10) is independent of the distribution $\boldsymbol{\xi}$; no matter how $\boldsymbol{\xi}$ is designed, the optimality gap $\Delta v(P_J)$ is guaranteed to converge with rate $1/\sqrt{K}$. Second, the dependence of the bound on the confidence parameter δ is via $\sqrt{2\log(2/\delta)}$ in Theorem 11 or $\sqrt{2\log(1/\delta)}$ in Theorem 12. This implies that very small values of δ will not significantly increase the upper bound on $\Delta v(P_J)$.

Computational Strengths and Weaknesses

We compare the column randomization method to the CG method from a computational viewpoint. An obvious characteristic of the CG method is that it is a serial algorithm: to introduce a new column, one needs the dual solution of the restricted problem that consists of columns generated in previous iterations. This sequential nature unfortunately prevents the CG method from being parallelized. In contrast, the column randomization method is amenable to parallelization. Given a collection of processors, each processor can be used to sample a column and compute the constraint and objective coefficients in parallel, until K columns in total are sampled across all processors. This can be especially advantageous in cases where the objective or constraint coefficients require significant effort compute, such as solving a dynamic program or integer program. For example, [12] considers a set partitioning model of a pickup and delivery problem arising in airlift operations, where each decision variable $x_{v,S}$ corresponds to an aircraft v being assigned to a collection of shipments S and the cost coefficient $c_{v,S}$ is the optimal value of a scheduling problem that determines the sequence of pickups and dropoffs of the shipments in S.

An obvious disadvantage of the column randomization method is that it does not guarantee optimality. Even if there exists an optimal solution of the complete problem P that belongs to the feasible set $\mathcal{F}(P_{\text{distr}})$ of problem P_{distr}, the optimality gap still converges with rate $1/\sqrt{K}$, which implies that the "last-mile" shrinkage of the optimality gap requires an increasing number of additional sampled columns. If optimality is a concern, instead of solely using the column randomization method, one could use it as a warm-start for the CG method. Specifically, let $J_{\text{nz}} = \{j \mid \tilde{x}^*_j > 0\}$, where $\tilde{\mathbf{x}}^*$ is the solution returned by Algorithm 8. Then, the set of variables $(x_j)_{J \in J_{\text{nz}}}$ and the columns $\mathbf{A}_{J_{\text{nz}}}$ can be used as the initial solution for the CG method.

5.3 Special Structures and Extensions

In this section, we demonstrate how the results of Sections 5.2 can be applied to LPs with specific problem structures, including LPs with totally unimodular constraints (Section 5.3.1), Markov decision processes (Section 5.3.2), covering problems (Section 5.3.3) and packing problems (Section 5.3.4). In Section 5.3.5, we consider the portfolio optimization problem, which is in general not an LP, but is amenable to the same type of analysis.

5.3.1 LPs with Totally Unimodular Constraints

Consider a linear program with a totally unimodular constraint matrix, i.e., every square submatrix of $\mathbf{A}$ has determinant 0, 1, or -1. Such LPs appear in various applications, such as minimum cost network flow problems and assignment problems [11]. In such problems, it is not uncommon to encounter the situation where the number of variables is much larger than the number of constraints. For example, in a minimum cost network flow problem, each constraint corresponds to a flow-balance constraint at a given node, while each variable corresponds to the flow over an edge; in a graph of n nodes, one will therefore have n constraints and as many as $\binom{n}{2}$ decision variables. We can thus consider solving the problem using the column randomization method. We obtain the following guarantee on the objective value of the column randomization method when applied to linear programs with totally unimodular constraints.

Proposition 11 *Assume the constraint matrix of* $\mathbf{A}$ *of the complete problem* P *is totally unimodular. Define* C, P_{distr}, P_J *and* $\mathbf{A}_J$ *as in Theorem 11. For any* $\delta \in (0,1)$, *with probability at least* $1-\delta$ *over the set* J, *the following holds: if* P_J *is feasible and* $\operatorname{rank}(\mathbf{A}_J) = m$, *then*

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + \frac{C\left(1+m^2\|\mathbf{c}\|_\infty\right)}{\sqrt{K}}\left(1+\sqrt{2\log\frac{2}{\delta}}\right). \tag{5.12}$$

Proof: Any basic solution $\mathbf{p}$ to the dual problem D can be written as $\mathbf{p}^T = \mathbf{c}_B^T\mathbf{A}_B^{-1}$,

where B is a basis. In addition, since $\mathbf{A}$ is totally unimodular, any element of $\mathbf{A}_B^{-1}$ is either 1, -1, or 0. Therefore, the ith component of $\mathbf{p}$ satisfies $p_i = \sum_{j=1}^{m} [\mathbf{A}_B^{-1}]_{ji} (\mathbf{c}_B)_j \leq m \cdot \|\mathbf{c}\|_\infty$ for all $i \in [m]$. Thus, we set $\gamma = m\|\mathbf{c}\|_\infty$. Along with the fact $\|\mathbf{A}\|_{\max} = 1$ for any totally unimodular matrix $\mathbf{A}$, we finish the proof by invoking Theorem 11. □

5.3.2 Markov Decision Processes

Consider a discounted infinite horizon MDP, with n_s states and n_a actions. The cost function $c(s, a)$ represents the immediate cost of taking action a in state s. The transition probability $P_s(s', a)$ represents the probability of being in state s' after taking action a in state s. Let $\theta \in (0, 1)$ be the discount factor. One can solve the MDP by formulating a linear program [81]:

$$\begin{aligned} \underset{\mathbf{x}_1,\ldots,\mathbf{x}_{n_s} \in \mathbb{R}^{n_a}}{\text{minimize}} \quad & \mathbf{c}_1^T \mathbf{x}_1 + \ldots + \mathbf{c}_s^T \mathbf{x}_s + \ldots + \mathbf{c}_{n_s}^T \mathbf{x}_{n_s} \\ \text{such that} \quad & (\mathbf{E}_1 - \theta \mathbf{P}_1)\mathbf{x}_1 + \ldots + (\mathbf{E}_s - \theta \mathbf{P}_s)\mathbf{x}_j + \ldots + (\mathbf{E}_{n_s} - \theta \mathbf{P}_{n_s})\mathbf{x}_{n_s} = \mathbf{1}, \\ & \mathbf{x}_1, \ldots, \mathbf{x}_s, \ldots, \mathbf{x}_{n_s} \geq \mathbf{0}, \end{aligned}$$

where $\mathbf{E}_j$ is a $n_s \times n_a$ matrix such that the jth row is all ones and every other entry is zero. The vector $\mathbf{c}_s$ is of size n_a such that its ath component is equal to $c(s, a)$. The matrix $\mathbf{P}_s$ is of size $n_s \times n_a$ such that its (s', a)-th component represents $P_s(s', a)$. Notice that matrix $\mathbf{P}_s$ is a column stochastic matrix, i.e., $\mathbf{1}^T \mathbf{P}_s = \mathbf{1}^T$ and $\mathbf{P}_s \geq \mathbf{0}$ for all $s \in [n_s]$. The decision variable vector $\mathbf{x}_s$ is of size n_a, where the ath entry represents the expected discounted long-run frequency of the system being in state s and action a being taken. If one sorts the decision variables by actions [127], then the linear program can be re-written as:

$$\underset{\tilde{\mathbf{x}}^1,\ldots,\tilde{\mathbf{x}}^{n_a} \in \mathbb{R}^{n_s}}{\text{minimize}} \quad \tilde{\mathbf{c}}_1^T \tilde{\mathbf{x}}_1 + \ldots + \tilde{\mathbf{c}}_a^T \tilde{\mathbf{x}}_a + \ldots + \tilde{\mathbf{c}}_{n_a}^T \tilde{\mathbf{x}}_{n_a} \tag{5.13a}$$

$$\text{such that} \quad (\mathbf{I} - \theta \tilde{\mathbf{P}}_1)\tilde{\mathbf{x}}_1 + \ldots + (\mathbf{I} - \theta \tilde{\mathbf{P}}_a)\tilde{\mathbf{x}}_a + \ldots + (\mathbf{I} - \theta \tilde{\mathbf{P}}_{n_a})\tilde{\mathbf{x}}_{n_a} = \mathbf{1}, \tag{5.13b}$$

$$\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_a, \ldots, \tilde{\mathbf{x}}_{n_a} \geq \mathbf{0}, \tag{5.13c}$$

where $\tilde{\mathbf{c}}_a = [c(1,a); \ldots; c(s,a); \ldots; c(n_s,a)]$ for $a \in [n_a]$ and $\tilde{\mathbf{P}}_a$ is a $n_s \times n_s$ matrix such that its (s',s)-th element is equal to $P_s(s',a)$. Note that problem (5.13) is a standard form LP and has more columns than rows. We can therefore apply the column randomization method to solve problem (5.13), leading to the following proposition.

Theorem 13 *Consider solving a discounted infinite horizon MDP with n_s states and n_a actions by the column randomization method. Define C, P_{distr}, P_J and $\mathbf{A}_J$ as in Theorem 11. For any $\delta \in (0,1)$, with probability at least $1-\delta$, the following holds: if P_J is feasible and* $\text{rank}(\mathbf{A}_J) = n_s$*, then*

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + \frac{C}{\sqrt{K}} \cdot \left(1 + \frac{n_s \|\mathbf{c}\|_\infty}{1-\theta}\right) \cdot \left(1 + \sqrt{2 \log \frac{2}{\delta}}\right). \tag{5.14}$$

Proof: Similarly to Proposition 11, we prove Theorem 13 by bounding $\|\mathbf{A}\|_{\max}$ and γ. Obviously, $\|\mathbf{A}\|_{\max} \leq 1$. Again, any basic solution $\mathbf{p}$ of the dual has the form $\mathbf{p}^T = \mathbf{c}_B^T \mathbf{A}_B^{-1}$, where B is a basis of the linear program (5.13). Note that $\mathbf{A}_B$ has the form $\mathbf{A}_B = \mathbf{I} - \theta \mathbf{P}$, where $\mathbf{P}$ is an $n_s \times n_s$ matrix such that each of its columns is selected from the columns of $[\tilde{\mathbf{P}}_1, \ldots, \tilde{\mathbf{P}}_{n_a}]$ (see [127]). In addition, a standard property of $\mathbf{A}_B^{-1}$ is that it can be written as the following infinite series:

$$\mathbf{A}_B^{-1} = \mathbf{I} + \theta \mathbf{P} + \theta^2 \mathbf{P}^2 + \cdots = \mathbf{I} + \sum_{n=1}^{\infty} \theta^n \cdot \mathbf{P}^n.$$

Thus, we can bound $\|\mathbf{p}\|_\infty$ as $\|\mathbf{p}^T\|_\infty \leq \|\mathbf{c}_B^T\|_\infty + \sum_{n=1}^{\infty} \theta^n \cdot \|\mathbf{c}_B^T \mathbf{P}^n\|_\infty$. Note that for any $n \in \mathbb{N}$ and vector $\mathbf{v} \in \mathbb{R}^{n_s}$, we have

$$\|\mathbf{v}^T \mathbf{P}^n\|_\infty = \max_{s \in [n_s]} \left| \sum_{s' \in [n_s]} v_{s'} \mathbf{P}^n_{(s',s)} \right| \leq \max_{s \in [n_s]} \sum_{s' \in [n_s]} |v_{s'}| \cdot \mathbf{P}^n_{(s',s)} \leq \|\mathbf{v}\|_\infty \cdot \max_{s \in [n_s]} \sum_{s' \in [n_s]} \mathbf{P}^n_{(s',s)} = \|\mathbf{v}\|_\infty,$$

where $\mathbf{P}^n_{(s',s)}$ is the (s',s)th entry of $\mathbf{P}^n$. Therefore, we obtain that

$$\|\mathbf{p}^T\|_\infty = \|\mathbf{c}_B^T \mathbf{A}_B^{-1}\|_\infty \leq \|\mathbf{c}_B\|_\infty + \sum_{n=1}^{\infty} \theta^n \cdot \|\mathbf{c}_B^T \mathbf{P}^n\|_\infty \leq \|\mathbf{c}_B\|_\infty / (1-\theta) \leq \|\mathbf{c}\|_\infty / (1-\theta).$$

Since $\mathbf{p}$ was an arbitrary basic solution of the complete dual of problem (5.13), we can therefore set $\gamma = \|\mathbf{c}\|_\infty / (1-\theta)$. The rest of the proof follows by an application of Theorem 11. □

5.3.3 Covering Problems

A covering linear program can be formulated as

$$P^{\text{covering}} : \underset{\mathbf{x}}{\text{minimize}} \quad \mathbf{c}^T\mathbf{x} \tag{5.15a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b}, \tag{5.15b}$$

$$\mathbf{x} \geq \mathbf{0}, \tag{5.15c}$$

where $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ are all nonnegative, and we additionally assume that for every $i \in [m]$, there exists a $j \in [n]$ such that $A_{i,j} > 0$. This type of problem arises in numerous applications such as facility location [94]. The column-randomized counterpart of this problem and its dual can be written as

$$\begin{aligned} P_J^{\text{covering}} &: \min\{\mathbf{c}_J^T\tilde{\mathbf{x}} \mid \mathbf{A}_J\tilde{\mathbf{x}} \geq \mathbf{b}, \tilde{\mathbf{x}} \geq \mathbf{0}\}, \\ D_J^{\text{covering}} &: \max\{\mathbf{p}^T\mathbf{b} \mid \mathbf{p}^T\mathbf{A}_J \leq \mathbf{c}_J^T, \mathbf{p} \geq \mathbf{0}\}. \end{aligned}$$

Although P^{covering} is not a standard form LP, it is straightforward to extend Proposition 17 to this problem, leading to the following result. We omit the proof for brevity.

Proposition 12 *Let C be a nonnegative constant and define $P_{distr}^{covering}$ as*

$$P_{distr}^{covering} \equiv \min\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{0} \leq \mathbf{x} \leq C\boldsymbol{\xi}\}.$$

For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sample J, the following holds: if $P_J^{covering}$ is feasible, then

$$\Delta v(P_J^{covering}) \leq \Delta v(P_{distr}^{covering}) + \frac{C}{\sqrt{K}} \cdot (1 + \|\mathbf{p}\|_\infty \cdot m \cdot \|\mathbf{A}\|_{\max}) \cdot \left(1 + \sqrt{2\log\frac{2}{\delta}}\right)$$

for any optimal solution $\mathbf{p}$ of $D_J^{covering}$.

To now use this result, we need to be able to bound $\|\mathbf{p}\|_\infty$ for any solution $\mathbf{p}$ of any dual D_J^{covering} of the column-randomized problem. Let us define the quantity U^{covering} as

$$U^{\text{covering}} = \max_{i,j}\left\{\frac{c_j}{A_{i,j}} \;\middle|\; A_{i,j} > 0\right\}.$$

We then have the following result.

Theorem 14 *Let C and $P^{covering}_{distr}$ be defined as in Proposition 12. For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sample J, the following holds: if $P^{covering}_J$ is feasible, then*

$$\Delta v(P^{covering}_J) \leq \Delta v(P^{covering}_{distr}) + \frac{C}{\sqrt{K}} \cdot (1 + U^{covering} \cdot m \cdot \|\mathbf{A}\|_{\max}) \cdot \left(1 + \sqrt{2 \log \frac{2}{\delta}}\right).$$

The proof (see Section C.1.3) follows by showing that U^{covering} is a bound on $\|\mathbf{p}\|_\infty$ for any feasible solution $\mathbf{p}$ of the dual D^{covering}_J, for any J such that P^{covering}_J is feasible. (Note that the bound applies to any feasible solution of D^{covering}_J, not just the optimal solutions of D^{covering}_J.)

5.3.4 Packing Problems

A packing linear program is defined as

$$\begin{aligned} P^{\text{packing}} : \underset{\mathbf{x}}{\text{maximize}} \quad & \mathbf{c}^T\mathbf{x} && \text{(5.16a)} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, && \text{(5.16b)} \\ & \mathbf{x} \geq \mathbf{0}, && \text{(5.16c)} \end{aligned}$$

where we assume that $\mathbf{c} \geq \mathbf{0}$, $\mathbf{b} > \mathbf{0}$, and that $\mathbf{A}$ is such that for every column $j \in [n]$, there exists an $i \in [m]$ such that $A_{i,j} > 0$. Packing problems have numerous applications, such as network revenue management [114].

The column-randomized counterpart of this problem and its dual can be written as

$$\begin{aligned} P^{\text{packing}}_J &: \max\{\mathbf{c}_J^T\tilde{\mathbf{x}} \mid \mathbf{A}_J\tilde{\mathbf{x}} \leq \mathbf{b}, \tilde{\mathbf{x}} \geq \mathbf{0}\}, \\ D^{\text{packing}}_J &: \min\{\mathbf{p}^T\mathbf{b} \mid \mathbf{p}^T\mathbf{A}_J \geq \mathbf{c}_J^T, \mathbf{p} \geq \mathbf{0}\}. \end{aligned}$$

As with covering problems, the packing problem P^{packing} is not a standard form LP, but we can derive a counterpart of Proposition 17 for P^{packing}. Note that in this guarantee, for a problem P' with the same feasible region as P^{packing}, the optimality gap $\Delta v(P')$ is defined as $\Delta v(P') = v(P^{\text{packing}}) - v(P')$, since the complete problem P^{packing} is a maximization problem. As with Proposition 12, the proof is straightforward, and thus omitted.

Proposition 13 *Let C be a nonnegative constant and define $P_{distr}^{covering}$ as*

$$P_{distr}^{packing} \equiv \max\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{0} \leq \mathbf{x} \leq C\boldsymbol{\xi}\}.$$

For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sample J, the following holds: if $P_J^{packing}$ is feasible, then

$$\Delta v(P_J^{packing}) \leq \Delta v(P_{distr}^{packing}) + \frac{C}{\sqrt{K}} \cdot (1 + \|\mathbf{p}\|_\infty \cdot m \cdot \|\mathbf{A}\|_{\max}) \cdot \left(1 + \sqrt{2\log\frac{2}{\delta}}\right)$$

for any optimal solution $\mathbf{p}$ of $D_J^{packing}$.

To obtain a more specific guarantee, define for each i the following quantities:

$$r_i = \max\left\{\frac{c_j}{A_{i,j}} \,\middle|\, A_{i,j} > 0\right\},$$
$$j_i^* = \arg\max_j \left\{\frac{c_j}{A_{i,j}} \,\middle|\, A_{i,j} > 0\right\}.$$

These two quantities can be understood by interpreting each i as a resource constraint, and b_i as the available amount of resource i. The column j_i^* is the column that has the best rate of objective value garnered per unit of resource i consumed, and the quantity r_i is that corresponding rate. Define now W as

$$W = \sum_{i'=1}^{m} r_{i'} b_{i'},$$

and U^{packing} as the maximum over i of W/b_i, i.e.,

$$U^{\text{packing}} = \max_{i\in[m]} \frac{W}{b_i} = \frac{\sum_{i'=1}^m r_{i'} b_{i'}}{\min_{i\in[m]} b_i}.$$

We then have the following specific guarantee for packing LPs.

Theorem 15 *Let C and $P_{distr}^{packing}$ be defined as in Proposition 13. For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sample J, the following holds: if $P_J^{packing}$ is feasible, then*

$$\Delta v(P_J^{packing}) \leq \Delta v(P_{distr}^{packing}) + \frac{C}{\sqrt{K}} \cdot (1 + U^{packing} \cdot m \cdot \|\mathbf{A}\|_{\max}) \cdot \left(1 + \sqrt{2\log\frac{2}{\delta}}\right).$$

The proof of this result (see Section C.1.4) follows by establishing that W is an upper bound on $v(P_J^{\text{packing}})$, and then bounding each $|p_i|$ by solving a modified version of D_J^{packing} which is defined using W. We remark that our choice of W is special only in that it bounds $v(P_J^{\text{packing}})$. For particular packing problems, if one has access to a problem-specific bound W' on $v(P_J^{\text{packing}})$, one could define U^{packing} with W' instead to obtain a more refined bound.

5.3.5 Portfolio Optimization

In this last section, we deviate slightly from our previous examples by showing how our approach can be applied to problems that are not linear programs. The specific problem that we consider is the portfolio optimization problem, which is defined as

$$
\begin{aligned}
P^{\text{portfolio}}: \quad & \underset{\mathbf{x}\in\mathbb{R}^n,\mathbf{r}\in\mathbb{R}^m}{\text{minimize}} \quad f(r_1,\ldots,r_m) && \text{(5.17a)}\\
& \text{such that} \quad \sum_{j=1}^{n} \alpha_{ij} x_j = r_i, \quad \forall i \in [m] && \text{(5.17b)}\\
& \qquad\qquad \sum_{j=1}^{n} x_j = 1, && \text{(5.17c)}\\
& \qquad\qquad \mathbf{x} \geq \mathbf{0}, && \text{(5.17d)}
\end{aligned}
$$

where both $\mathbf{x}$ and $\mathbf{r}$ are decision variables. Problem (5.17) can be interpreted as follows: a decision maker seeks an optimal portfolio, which is a distribution over instruments, according to some objectives. The decision variable x_j represents the fraction of allocation committed to instrument j, the constraint parameter α_{ij} represents the return of instrument j in scenario i, and r_i is the total return in ith scenario. The objective function f is a function measuring the risk of the returns $r_1,\ldots,r_m$. Unlike the optimization problems we discussed so far, we assume that f is any Lipschitz continuous function with Lipschitz constant L, and is not necessarily a linear function of $\mathbf{r}$.

Although problem $P^{\text{portfolio}}$ is not in general a linear program, we can still apply the column randomization method to solve the problem. We describe the procedure in Algorithm 9.

Notice that, unlike Algorithm 8 which samples columns associated with all variables, here we only sample columns associated with $\mathbf{x}$.

Algorithm 9 The Column Randomization Method - Portfolio Optimization

1: Sample K i.i.d. indices in $[n]$ as $J \equiv \{J_1, \ldots, J_K\}$ according to ρ.

2: Solve the sampled optimization problem:

$$P_J^{\text{portfolio}}: \quad \min \left\{ f(\mathbf{r}) \;\middle|\; \sum_{j \in J} \alpha_{ij} \tilde{x}_j = r_i, \ \forall\, i \in [m], \ \sum_{j \in J} \tilde{x}_j = 1, \ \tilde{\mathbf{x}} \geq \mathbf{0} \right\} \tag{5.18}$$

return optimal solution $(\tilde{\mathbf{x}}^*, \mathbf{r}^*)$ and optimal objective value $f(\mathbf{r}^*)$

Proposition 14 *Assume vectors $\boldsymbol{\alpha}_j = (\alpha_{ij})_{i \in [m]}$ in problem $P^{portfolio}$ satisfying $\|\boldsymbol{\alpha}_j\|_2 \leq H$ for all $j \in [n]$. Let $C \geq 1$ be an arbitrary constant and define the optimization problem*

$$P_{distr}^{portfolio}: \quad \min_{\mathbf{x},\mathbf{r}} \left\{ f(\mathbf{r}) \;\middle|\; \sum_{j \in [n]} \boldsymbol{\alpha}_j x_j = \mathbf{r}, \ \mathbf{1}^T \mathbf{x} = 1, \ \mathbf{0} \leq \mathbf{x} \leq C\boldsymbol{\xi} \right\}. \tag{5.19}$$

Denote F, F_{distr}, and F_J as optimal objective values of problems $P^{portfolio}$, $P_{distr}^{portfolio}$, and $P_J^{portfolio}$, respectively. Define $\Delta F_J \equiv F_J - F$ and $\Delta F_{distr} = F_{distr} - F$. For any $\delta \in (0,1)$, with probability at least $1 - \delta$, the following statement holds:

$$\Delta F_J \leq \Delta F_{distr} + \frac{CLH}{\sqrt{K}} \left(1 + 3\sqrt{\frac{1}{2} \log \frac{4}{\delta}} \right). \tag{5.20}$$

For brevity, the proof is relegated to the ecompanion (see Section C.1.5). While the proof is similar to that of Proposition 17 in the construction of a random solution that is close to the solution of the distributional counterpart problem $P_{\text{distr}}^{\text{portfolio}}$, the main difference is that it relies on Lipschitz continuity, rather than LP duality.

It is worthwhile to point out several aspects about this result and the portfolio optimization problem. First, the portfolio optimization problem (5.17) is not required to be a convex optimization problem; the objective function f can be non-convex, so long as it is Lipschitz continuous. Second, this result is related to Theorem 4 in Section 3.3.2, where we

consider the problem of estimating the decision forest choice model, which is a probability distribution over a collection of decision trees. We show that by solving an optimization problem over a random sample of trees, one can obtain a gap on the ℓ_1 training error of the model that decays with rate $1/\sqrt{K}$. Proposition 14 is a generalization of that result to more general optimization problems outside of choice model estimation, and allows for objective functions more general than those based on ℓ_1 distance.

5.4 Statistically-Dependent Columns

So far we have assumed that each column in the column-randomized linear program is sampled independently. In this section, we show how this assumption can be relaxed. We state our main performance guarantee in Section 5.4.1. In Section 5.4.2, we consider a specific non-i.i.d. column sampling scheme – *groupwise sampling* – which has natural applications in problems such as Markov decision processes, and apply our guarantee from Section 5.4.1 to this sampling scheme.

5.4.1 Guarantees via Dependency Graph and Forest Complexity

We begin by assuming that the randomization scheme ρ is such that $j_1, \ldots, j_K$ still follow the distribution $\boldsymbol{\xi}$, i.e., $\mathbf{Pr}[j_k = t] = \xi_t$ for $k \in [K]$ and $t \in [n]$, but they are no longer independent. Thus, the indices $j_1, \ldots, j_K$ are no longer an i.i.d. sample from $\boldsymbol{\xi}$, and we require a different set of tools to analyze Algorithm 8 and $\Delta v(P_J)$ in this setting.

To analyze the column randomization method, we will make use of a specific concentration inequality from [79], which requires specifying the dependence structure of a collection of random variables through a specific type of graph. We thus begin by briefly defining the relevant graph-theoretic concepts.

Given an undirected graph G, we use $V(G)$ to denote the vertices of G, and $E(G)$ to denote the edges of G. Given two vertices $u, v \in V(G)$, the edge between u and v is denoted

by $\langle u, v\rangle$. We say that u and v are adjacent if $\langle u, v\rangle \in E(G)$. We say that u and v are non-adjacent if they are not adjacent. For two sets of nodes $U, V \subseteq V(G)$, we say that U and V are non-adjacent if u and v are non-adjacent for every $u \in U$ and $v \in V$. Lastly, a graph G is a *forest* if it does not contain any cycles, and is a *tree* if it does not contain any cycles and consists of a single connected component.

With this definitions, we now define the dependency graph, which is a representation of the dependency structure within a collection of random variables.

Definition 8 (Dependency graph) *An undirected graph G is called a* dependency graph *of a set of random variables $X_1, X_2, \ldots, X_K$ if it satisfies the following two properties:*

1. $V(G) = [K]$.
2. *For every $I, J \subseteq [K]$, $I \cap J = \emptyset$ such that I and J are non-adjacent, $\{X_i\}_{i\in I}$ and $\{X_j\}_{j\in J}$ are independent.*

We now introduce the concept of a forest approximation from [79].

Definition 9 (Forest approximation, [79]) *Given a graph G, a forest F, and a mapping $\phi : V(G) \to V(F)$, we say that (ϕ, F) is a* forest approximation *of G if, for any $u, v \in V(G)$ such that $\langle u, v\rangle \in E(G)$, either $\phi(u) = \phi(v)$ or $\langle \phi(u), \phi(v)\rangle \in E(F)$.*

In words, a forest approximation is a mapping of a general graph G to a smaller forest F that is obtained by merging nodes in G. For a given node $v \in V(F)$, the set $\phi^{-1}(v)$ corresponds to the set of nodes in $V(G)$ that were merged to obtain the node v. Using the notion of a forest approximation, we can now define the forest complexity of a graph G.

Definition 10 (Forest complexity, [79]) *Let $\Phi(G)$ denote the set of all forest approximations of G. Given a forest approximation (ϕ, F), define $\lambda_{(\phi,F)}$ as*

$$\lambda_{(\phi,F)} = \sum_{\langle u,v\rangle \in E(F)} \left(|\phi^{-1}(u)| + |\phi^{-1}(v)|\right)^2 + \sum_{i=1}^{k} \min_{u \in V(T_i)} |\phi^{-1}(u)|^2$$

where $T_1, \ldots, T_k$ is the collection of trees that comprise F. We call $\Lambda(G) = \min_{(\phi,F)\in\Phi(G)} \lambda_{(\phi,F)}$ the forest complexity *of G.*

The forest complexity $\Lambda(G)$ quantifies how much the graph G looks like a forest. Notice that $\Lambda(G) \geq |V(G)|$ for any graph G. In practice, we only need an upper bound on $\Lambda(G)$, rather than its exact value; we refer readers to [79] for several examples on how $\Lambda(G)$ can be bounded.

Given a dependency graph G for the random indices in the set J, we now bound the optimality gap of the column-randomized linear program.

Theorem 16 *Let C be a nonnegative constant, define P_{distr} as in Theorem 11 and assume the random indices in J follow the dependency graph G with forest complexity $\Lambda(G)$. For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sample J, the following holds: if P_J is feasible and $\operatorname{rank}(\mathbf{A}_J) = m$, then*

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + C \cdot (1 + m\gamma\|\mathbf{A}\|_{\max}) \cdot \left(\sqrt{\frac{K + 2|E(G)|}{K^2}} + \sqrt{\frac{2\Lambda(G)\log(2/\delta)}{K^2}} \right), \tag{5.21}$$

where γ and $\|\mathbf{A}\|_{\max}$ are defined as in Theorem 11.

Under the same conditions, with probability at least $1-\delta$ over the sample J, the following holds: if P_J is feasible and $\operatorname{rank}(\mathbf{A}_J) = m$, then

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + C \cdot \chi \cdot \left(\sqrt{\frac{K + 2|E(G)|}{K^2}} + \sqrt{\frac{2\Lambda(G)\log(1/\delta)}{K^2}} \right), \tag{5.22}$$

where χ is defined as in Theorem 12.

The proof (see Section C.1.6) follows by utilizing the McDiarmid inequality for dependent random variables from [79]. We note that Theorem 16 is a generalization of Theorems 11 and 12. If $j_1, j_2, \ldots, j_K$ are independent, then the dependency graph G has no edges, and thus $|E(G)| = 0$ and $\Lambda(G) = K$. Therefore, when each column is generated independently, the upper bounds in Theorem 16 are equivalent to the bounds in Theorem 11 and 12.

5.4.2 Groupwise Column Sampling

In many linear programs, we can naturally rearrange and group related columns together. For example, in the LP formulation of an MDP, one can collect columns associated with state s into a set $\mathcal{G}(s)$; the collection of all columns is simply the disjoint union $\bigcup_{s=1}^{n_s} \mathcal{G}(s)$, where n_s is number of states in the MDP and each $\mathcal{G}(s) = \{(s, a) \mid a \in [n_a]\}$. For such a problem, sampling $J = \{j_1, \ldots, j_K\}$ independently from the complete collection of columns, i.e., from $[n_s] \times [n_a]$, may not be attractive. The reason for this is that we may sample the columns in such a way that we do not sample any columns corresponding to a particular state $\tilde{s}$; in such a scenario, the sampled problem P_J will automatically be infeasible.

In the presence of a natural group structure of the columns, rather than sampling columns in total across all n columns, one could consider sampling n_r columns from each group. In the MDP example, this would correspond to sampling n_r columns (which correspond to state-action pairs) for each state s. The resulting column-randomized linear program P_J corresponds to an MDP where there is a random set of n_r actions out of the complete set of n_a actions available in each state s. Most importantly, P_J is guaranteed to be feasible.

It turns out that our results for dependent columns can be used to study column-randomized LPs where columns are sampled by groups. We refer to such a mechanism as a *groupwise randomization scheme* and define it formally below.

Definition 11 (Groupwise Randomization Scheme) *Assume the set of indices $[n]$ can be organized into $n_{\mathcal{G}}$ groups, i.e., $[n]$ is the disjoint union of sets $\mathcal{G}_g$ for $g = 1, 2, \ldots, n_{\mathcal{G}}$. Consider a randomization scheme ρ such that (i) it samples indices in n_r rounds of sampling; (ii) in each round, it samples $n_{\mathcal{G}}$ indices as follows: for $i = 1, \ldots, n_{\mathcal{G}}$, it first uniformly at random chooses an index g_i from $[n_{\mathcal{G}}] \setminus \{g_j \mid j \in [i-1]\}$ then samples an index from group $\mathcal{G}_{g_i}$ according to a distribution $\boldsymbol{\xi}^{g_i}$. We refer to such a randomization scheme ρ as a* groupwise randomization scheme.

Note that the randomization scheme ρ samples $K = n_r n_{\mathcal{G}}$ indices in total, and samples

n_r columns in each group. By design, each random index j follows the distribution $\boldsymbol{\xi}$, whose probabilities are given by

$$\xi_t \equiv \mathbf{Pr}\left[j = t\right] = \frac{1}{n_{\mathcal{G}}} \sum_{g \in [n_{\mathcal{G}}]} \mathbb{I}\{t \in \mathcal{G}_g\} \cdot \xi_t^g = \frac{1}{n_{\mathcal{G}}} \cdot \xi_t^{\mathcal{G}(t)}$$

where $\mathcal{G}(t)$ is the group to which column $t \in [n]$ belongs to.

By using our general result for dependent columns (Theorem 16), we obtain a specific guarantee for column-randomized LPs obtained by groupwise randomization schemes.

Theorem 17 *Let J be a sample of $K = n_r n_{\mathcal{G}}$ indices sampled according to a groupwise randomization scheme ρ. Let C be a nonnegative constant and define P_{distr} as in Theorem 11. For any $\delta \in (0,1)$, with probability at least $1-\delta$, the following holds: if P_J is feasible and* $\operatorname{rank}(\mathbf{A}_J) = m$, *then*

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + \frac{C\left(1 + m\gamma\|\mathbf{A}\|_{\max}\right)}{\sqrt{n_r}} \left(1 + \sqrt{2\log\frac{2}{\delta}}\right),$$

where γ and $\|\mathbf{A}\|_{\max}$ are defined as in Theorem 11. Under the same assumption, with probability at least $1-\delta$, the following holds: if P_J is feasible and $\operatorname{rank}(\mathbf{A}_J) = m$, *then*

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + \frac{C \cdot \chi}{\sqrt{n_r}} \left(1 + \sqrt{2\log\frac{1}{\delta}}\right),$$

where χ is defined as in Theorem 12.

Proof: The dependency graph G of $K = n_r n_{\mathcal{G}}$ random indices that are sampled by ρ consists of n_r cliques of size $n_{\mathcal{G}}$; Figure 5.1 provides an example of the dependency graph for $n_r = 3$ and $n_{\mathcal{G}} = 4$. Therefore, $|E(G)| = n_r n_{\mathcal{G}}(n_{\mathcal{G}} - 1)/2$ and $\Lambda(G) \leq \lambda(\phi, F) = n_r n_{\mathcal{G}}^2$ for a forest approximation (ϕ, F) that maps each clique in G as a node in F. By upper bounding $\Lambda(G)$ by $n_r n_{\mathcal{G}}^2$ in Theorem 16, and using the fact that $K = n_r n_{\mathcal{G}}$, we complete the proof. □

Theorem 17 can be interpreted as a guarantee on the optimality gap as a function of the number of columns sampled *per group*: for a groupwise randomization scheme, the

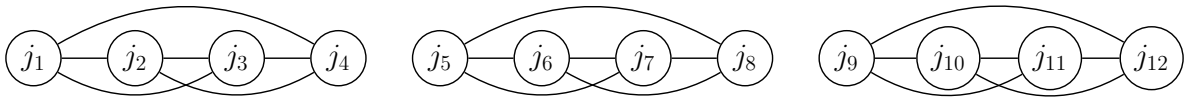


Figure 5.1: Dependency graph of random indices sampled by the groupwise randomization scheme with $n_{\mathcal{G}} = 4$ and $n_r = 3$.

gap decreases at a rate of $1/\sqrt{n_r}$, where n_r is the number of columns sampled per group. Compared to Theorem 11 and 12, the rate of convergence in Theorem 17 in terms of the *total* number of columns sampled, which is $K = n_r n_{\mathcal{G}}$, is slower; Theorem 11 and 12 both have a rate of $1/\sqrt{K}$, while Theorem 17 has a rate of $1/\sqrt{n_r} \equiv \sqrt{n_{\mathcal{G}}/K}$.

5.5 Numerical Experiments

In this section, we apply the column randomization method to two applications of large-scale linear programs that are commonly solved by CG. We demonstrate the effectiveness of the column randomization method by comparing its performance to that of the CG method. We also use these two applications to show that how one can design a randomization scheme based on the problem structure. All linear and mixed-integer programs in this section are formulated in the Julia programming language [17] with the JuMP package [47] and solved by Gurobi [93].

5.5.1 Cutting-Stock Problem

The first application we consider is the classic cutting-stock problem. We follow the notation in [14] and for completeness, briefly review the problem. A paper company needs to satisfy a demand of b_i rolls of paper of width w_i, for each $i \in [m]$. The company has supply of large rolls of paper of width W such that $W \geq w_i$ for $i \in [m]$. To meet the demand, the company slices the large rolls into smaller rolls according to *patterns*. A pattern is a vector of nonnegative integers $(a_1, a_2, \ldots, a_m)$ that satisfies $\sum_{i=1}^{m} a_i w_i \leq W$, where each a_i is the

number of rolls of width w_i to cut from the large roll. Let n be the number of all feasible patterns and let $(a_{1j}, a_{2j}, \ldots, a_{mj})$ be the jth pattern for $j \in [n]$. Let $\mathbf{A}$ be the matrix such that $A_{ij} = a_{ij}$ for $i \in [m]$ and $j \in [n]$. The *cutting-stock problem* is to minimize the number of large rolls of papers used while satisfying the demand, which can be formulated as the following covering LP:

$$P^{\text{CS}}: \quad \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad \sum_{j=1}^{n} x_j \tag{5.23a}$$

$$\text{such that} \quad \sum_{j=1}^{n} a_{ij}x_j \geq b_i, \quad \forall i \in [m], \tag{5.23b}$$

$$x_j \geq 0, \qquad \forall j \in [n]. \tag{5.23c}$$

Explicitly representing the constraint matrix $\mathbf{A}$ in full is usually impossible: the number of feasible patterns n can be huge even if the number of demanded widths m is small. A typical solution method is column generation, in which each iteration proceeds as follows. Given a set of patterns $J = \{j_1, j_2, \ldots, j_K\}$, solve the restricted problem $P^{\text{CS}}(J)$: $\underset{\tilde{\mathbf{x}}\in\mathbb{R}^K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \tilde{x}_k \mid \sum_{k=1}^{K} \mathbf{A}_{j_k}\tilde{x}_k \geq \mathbf{b}, \tilde{\mathbf{x}} \geq \mathbf{0} \right\}$ and let $\mathbf{p}$ be the optimal dual solution. Then find a new pattern j_{K+1} such that the corresponding new column has the most negative reduced cost $1 - \mathbf{p}^T\mathbf{A}_{j_{K+1}}$. If the reduced cost is nonnegative, the current solution is optimal and the procedure terminates; otherwise, we add j_{K+1} to the collection J and repeat the procedure. The problem of finding the column with the most negative reduced cost is equivalent to solving the following subproblem:

$$P^{\text{CS-sub}}: \quad \underset{\mathbf{a}}{\text{maximize}} \quad \sum_{i=1}^{m} p_i^* a_i \tag{5.24a}$$

$$\text{such that} \quad \sum_{i=1}^{m} w_i a_i \leq W, \tag{5.24b}$$

$$a_i \in \mathbb{N}^+, \qquad \forall i \in [m], \tag{5.24c}$$

where $\mathbb{N}^+$ is the set of nonnegative integers; if the optimal value $v(P^{\text{CS-sub}})$ is smaller than 1, then we terminate the column generation procedure; otherwise, we let pattern j_{K+1} correspond to the optimal solution of $P^{\text{CS-sub}}$ and add it to J.

Instead of column generation, we can consider solving the cutting-stock problem by the column randomization method. In our implementation of the column randomization method, we consider the randomization scheme described in Algorithm 10. The randomization scheme essentially starts with an empty pattern, i.e., $(a_1, \ldots, a_m) = (0, \ldots, 0)$ and at each iteration, it increments a_i for a randomly chosen i, while ensuring that it does not run out of unused width. We note that Algorithm 10 is not the only way to sample columns of $\mathbf{A}$, and one can consider other randomization schemes that would lead to potentially better performance of the column randomization method. Our intention here is to provide a simple example of how one can design a randomization scheme based on problem structure.

Algorithm 10 Sampling a Column for the Cutting-Stock Problem

Column $\mathbf{a}$ is a zero vector of length m and $\zeta \leftarrow W$.
while $\zeta > 0$ **do**
    $I \leftarrow \{i \mid w_i \leq \zeta\}$.
    if $|I| \geq 1$ **then**
        Sample an index i uniformly at random from I.
        Update $a_i \leftarrow a_i + 1$ and $\zeta \leftarrow \zeta - w_i$.
    else
        Break the while loop
return Column $\mathbf{a}$.

In Figure 5.2, we illustrate the performance of column-randomized linear programs for the cutting-stock problem with respect to number of columns $K \in \{2 \times 10^4, 4 \times 10^4, 6 \times 10^4, 8 \times 10^4\}$ and number of required widths $m \in \{1000, 2000, 4000\}$. We note that the value of m significantly affects size and complexity of the problem: as m increases, there are more possible patterns and thus n increases as well. For the CG approach, m defines the number of integer variables in the subproblem (5.24); as it increases, the subproblem becomes more challenging. We set $W = 10^5$; we draw each w_i uniformly at random from $\{W/10, W/10 + 1, \ldots, W/4 - 1, W/4\}$ without replacement; and we draw each b_i independently uniformly

at random from $\{1, \ldots, 100\}$. We measure the performance of column-randomized linear programs P_J^{CS}, where each column is obtained by Algorithm 10, by its relative optimality gap $\Delta v(P_J^{\mathrm{CS}})/v(P^{\mathrm{CS}})$. For each value of m and K, we run the column-randomized method 20 times and compute the average optimality gap, which is plotted in Figure 5.2. Before continuing, we note here that there are many ways to randomly generate cutting-stock instances. Our goal is not to exhaustively evaluate the numerical performance of the column randomization method on every possible family of instances, but rather to understand its performance on a reasonably general set of instances.

We first observe that the curves in Figure 5.2 approximately match the convergence rate of $1/\sqrt{K}$ in Theorems 11 and 12. In addition, the speed of convergence significantly slows down after the optimality is smaller than 2%; see the curve for $m = 1000$. Second, as the problem size increases, we need more samples to return comparable performance in terms of optimality gap. This is reflected by the fact that for a fixed number of columns K, the optimality gap is larger for larger value of m.

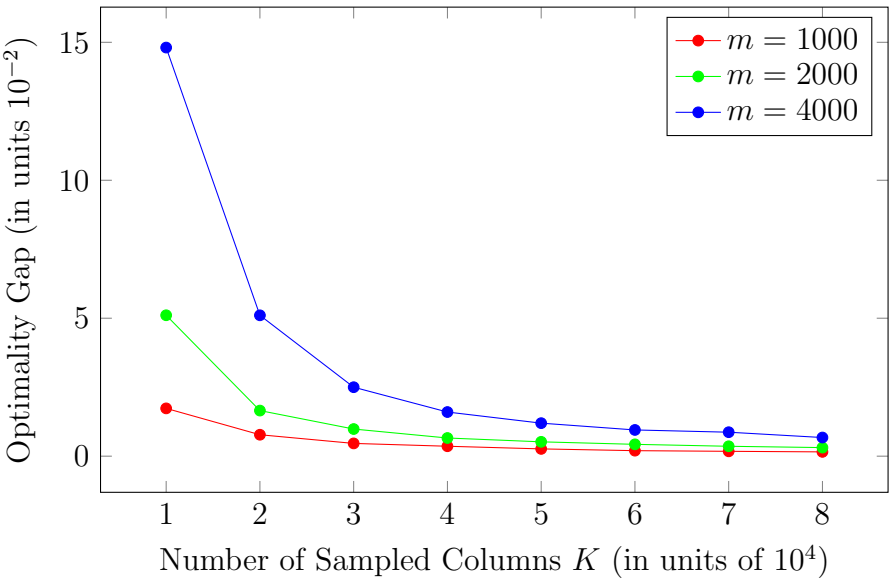


Figure 5.2: Performance of the column randomization method on the cutting-stock problem with respect to number of columns K and number of required widths m.

We further compare the runtime of the column randomization method to that of the CG method in Table 5.1. The first column of the table indicates the value of m, which quantifies the problem size and subproblem complexity. The second column indicates the number of sampled columns K in the column-randomized linear program. The third and fourth columns indicate relative optimality gap $\Delta v(P^{\mathrm{CS}}(J))/v(P^{\mathrm{CS}})$ and runtime of the column randomization method, respectively; for both of these metrics, we report the average over 20 runs of the column-randomized method. The fifth column shows the time required by the CG method to reach the same (average) relative optimality gap. We also list the total duration for CG (i.e., the time required for CG to reach a 0% optimality gap) in the fifth column, and denote it by "(total)".

Table 5.1 shows that, when the problem is small ($m = 1000$), the column randomization method returns a high-quality solution with an optimality gap below 1%, within 30 seconds and with 2×10^4 sampled columns. Doubling or tripling the number of sampled columns does not significantly improve the performance, as the optimality gap is already small. Meanwhile, CG also works well when $m = 1000$, obtaining the optimal solution in a reasonable time (within fifteen minutes). On the other hand, when the problem is large ($m = 4000$), the runtime of CG dramatically increases, as it needs almost 5000 seconds (just under 1.5 hours) to reach a 5% optimality gap. The computational limiting factor comes from solving the subproblem, which becomes more difficult as m increases. On the other hand, the column randomization method only needs ten minutes to reach a 1% optimality gap. This demonstrates the value of solving linear programs by the column randomization method in lieu of CG when the subproblem is intractable. If one requires perfectly optimal solutions (gap of 0%), one can use the result of the column randomization method as an initial warm-start solution for the column generation approach. In the case of $m = 4000$, if one uses the result of column-randomization method with $K = 4 \times 10^4$ as a warm start, the runtime of the column generation method could potentially be reduced by more than 50%.

Demand Types (m)	Columns (K)	Optimality Gap (%)	Runtime (s)	CG Runtime (s)
1000	2×10^4	0.78	28.4	365.5
	4×10^4	0.36	56.4	411.7
	6×10^4	0.20	89.3	456.4
	8×10^4	0.16	122.5	475.1
				(total) 775.4
2000	2×10^4	1.65	58.9	1330.6
	4×10^4	0.65	120.1	1622.8
	6×10^4	0.43	197.9	1732.2
	8×10^4	0.31	287.6	1805.0
				(total) 2932.92
4000	2×10^4	5.10	139.4	4979.8
	4×10^4	1.59	314.2	7175.2
	6×10^4	0.95	527.1	7670.1
	8×10^4	0.68	768.6	7940.0
				(total) 13336.1

Table 5.1: Performance of the column randomization method on the cutting stock problem for different problem sizes and numbers of sampled columns.

5.5.2 Nonparametric Choice Model Estimation

The second problem we consider is nonparametric choice model estimation, which is a modern application of large-scale linear programming and CG. In particular, we consider estimating the ranking-based choice model from data [52, 88, 120]. In this model, we assume that a retailer offers N different products, indexed from 1 to N. We use the index 0 to represent the no-purchase alternative, which is always available to customer. Together, we refer to the set $[N]^+ \equiv \{0, 1, 2, \ldots, N\}$ as the set of purchase options. A ranking-based choice model $(\boldsymbol{\Sigma}, \boldsymbol{\lambda})$ consists of two components. The first component $\boldsymbol{\Sigma}$ is a collection of rankings over options $[N]^+$, in which each ranking represents a customer type. We use $\sigma(i)$ to indicate the rank of option i, where $\sigma(i) < \sigma(j)$ implies that i is more preferred to j under the ranking σ. When a set of products $S \subseteq [N]$ is offered, a customer of type σ selects option i from the set $S \cup \{0\}$ with the lowest rank, i.e., the option $\arg\min_{i \in S \cup \{0\}} \sigma(i)$. The second component $\boldsymbol{\lambda}$ is a probability distribution over rankings in the set $\boldsymbol{\Sigma}$; the element λ_σ can be interpreted as the probability that a random customer would make decisions according to ranking σ.

To estimate a ranking-based model, we utilize data in the form of past sales rate information. Here we consider the type of data described in [52]; we refer readers to that paper for more details. Assume that the retailer has provided M assortments $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$ in the past, where each $S_m \subseteq [N]$. For each assortment S_m, the retailer observes the choice probability $v_{i,m}$ for assortment S_m and option i, which is the fraction of past transactions in which a customer chose i, given that assortment S_m was offered. We let $v_{(i,m)} \equiv 0$ if $i \notin S \cup \{0\}$.

The estimation of a ranking-based choice model $(\boldsymbol{\Sigma}, \boldsymbol{\lambda})$ can be formulated in the form of problem $P^{\text{portfolio}}$ (Section 5.3.5). We first notice that there are in total $(N+1)!$ rankings over $[N]^+$, which we enumerate as $\sigma_1, \sigma_2, \ldots, \sigma_{(N+1)!}$. We let the kth column of the problem correspond to ranking σ_k, for $k \in [(N+1)!]$. We use $\alpha_{(i,m),k}$ to indicate whether a customer following ranking σ_k would choose option k when offered assortment S_m. The estimation

problem can then be written as

$$P^{\text{EST}}: \quad \underset{\boldsymbol{\lambda},\hat{\mathbf{v}}}{\text{minimize}} \quad \mathcal{D}(\hat{\mathbf{v}}, \mathbf{v}) \tag{5.25a}$$

$$\text{such that} \quad \sum_{k=1}^{(N+1)!} \alpha_{(i,m),k} \cdot \lambda_k = \hat{v}_{(i,m)}, \quad \forall m \in [M], i \in [N]^+, \tag{5.25b}$$

$$\sum_{k=1}^{(N+1)!} \lambda_k = 1, \tag{5.25c}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \tag{5.25d}$$

where $\hat{\mathbf{v}}$ and $\mathbf{v}$ are vectors of $\hat{v}_{(i,m)}$ and $v_{(i,m)}$ values, respectively, for $i \in [N]^+$ and $m \in [M]$. The function $\mathcal{D}$ measures the error between the predicted choice probabilities $\hat{\mathbf{v}}$ and the actual choice probabilities $\mathbf{v}$. We follow [88] and set $\mathcal{D} = \|\hat{\mathbf{v}} - \mathbf{v}\|_1$, which has Lipschitz constant $\sqrt{M(N+1)}$.

We notice that even if N is merely 10, problem P^{EST} has nearly 4×10^7 columns. Given that problem P^{EST} may have an intractable number of columns, [88] and [120] applied CG to solve the problem. Alternatively, we can apply the column randomization method. We consider the randomization scheme described in Algorithm 11, where we first randomly generate a ranking (line 2) and then map its decision under each assortment to form a column (lines 3-5). Before continuing, we pause to make three important remarks. First, we note that sampling a ranking uniformly at random (line 2) requires minimal computational effort, and can be done by a single function call in most programming languages. Second, we also note that while in Algorithm 10 we directly sample the coefficients of a column, in Algorithm 11 we instead first sample the underlying "structure" of the column (a ranking) then obtain the corresponding coefficients; this illustrates the problem-specific nature of the randomization scheme. Lastly, we note that the paper of [52] considered a linear program for computing the worst-case revenue of an assortment, which is effectively the minimization of a linear function of $\boldsymbol{\lambda}$ subject to constraints (5.25b)–(5.25d). The paper considered a solution method for this problem based on sampling constraints in the dual (which is equivalent to sampling columns in the primal), but did not compare this approach to column generation,

which will do shortly.

Algorithm 11 Sampling a Column for the Ranking Estimation Problem

1: Initialize $\alpha_{(i,m)} \leftarrow 0$ for $i \in [N]^+$ and $m \in [M]$.

2: Sample a ranking/permutation $\sigma : [N]^+ \to [N]^+$ uniformly at random.

3: **for** $m \in [M]$ **do**

4: $\quad i^* \leftarrow \arg\min_{i \in S_m \cup \{0\}} \sigma(i)$.

5: $\quad \alpha_{(i^*,m)} \leftarrow 1$

return Column $\boldsymbol{\alpha} = (\alpha_{(i,m)})_{i \in [N]^+, m \in [M]}$.

We compare the performance of the column randomization method to that of CG with the following experiment setup. We assume that customers follow multinomial logit (MNL) model to make decision, that is, the choice probability $v_{i,m}$ follows

$$v_{i,m} = \frac{\exp(u_i)}{1 + \sum_{j \in S_m} \exp(u_j)}$$

for a given assortment S_m, where each parameter u_i represents the expected utility of product i. We choose each $u_i \sim U[0, 1]$, i.e., uniformly at random from interval $[0, 1]$. We also choose the set of historical assortments $\mathcal{S} = \{S_1, \ldots, S_M\}$ uniformly at randomly from all possible 2^N assortments of N products. We examine the performance of the column randomization method under various problem sizes, using different values of N and M. For the CG method, we use the method in [88], and solve the subproblem as an IP using the formulation from [120].

Table 5.2 shows the performance of the column randomization method. The first two columns of the table indicate the problem size. The third column shows the number of sampled columns. The fourth and fifth columns display the optimality gap and the runtime, respectively; for both of these metrics, we report the average value of the metric over 20 runs of the column randomization method. The sixth column denotes the duration of the CG method to reach the same (average) optimality gap as the column randomization method. We remark that the optimal objective value $v(P^{\text{EST}})$ is always zero, since random utility

maximization models such as the MNL model can be represented as ranking-based models [20]. Thus, instead of showing relative optimality gap as in Table 5.1, we directly show the objective value of the column-randomized linear program in Table 5.2.

In all cases listed in Table 5.2, the column randomization method outperforms the CG method by a large margin. It only requires a fraction of the runtime of the CG method to reach the same optimality level. In particular, when $(N, M) = (10, 150)$, the column randomization method only needs three seconds to reach the optimal objective value, which is zero, while the CG method needs over ten thousand seconds (almost three hours). In real-world applications, the number of products N is usually significantly larger than 10. In those cases, the advantage of column randomization will be even more pronounced. We note that in the IP formulation of the CG subproblem, the number of binary variables scales as $O(N^2 + NM)$. Thus, as N increases, the subproblem quickly becomes intractable. (We additionally note that [120] showed this subproblem to be NP-hard.)

N	M	Columns (K)	Objective	Runtime (s)	CG Runtime (s)
6	50	500	0.05	0.03	20.58
		1000	0.00	0.07	30.44
8	50	500	0.13	0.10	52.32
		1000	0.00	0.12	88.25
8	100	500	0.92	0.21	120.14
		1000	0.07	0.45	414.43
		1500	0.00	0.66	632.23
10	50	500	0.27	0.17	11.93
		1000	0.00	0.22	282.78
10	100	500	1.60	0.28	240.23
		1000	0.40	0.53	774.66
		1500	0.06	0.71	1423.71
		2000	0.00	1.57	2234.52
10	150	500	2.91	0.69	507.63
		1000	0.98	1.07	1399.22
		1500	0.43	1.33	2635.36
		2000	0.18	2.01	4524.72
		2500	0.00	3.14	10143.93

Table 5.2: Performance of the column randomization method on the estimation problem P^{EST} under varying problem sizes and numbers of sampled columns.

CHAPTER 6

Conclusion

In this thesis, we have studied following question: how can we model non-rational purchase choice from a data-driven perspective to create value? We have attempted to answer this question in the following, concrete settings:

1. We proposed the decision forest model, which can model any discrete choice behavior, regardless of whether it belongs to the RUM class or not. Given data in the form of a collection of historical assortments, we proved that simple trees, whose depth scales logarithmically and leaf complexity scales linearly with the number of assortments, are sufficient to fit the data. We further proposed two practical estimation methods for learning the decision forest model from historical assortments. Through experiments with real data, we showed that the decision forest model generally outperforms other rational and non-rational models in out-of-sample prediction in the presence of non-rational customer behavior, and can be used to generate insights on the complementarity/substitution behaviors between products.

2. We developed a mixed-integer optimization methodology for solving the assortment optimization problem when the choice model is a decision forest model. This methodology allows a firm to find optimal or near optimal assortments given a decision forest model, which is valuable due to the ability of the decision forest model to capture non-rational customer behavior. We developed three different formulations of increasing strength. We analyzed the solvability of the Benders decomposition subproblem for each formulation under integral and fractional master solution. Using synthetic

data, we showed that our formulations can be solved directly to optimality or near optimality at a small to medium scale, and using our Benders approach, we show that it is possible to solve large instances with up to 3000 products to a low optimality gap within an operationally feasible timeframe.

3. We generalized an estimation method of decision forest model as a solution method for large-scale linear programs. The method involves simply randomly sampling a collection of K columns from the constraint matrix and then solving the corresponding problem. We developed two performance guarantees for the solution one obtains from this approach, one involving a bound on dual solution and one involving a bound on reduced costs, and showed how these guarantees and the overall approach can be applied to specific problems, such as LPs with totally unimodular constraints, Markov decision processes and covering problems. In numerical experiments with the cutting stock problem and the nonparametric choice model estimation problem, we showed that the proposed approach can obtain near-optimal solutions in a fraction of the computational time required by column generation.

APPENDIX A

Appendix to Chapter 3

A.1 Omitted Proofs

A.1.1 Proof of Proposition 1

For each ranking σ_j, $j = 1, 2, \ldots, m$, we can write down its preference order explicitly as $\sigma_j = \{p_1^{(j)} \succ p_2^{(j)} \succ \ldots \succ p_{K_j}^{(j)} \succ 0\}$, where $a \succ b$ denotes that a is preferred to b; for this ranking, K_j products are preferred over the no-purchase option, and product $p_1^{(j)}$ is the most preferred. Assume each ranking σ_j has probability weight λ_j. Now we construct the forest F as follows: for $j = 1, \ldots, m$, we build a decision tree t_j with the structure shown in Figure A.1. Additionally, we associate tree t_j with probability λ_j. Note that ranking $\sigma_j = \{p_1^{(j)} \succ p_2^{(j)} \succ \ldots \succ p_{K_j}^{(j)} \succ 0\}$ and the decision tree in Figure A.1 give the same decision process: if product $p_1^{(j)}$ is in the assortment, we buy it; otherwise, if product $p_1^{(j)}$ is not in the assortment but $p_2^{(j)}$ is, we buy $p_2^{(j)}$; otherwise, if both $p_1^{(j)}$ and $p_1^{(j)}$ are not in assortment but $p_3^{(j)}$ is, we buy $p_3^{(j)}$; and so on. Therefore, for any option o and any assortment S, we have

$$\begin{aligned}\mathbf{P}^{(F,\lambda)}(o \mid S) &= \sum_{j=1}^{m} \lambda_j \cdot \mathbb{I}\{o = \hat{A}(S, t_j)\} \\ &= \sum_{j=1}^{m} \lambda_j \cdot \mathbb{I}\{\text{ranking } \sigma_j \text{ selects option } o \text{ given } S\} = \mathbf{P}^{(\Sigma,\boldsymbol{\lambda})}(o \mid S),\end{aligned}$$

where the second equality comes from the fact that ranking σ_i and tree t_i makes same decision given S, and the third equality is from the definition of the ranking-based model. □

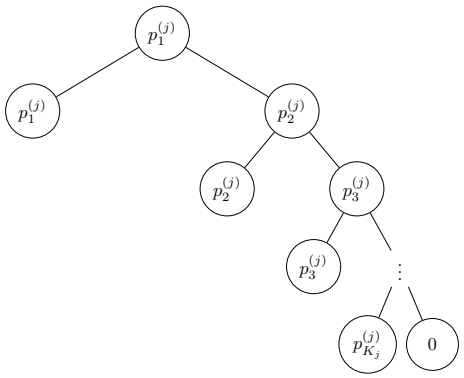


Figure A.1: Ranking $\sigma_j = \{p_1^{(j)} \succ p_2^{(j)} \succ \ldots \succ p_{K_j}^{(j)} \succ 0\}$ can be represented as a purchase decision tree.

A.1.2 Proof of Theorem 1

We prove the theorem by constructing a forest F consisting of $(N+1)^{2^N}$ balanced trees. We first define the common structure of each tree in the forest. Each tree t in the forest has depth $N+1$ and shares the following structure for first the N levels:

$$x_s = 1, \quad \forall\ s \in \textbf{splits}(t) \text{ such that } \text{dist}(r(t), s) = 0, \tag{A.1}$$

$$x_s = 2, \quad \forall\ s \in \textbf{splits}(t) \text{ such that } \text{dist}(r(t), s) = 1, \tag{A.2}$$

$$x_s = 3, \quad \forall\ s \in \textbf{splits}(t) \text{ such that } \text{dist}(r(t), s) = 2, \tag{A.3}$$

$$\vdots$$

$$x_s = N, \quad \forall\ s \in \textbf{splits}(t) \text{ such that } \text{dist}(r(t), s) = N - 1, \tag{A.4}$$

In words, the root split node $r(t)$ checks for the existence of product 1 in the assortment; the split nodes in the second level of the tree (those with $\text{dist}(r(t), s) = 1$) check for product 2; the split nodes in the third level check for product 3; and so on, all the way to the Nth level, at which all splits node check for product N. Figure A.2 provides an example of this tree structure for $N = 3$. In this tree, the left-most leaf node corresponds to assortment $S = \{1, 2, 3\}$, the second leaf node from the left corresponds to assortment $S = \{1, 2\}$, and so on, until the right-most leaf which corresponds to the empty assortment ($S = \emptyset$).

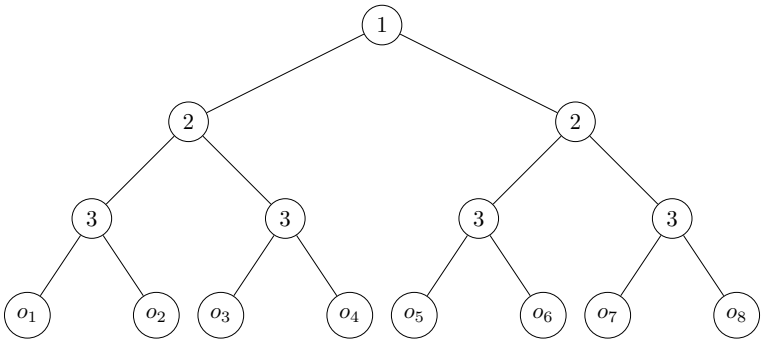


Figure A.2: An example of the structure of the trees in the forest needed for the proof of Theorem 1 for $N = 3$.

In this tree structure, there are exactly 2^N leaf nodes, which we will index from left to right as $l_1, \ldots, l_{2^N}$. Note that each leaf node of a tree in F has a one-to-one correspondence with one of the 2^N possible assortments of the products.

To specify the leaves, we require some additional definitions. Let us denote the 2^N possible assortments of the N products by $S_1, S_2, \ldots, S_{2^N}$, in correspondence with the leaf nodes $l_1, \ldots, l_{2^N}$, respectively. For the leaves $\ell_1, \ldots, \ell_{2^N}$, we use $o_1, o_2, \ldots, o_{2^N}$ to denote the purchase decisions associated with those leaves, and we use $\mathbf{o} = (o_1, \ldots, o_{2^N})$ to denote the 2^N-tuple of purchase decisions. Let $\mathbf{o}^1, \ldots, \mathbf{o}^{(N+1)^{2^N}}$ be a complete enumeration of the $(N+1)^{2^N}$ possible 2^N-tuples of the purchase decisions from $\prod_{i=1}^{2^N} \mathcal{N}_+$.

With these definitions, we define our forest F as consisting of $(N+1)^{2^N}$ trees, where each tree follows the structure in equations (A.1)–(A.4), each tree is indexed from $t = 1$ to $t = (N+1)^{2^N}$ and the purchase decisions of the leaves in tree t are given by the tuple $\mathbf{o}^t$ as defined above. We define the probability distribution $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{(N+1)^{2^N}})$ by defining the probability λ_t of each tree t as:

$$\lambda_t = \prod_{j=1}^{2^N} \mathbf{P}(o_j^t \mid S_j). \tag{A.5}$$

It is straightforward to verify that $\boldsymbol{\lambda}$ is nonnegative and sums to one. Before continuing, we note that not all of the trees defined in this way will satisfy Requirement 2 from Definition 6, which requires that the purchase decision in each leaf must be consistent with the products

along the path of splits from the root node to the leaf; in other words, for some trees o_j^t will not be contained in S_j. However, for any tree where this is the case, by the definition in equation (A.5), the corresponding λ_t will be zero because $\mathbf{P}(o_j^t \mid S_j)$ is zero whenever $o_j^t \notin S_j$. Thus, those trees can be safely removed from F without changing the overall model.

We now show that the decision forest model $(F, \boldsymbol{\lambda})$ outputs the same choice probabilities as the true model $\mathbf{P}(\cdot \mid \cdot)$. For any assortment S_i and option o, we have

$$
\begin{aligned}
\mathbf{P}^{(F,\lambda)}(o \mid S_i) &= \sum_{t=1}^{(N+1)^{2^N}} \lambda_t \cdot \mathbb{I}\{o = \hat{A}(S_i, t)\} \\
&= \sum_{t\,:\,o_i^t = o} \lambda_t \\
&= \sum_{t\,:\,o_i^t = o} \prod_{i'=1}^{2^N} \mathbf{P}(o_{i'}^t \mid S_{i'}) \\
&= \mathbf{P}(o \mid S_i) \cdot \sum_{o_1} \mathbf{P}(o_1 \mid S_1) \cdots \sum_{o_{i-1}} \mathbf{P}(o_{i-1} \mid S_{i-1}) \cdot \sum_{o_{i+1}} \mathbf{P}(o_{i+1} \mid S_{i+1}) \cdots \sum_{o_{2^N}} \mathbf{P}(o_{2^N} \mid S_{2^N}) \\
&= \mathbf{P}(o \mid S_i),
\end{aligned}
$$

where the first equality follows from how choice probabilities under the decision forest model are defined in equation (3.1); the second follows from how our forest is constructed and how the $\mathbf{o}^t$ tuples are defined; the third equality follows from the definition of $\boldsymbol{\lambda}$; the forth by algebra; and the last by recognizing that the choice probabilities for a given assortment must sum to one. □

A.1.3 Proof of Theorem 2

Before we prove Theorem 2, we establish three useful lemmas. Some of the results will be also used in the proof of Theorem 3 in Section A.1.4.

Lemma 1 *For any decision tree t that satisfies Requirements 1 and 2 in Definition 6, there exists a purchase decision tree t' such that: (1)Tree t' satisfies Requirement 1, 2 and 3; (2)*

Tree t and t' have same purchase decision under any given assortment; and (3) $\text{Depth}(t') \leq \text{Depth}(t)$ *and* $|\mathbf{leaves}(t')| \leq |\mathbf{leaves}(t)|$.

Proof of Lemma 1: Let t be a tree that does not satisfy Requirement 3, and consider the following procedure:

1. Select any leaf $\ell \in \mathbf{leaves}(t)$ for which Requirement 3 is violated. Let $path(\ell) \equiv \{s_1, s_2, \ldots, s_{d-1}\}$ be the sequence of splits from root $r(t) \equiv s_1$ to s_{d-1}, which is the parent node of ℓ. Let $i \in \mathcal{N}$ be a product encountered at least two times as the decision process traverses $path(\ell)$. Let $s_{u_1}, s_{u_2}, \ldots, s_{u_{e_i}}$, where $u_1 < u_2 < \cdots < u_{e_i}$, be the subsequence of split nodes associated with product i in $path(\ell)$. (Note that $e_i \geq 2$.)

2. If $s_{u_1} \in \mathbf{LS}(\ell)$, then remove all right subtrees branched at $s_{u_2}, \ldots, s_{u_{e_i}}$ from t. Otherwise, if $s_{u_1} \in \mathbf{RS}(\ell)$, then remove all left subtrees branched at $s_{u_2}, \ldots, s_{u_{e_i}}$ from t.

3. Delete split nodes $s_{u_2}, \ldots, s_{u_{e_i}}$ from tree t, and "glue" the remaining pieces by setting $s_{u_j - 1}$ as the parent node of the remaining child node of s_{u_j} for $j = 2, \ldots, e_i$.

Consider now applying steps 1-3 repeatedly to tree t, until all of the leaves in the tree satisfy Requirement 3; let the resulting tree be denoted by t'. Note that we are guaranteed to terminate with such a tree, because each time we apply steps 1-3, we delete at least one subtree from the tree (and thus at least one leaf), and the tree contains finitely many leaves. In addition to Requirement 3, steps 1-3 also preserve Requirements 1 and 2. Thus, tree t' satisfies condition (1) of the lemma.

Note also that for a given tree, the tree we obtain after applying steps 1-3 is equivalent to that initial tree, in that any assortment is mapped by the two trees to the same purchase decision. This is true because the leaves within the subtrees that are removed are leaves that are unreachable (i.e., there does not exist an assortment that can reach them). For

example, if $s_{u_1} \in \mathbf{LS}(\ell)$, then any leaf in the right subtree branched at $s_{u_2}, \ldots, s_{u_{e_i}}$ is such that product i must be in the assortment and not in the assortment in order to reach the leaf, which is impossible. Thus, tree t' satisfies condition (2) of the lemma.

Lastly, since steps 1-3 involve deleting subtrees and splits and re-attaching the disconnected pieces, tree t' is no deeper than tree t and has no more leaves than tree t, thus verifying condition (3) of the lemma. □

Lemma 1 plays an important role in the proofs of Theorem 2 and 3. In fact, we prove both theorems by directly constructing decision forest models. However, the trees in the forest may violate Requirement 3 in Definition 6. If such violation happens on a tree t, we will use Lemma 1 to find an equivalent tree t' that satisfies Requirement 3 without increasing either the depth or the number of leaves, and replace t by t' in the forest. For convenience, we summarize the procedure in Lemma 1 as the following algorithm.

Our next result, Lemma 2 states that a data set of a single assortment can be perfectly fit by a depth 2 decision forest.

Lemma 2 *For any dataset $\mathcal{S}$ consisting of only one assortment S, there exists a forest F of depth 2 and of leaf complexity 2 and a probability distribution $\boldsymbol{\lambda}$ over F such that $\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$ for every $o \in \mathcal{N}^+$.*

Proof of Lemma 2: Consider a forest F of depth 2 such that $F = \{t_1, t_2, \ldots, t_N, t_0\}$, where each tree is as shown in Figure A.3, and define the probability distribution $\boldsymbol{\lambda}$ so that $\lambda_{t_o} = v_{o,S}$ for each $o \in \mathcal{N}^+$; by construction, $\sum_{t \in F} \lambda_t = 1$ and $\lambda_t \geq 0$ for each $t \in F$. For this forest, each option $o \in \mathcal{N}^+$ is chosen by exactly one tree, t_o, and the probability mass of that tree is $v_{o,S}$, which establishes that $\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$ for all $o \in \mathcal{N}^+$. □

Lemma 3 *Consider two sets of assortments $\mathcal{S}_1$ and $\mathcal{S}_2$ satisfying the following two conditions: (1) for $i = 1, 2$, there exists a forest F_i of depth at most d_i and of leaf complexity at*

Algorithm 12 Modifying a tree of Requirements 1-2 to satisfy Requirement 3 in Definition 6. (Section 3.1.1).

procedure TREEMODIFICATION(a tree t satisfying Requirement 1 and 2)
  while there exists a leaf $\ell \in$ **leaves**(t) violates Requirement 3 **do**
    Set $path(\ell) \leftarrow \{s_1, s_2, \ldots, s_{d-1}\}$ (see its definition in proof of Lemma 1).
    Set $i \leftarrow \arg\max_{k \in \mathcal{N}} \sum_{j=1}^{d-1} \mathbb{I}\{k = x_{s_j}\}$.
    Set $\{u_1 < u_2 < \ldots < u_{e_i}\} \leftarrow \{j \mid x_{s_j} = i, j \in \{1, 2, \ldots, d-1\}\}$.
    if $s_{u_1} \in$ **LS**(ℓ) **then**
      Remove all right subtrees branched at $s_{u_2}, \ldots, s_{u_{e_i}}$ from tree t.
    else
      Remove all left subtrees branched at $s_{u_2}, \ldots, s_{u_{e_i}}$ from tree t.
    for $w = e_i, e_i - 1, \ldots, 2$ **do**
      Delete split node s_w from tree t.
      Set remaining child node of s_w as the child node of s_{w-1}.
  return t

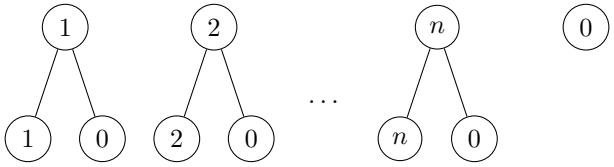


Figure A.3: Trees $t_1, t_2, \ldots, t_N$ and t_0 (shown from left to right) for the forest F described in Lemma 2

most L_i, and a probability distribution $\boldsymbol{\lambda}^{(i)}$ such that $\mathbf{P}^{(F_i,\boldsymbol{\lambda}^{(i)})}(o \mid S) = v_{o,S}$ for all $S \in \mathcal{S}_i$; and (2)there exists a product p such that $p \in S$ for all $S \in \mathcal{S}_1$ and $p \notin S$ for all $S \in \mathcal{S}_2$.

Then there exists a probability distribution $\boldsymbol{\lambda}$ and a forest F of depth at most $1 + \max\{d_1, d_2\}$ and of leaf complexity at most $L_1 + L_2$ such that $\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$ for all $S \in \mathcal{S}_1 \cup \mathcal{S}_2$.

Proof of Lemma 3: We prove the statement by constructing an appropriate forest F and a probability distribution $\boldsymbol{\lambda}$ such that $\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$ for all $S \in \mathcal{S}_1 \cup \mathcal{S}_2$. For $i = 1, 2$, let us denote the trees in forest F_i by $t_1^{(i)}, t_2^{(i)}, \ldots, t_{n_i}^{(i)}$ and the corresponding probability distribution by $\boldsymbol{\lambda}^{(i)} = (\lambda_1^{(i)}, \ldots \lambda_{n_i}^{(i)})$. Let us construct the forest F and probability distribution $\boldsymbol{\lambda}$ for $\mathcal{S}_1 \cup \mathcal{S}_2$ as follows. We define the forest F as

$$F = \{t_{\alpha,\beta} \mid \alpha \in \{1, 2, \ldots, n_1\}, \beta \in \{1, 2, \ldots, n_2\}\},$$

where each tree $t_{\alpha,\beta}$ is formed by placing product p at the root node, placing $t_\alpha^{(1)}$ as the left subtree of the root node, and $t_\beta^{(2)}$ as the right subtree of the root node. For the probability distribution $\boldsymbol{\lambda}$ over F, we set the probability of each tree $\lambda_{\alpha,\beta} = \lambda_\alpha^{(1)} \cdot \lambda_\beta^{(2)}$. By construction, $\boldsymbol{\lambda}$ is nonnegative, and adds up to 1, since

$$\sum_{\alpha=1}^{n_1} \sum_{\beta=1}^{n_2} \lambda_{\alpha,\beta} = \sum_{\alpha=1}^{n_1} \sum_{\beta=1}^{n_2} \lambda_\alpha^{(1)} \cdot \lambda_\beta^{(2)} = \left(\sum_{\alpha=1}^{n_1} \lambda_\alpha^{(1)} \right) \left(\sum_{\beta=1}^{n_2} \lambda_\beta^{(2)} \right) = 1.$$

We now show that $(F, \boldsymbol{\lambda})$ ensures that $\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$ for all $S \in \mathcal{S}_1 \cup \mathcal{S}_2$. For any $S \in \mathcal{S}_1$, we know that $p \in S$, and thus each purchase decision tree $t_{\alpha,\beta} \in F$ will immediately

take the left branch at the root node. This implies that the purchase decision of tree $t_{\alpha,\beta}$ will be exactly the same as its left subtree $t_\alpha^{(1)}$ when any $S \in \mathcal{S}_1$ is given. Thus, for any $S \in \mathcal{S}_1$ and $o \in \mathcal{N}^+$:

$$\begin{aligned}
\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) &= \sum_{\alpha=1}^{n_1} \sum_{\beta=1}^{n_2} \lambda_{\alpha,\beta} \cdot \mathbb{I}\{o = \hat{A}(S, t_{\alpha,\beta})\} \\
&= \sum_{\alpha=1}^{n_1} \sum_{\beta=1}^{n_2} \lambda_\alpha^{(1)} \cdot \lambda_\beta^{(2)} \cdot \mathbb{I}\{o = \hat{A}(S, t_\alpha^{(1)})\} \\
&= \sum_{\alpha=1}^{n_1} \lambda_\alpha^{(1)} \cdot \mathbb{I}\{o = \hat{A}(S, t_\alpha^{(1)})\} = \mathbf{P}^{(F_1,\boldsymbol{\lambda}^{(1)})}(o \mid S) = v_{o,S},
\end{aligned}$$

where we recall that $\hat{A}(S,t)$ is the option chosen by tree t when given assortment S.

Similarly, we can also establish that for any $S \in \mathcal{S}_2$ and $o \in \mathcal{N}^+$, each tree $t_{\alpha,\beta}$ will make the same purchase decision as $t_\beta^{(2)}$, and so $\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = \mathbf{P}^{(F_2,\boldsymbol{\lambda}^{(2)})}(o \mid S) = v_{o,S}$. This establishes that $(F, \boldsymbol{\lambda})$ satisfies $\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$ for all $S \in \mathcal{S}_1 \cup \mathcal{S}_2$.

With regard to the depth of F, we observe that each tree in F is built by adding one level to trees from F_1 and F_2, and so the forest F will be of depth at most $1 + \max\{d_1, d_2\}$. With regard to the leaf complexity of F, each tree in F is built by combining two subtrees, where one has at most L_1 leaves and the other has at most L_2 leaves, so the forest F will have at most $L_1 + L_2$ leaves. Finally, trees in F may violate Requirement 3 in Definition 6. In that case, we apply the procedure in Lemma 1 (Algorithm 12) to find equivalent trees without increasing either the depth or leaf complexity of the forest. □

Proof of Theorem 2: With regard to depth and leaf complexity, we prove the statement by induction on the number of assortments M. The base case is established by Lemma 2. Assume the statement holds for all integers $k < M$. With M historical assortments $S_1, S_2, \ldots, S_M$, let p be a product included in at least one assortment and meanwhile not included in all assortments. Such a p must exist, otherwise $S_1 = S_2 = \ldots = S_M$, which violates the requirement of distinct assortments. Denote $\mathcal{S}_p = \{S \mid S \in \mathcal{S}, p \in S\}$ as the

collection of historical assortments that include product p, and $\mathcal{S}_p^c = \{S \mid S \in \mathcal{S}, p \notin S\}$ as the collection of historical assortments that do not include product p. We further denote their cardinalities as $M_p = |\mathcal{S}_p|$ and $M_p^c = |\mathcal{S}_p^c|$.

Note that $1 \leq M_p \leq M-1$ and $1 \leq M_p^c \leq M-1$ by the definition of product p. To prove the inductive step, we assume that there exists a forest F_p of depth at most M_p+1 and of leaf complexity at most $2M_p$, and a distribution $\boldsymbol{\lambda}_p$ such that $\mathbf{P}^{(F_p,\boldsymbol{\lambda}_p)}(o \mid S) = v_{o,S}$ for all $o \in \mathcal{N}^+$ and $S \in \mathcal{S}^p$. Similarly, we also assume there exists a forest F_p^c of depth at most $M_p^c + 1$ and of leaf complexity at most $2M_p^c$, and a distribution $\boldsymbol{\lambda}_p^c$ such that $\mathbf{P}^{(F_p^c,\boldsymbol{\lambda}_p^c)}(o \mid S) = v_{o,S}$ for all $o \in \mathcal{N}^+$ and $S \in \mathcal{S}_p^c$. By Lemma 3, there exists a distribution $\boldsymbol{\lambda}$ and a forest F of depth at most $1 + \max\{M_p + 1, M_p^c + 1\} \leq 1 + M$ and leaf complexity at most $2M_p + 2M_p^c = 2M$ such that $\mathbf{P}^{(F,\boldsymbol{\lambda})} = v_{o,S}$ for $S \in \mathcal{S}$.

Let $(F, \boldsymbol{\lambda})$ be the corresponding forest model. We can further use the procedure described in Lemma 1 to ensure that for any tree $t \in F$, any path from root to a leaf will not encounter the same product twice on split nodes. Since applying the procedure described in Lemma 1 (Algorithm 12) would not increase depth and leaf complexity, the resulting trees will again have depth at most $\min\{M+1, N+1\}$ and leaf complexity again at most $2M$.

Let $D^* = \min\{M+1, N+1\}$ and let $F_{D^*,2M}$ be the collection of all decision trees of depth at most $M+1$ and of at most $2M$ leaves. Obviously, $F_{D^*,2M}$ is a finite set. By the above induction proof, we know that the following constraint system has a solution:

$$\sum_{t \in F_{D^*,2M}} \mathbf{A}_{t,S} \lambda_t = \mathbf{v}_S, \quad \forall\, S \in \mathcal{S}, \tag{A.6a}$$

$$\mathbf{1}^T \boldsymbol{\lambda} = 1, \tag{A.6b}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}. \tag{A.6c}$$

The set defined by (A.6) is a polyhedron in standard form and is non-empty. Therefore, by standard linear optimization results (e.g., Corollary 2.2 of Bertsimas and Tsitsiklis 1997) there exists a basic feasible solution $\boldsymbol{\lambda}^*$ to (A.6), which possesses the property that $\lambda_t^* \geq 0$ for $M(N+1)+1$ trees $t \in F_{D^*,2M}$ and $\lambda_t^* = 0$ for all other trees, where $M(N+1)+1$ is

the number of equality constraints in (A.6). Defining F as $F = \{t \in F_{D^*,2M} \mid \boldsymbol{\lambda}^*_t > 0\}$ and $\boldsymbol{\lambda} = (\lambda^*_t)_{t \in F_{D^*,2M}}$, we obtain the required decision forest model. □

A.1.4 Proof of Theorem 3

Proof Strategy

Before diving into the proof, we first demonstrate the basic idea of the proof with a simple example, and then provide an informal overview of the strategy of the proof.

Example 4 *Suppose $\mathcal{S}$ is a collection of $M = 128$ assortments with sufficiently large N. By Theorem 2, there exists a decision forest model of depth at most 129 that perfectly fits $\mathcal{S}$. Now, consider a product p and the two subsets of $\mathcal{S}$ consisting of assortments that contain and do not contain p:*

$$\mathcal{S}_p = \{S \in \mathcal{S} \mid p \in S\},$$
$$\mathcal{S}^c_p = \{S \in \mathcal{S} \mid p \notin S\}.$$

Suppose that the product p is such that $|\mathcal{S}_p| = 64$ and $|\mathcal{S}^c_p| = 64$. By invoking Theorem 2, we obtain separate decision forest models $(F_1, \boldsymbol{\lambda}_1)$ and $(F_2, \boldsymbol{\lambda}_2)$ that respectively fit $\mathcal{S}_p$ and $\mathcal{S}^c_p$ that are of depth 65. By invoking Lemma 2, we can combine the two models into a single decision forest model of depth at most $1 + \max\{65, 65\} = 66$ that perfectly fits the assortment collection $\mathcal{S}$.

What the above example illustrates is that when we can find a product p that perfectly divides the assortment collection $\mathcal{S}$, we can actually fit a model where the depth is at most roughly $M/2$, instead of roughly M. In this example, we stopped after finding one product p that perfectly splits $\mathcal{S}$. However, there is nothing preventing us from repeating the process again with $\mathcal{S}_p$ and $\mathcal{S}^c_p$. If we can repeat the same process again with each of $\mathcal{S}_p$ and $\mathcal{S}^c_p$ – i.e., we find a product p' that perfectly splits $\mathcal{S}_p$, and a product p'' (possibly different from

p') that perfectly splits $\mathcal{S}_p^c$ – then we would be able to obtain a forest of depth at most $1 + \max\{1 + \max\{33, 33\}, 1 + \max\{33, 33\}\} = 35$ (roughly $M/4$).

We can keep repeating the process to obtain a smaller and smaller decision forest model; each time we can find such a splitting product, we reduce the size of the collections by half and we obtain roughly a factor of two reduction in the depth of the forest. This procedure naturally gives rise to a forest of depth $O(\log_2 M)$.

The above procedure assumes that we are always able to find a product that perfectly splits a given subcollection of assortments, that is obtained after some rounds of splitting. If the collection of assortments $\mathcal{S}$ is drawn randomly, then this will not always be possible. In addition, this will also not be possible if a subcollection that we encounter contains an odd number of assortments.

Instead of aiming to divide the collection of assortments $\mathcal{S}$ exactly in half by finding a "perfect" splitting product p, what we can instead hope to do is to split $\mathcal{S}$ almost evenly by finding a "good" splitting product p. To do this, we fix an $\epsilon \in (0, 1)$, and consider the factor $1/(2 - \epsilon)$, which is a number in the interval $(1/2, 1)$. The factor $1/(2 - \epsilon)$ defines how big the two subcollections, $\mathcal{S}_p$ and $\mathcal{S}_p^c$, should be relative to $\mathcal{S}$. In other words, we now look for a product p such that $|\mathcal{S}_p| \leq M/(2 - \epsilon)$ and $|\mathcal{S}_p^c| \leq M/(2 - \epsilon)$. If we succeed in doing this, then we will reduce the size of the collection of assortments by a factor of $1/(2 - \epsilon)$ with each splitting product we find, giving rise to a forest of depth $O(\log_{2-\epsilon} M)$.

The parameter ϵ controls a trade-off between the depth of the ultimate forest and the probability of being able to create that forest. When ϵ is small, the factor $1/(2 - \epsilon)$ will be closer to $1/2$, leading to a large reduction in the size of the subcollections and a small depth. However, the probability of finding a product that results in this split will be small. When we enlarge ϵ, then probability of existence of finding a "good" splitting product p increases, but this comes with a price, because the depth scales like $O(\log_{2-\epsilon} M)$, which is increasing in ϵ.

In the proof of Theorem 3, we essentially use this idea to obtain a bound on the probability of finding a forest of depth at most $\log_{2-\epsilon}(M/M_0) + M_0 + 1$, where M_0 is an integer constant that defines a limit on how small the size of a subcollection of assortments can be before applying Theorem 2. Due to the recursive nature of how the splitting procedure is applied to repeatedly divide the collection of assortments, the probability bound satisfies a recursive inequality: for a given collection of assortments, the bound on the probability of finding a forest of depth at most d that fits $\tilde{\mathcal{S}}$ is bounded by a quantity that involves the same probability bound but corresponds to a forest of depth at most $d-1$ that fits a smaller subcollection of assortments.

Notation

Before we are able to prove the theorem, we require some additional definitions. First, as discussed above, we let $\epsilon \in (0,1)$ be an arbitrary constant and let $M_0 > 1$ be an arbitrary fixed positive integer. As alluded to above in **Proof Strategy**, the integer M_0 will later serve as a stopping point for the partitioning process. That is, when the current collection of assortments is of size M_0 or lower, we stop partitioning assortments and apply the depth bound provided by Theorem 2.

For convenience, we will also use the constant k to denote $k = \epsilon^2/(2(2-\epsilon)^2)$, and the constant β to denote $\beta = 1/(2-\epsilon)$. Note that for all $\epsilon \in (0,1)$, $k > 0$ and $\beta \in (1/2, 1)$. The constant k is a quantity that will appear later in our application of Hoeffding's inequality to bound the probability of finding a good splitting product. The constant β is simply the reduction factor of $1/(2-\epsilon)$ that we desire for the splitting product (described above under **Proof Strategy**); in other words, when we split a collection of assortments, the cardinality of each subcollection should be at most a factor of β of the parent collection. Both k and β are introduced to make the mathematical expressions we encounter later less cumbersome.

Given a number of assortments $M > M_0$, we define the integer $\bar{d}$ as

$$\bar{d}(M, M_0, \epsilon) = \left\lceil \log_{2-\epsilon}\left(\frac{M}{M_0}\right)\right\rceil. \tag{A.7}$$

To understand the meaning of $\bar{d}$, observe that $\beta < 1$. The proof that we will present shortly relies on repeatedly dividing a collection of assortments by selecting a product p such that the subcollection of assortments with p and the subcollection that does not contain p both have cardinality that is at most β of the parent collection. The minimum number of such divisions needed to reach a collection of assortments of size M_0 or lower, starting with a collection of M assortments, is exactly $\bar{d}$. For ease of exposition, we will suppress the arguments of $\bar{d}$, but it should be regarded as a function of the number of assortments in the data set M, as well as the constants M_0 and ϵ.

We define the integer function $q(N, M, M_0)$ as

$$q(N, M, M_0) = \left\lfloor N/\bar{d}\right\rfloor. \tag{A.8}$$

The integer $q(N, M, M_0)$ is interpreted as the number of candidate splitting products that are considered at each level of splitting. The rationale for $q(N, M, M_0)$ is as follows. Since we assume that the M assortments are drawn independently and uniformly from the collection of all 2^N assortments, then for each product $p' \in \mathcal{N}$ and each assortment m, the presence of product p in assortment m, $\mathbb{I}\{p \in S_m\}$, is an independent Bernoulli(1/2) random variable. As described above in **Proof Strategy**, for a given collection of assortments, we need to find a product p that splits that collection into two subcollections that are a factor of $1/(2-\epsilon)$ $(= \beta)$ of the size of the parent collection. For a collection of M' independent random assortments, where each product is included in each assortment independently with probability 1/2, the probability that a single, *fixed* product p can split the collection in this way can be written as a binomial probability (i.e., the number X_p of assortments in the collection of M' assortments that contain product p is a Binomial(M', 1/2) random variable), and readily bounded using Hoeffding's inequality. If instead of considering a fixed product, we consider all N products, then the probability of finding a product p that can split the collection in this way increases.

However, if we condition on the event that there exists a good splitting product among all N products, then we can no longer guarantee that the random variables $\mathbb{I}\{p \in S\}$ are independent Bernoulli(1/2) random variables for any product p and any assortment S within either subcollection that is generated. This is problematic, because if the random variables $\mathbb{I}\{p \in S\}$ are no longer independent Bernoulli(1/2) variables within the subcollections, then we cannot bound the probability of finding a product that splits each subcollection, and we cannot succeed in constructing our bound.

Thus, when we search for a good splitting product, we do not search over all N products. Instead, we search over only $q(N, M, M_0)$ products. In that sense, when we condition on the existence of a good splitting product out of those $q(N, M, M_0)$ products, then we only "contaminate" the $q(N, M, M_0)$ products that we searched over, and we can protect the independent Bernoulli(1/2) nature of the remaining products. Since there are $\bar{d}$ levels of splitting, the size of the candidate splitting set should be such that we do not run out of products, i.e., we need $\bar{d} \cdot q(N, M, M_0) \leq N$. The choice of $q(N, M, M_0)$ as $\lfloor N/\bar{d} \rfloor$ gives us the largest possible size for the set of candidate splitting products without running out of products. As with $\bar{d}$, for ease of exposition we will suppress the arguments of q, but again, it is a quantity that depends on the number of assortments M, as well as M_0 and ϵ.

Given ϵ and M_0, we define the function $g(d)$ as

$$g(d) = (2^{d-M_0-1} - 1) \cdot 2^{-q(kM_0-1)}. \tag{A.9}$$

The function $g(d)$ will later serve as a upper bound of probability.

Main Proof

The event in the statement of the theorem is that there exists a forest of a particular depth that fits a set of assortments that are sampled uniformly from the set of all assortments over a fixed number of products. For an arbitrary number of assortments $M' > 0$ and an arbitrary depth d', let us define $R(M', d')$ to be the event that there exists a forest of depth at most d'

over the universe of N products that fits a set of M' assortments. We now need to carefully define the probability distribution with which we will measure the probability of this event and its complement. We will use F to denote a distribution according to which a collection of M' assortments are drawn. The probability $\mathbb{P}_F(R(M', d')^c)$ is the probability that we do not succeed in finding a decision forest of depth at most d' that fits M' assortments sampled from F. Define $\mathcal{F}(M', N')$ as the set of distributions over collections of M' assortments of the N products, such that at least N' products are sampled independently with probability $1/2$. (To "sample a product p independently with probability $1/2$" means to draw an independent Bernoulli$(1/2)$ random variable that is 1 if the product p is to be included, and 0 if it is not to be included.) We then define the maximum failure probability as

$$Q(M', N', d') = \sup_{F \in \mathcal{F}(M', N')} \mathbb{P}_F(R(M', d')^c). \tag{A.10}$$

Note that in the statement of Theorem 3, M assortments are sampled uniformly at random from the set of all assortments, which is exactly the same as independently sampling each of the N products with probability $1/2$, i.e., each assortment is generated by drawing, for each $p \in \{1, \ldots, N\}$, a Bernoulli$(1/2)$ variable that is 1 if product p is to be included, and 0 if it is not included. The set of distributions $\mathcal{F}(M, N)$ is thus a singleton consisting of exactly this distribution over M assortments.

To prove Theorem 3, we will prove a more general result concerning the maximum failure probability Q. Once we prove this general result, we will show that for specific choices of ϵ and M_0, the forest depth and the corresponding probability of the forest fitting the data will exhibit the asymptotic behavior stated in Theorem 3.

The general result we will prove is stated as follows:

Theorem 18 *Let $M > 0$, $N > 0$. Let $\epsilon \in (0, 1)$ be a fixed constant and M_0 be a positive integer such that $M_0 < M$, and let k, q, $\bar{d}$ and $g(d)$ be defined as above. We then have, for any $M' \leq M$,*

$$Q(M', N,\ \bar{d} + M_0 + 1) \leq g(\bar{d} + M_0 + 1) = (2^{\bar{d}} - 1) \cdot 2^{-q(kM_0 - 1)}. \tag{A.11}$$

We will prove this result by induction. To set up the induction proof, we need to set up two auxiliary results. The first result, Lemma 4, will serve to establish the base case for the induction proof.

Lemma 4 *For any positive integer $M' \leq M_0$, $N' \geq 0$, we have:*

$$Q(M', N', M_0 + 1) \leq g(M_0 + 1). \tag{A.12}$$

Proof of Lemma 4: Let $M'' \leq M' \leq M_0$ be the number of distinct assortments in the collection $\mathcal{S}'$ of M' assortments. Theorem 2 guarantees almost surely that there exists a forest of depth at most $M''+1$ such that $\mathbb{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$ for every o and $S \in \mathcal{S}'$. Since $M'' \leq M_0$, it immediately follows that the forest is of depth at most M_0+1. Thus, the maximum failure probability $Q(M', N', M_0+1)$ will be equal to zero for any $M' \leq M_0$. Since the upper bound $g(M_0 + 1)$ is exactly zero (by the definition of g in equation (A.9)), the lemma follows. $\square$

The second auxiliary result, Lemma 5, will serve to establish the induction hypothesis.

Lemma 5 *Let $N' \geq q$, $d \geq M_0 + 2$ and $M' > 0$. If the collection of inequalities*

$$Q(M'', N' - q, d - 1) \leq g(d - 1), \quad \forall\ M'' \leq \lfloor M'\beta \rfloor \tag{A.13}$$

holds, then we have

$$Q(M', N', d) \leq g(d). \tag{A.14}$$

Proof of Lemma 5: First, let us handle the case when $M' \leq M_0$. Note that in this case, by exactly the same reasoning as in Lemma 4, we automatically have $Q(M', N', d) \leq g(d)$: we

apply Theorem 2 to obtain a forest of depth at most $M'+1$, and since $d \geq M_0+2 \geq M'+1$, this forest is automatically of depth at most d as well. This establishes that $Q(M', N', d) = 0$, and since $g(d) \geq 0$, the statement follows, without any use of the hypothesis (A.13). Thus, in what follows, we will focus on the case when $M' > M_0$.

Let F be any distribution from $\mathcal{F}(M', N')$. Let $\mathcal{S}$ be the set of M' assortments drawn from F. Fix a set of products Ξ of size q from the set of products of size at least N' that are known to be independent. Define the set Ξ^* as

$$\Xi^* = \left\{ p \in \Xi \;\middle|\; \begin{array}{l} M'(1-\beta) \leq |\mathcal{S}_p| \leq M'\beta, \\ M'(1-\beta) \leq |\mathcal{S}_p^c| \leq M'\beta \end{array} \right\}, \tag{A.15}$$

where the collections $\mathcal{S}_p$ and $\mathcal{S}_p^c$ are defined as in the proof of Theorem 2:

$$\mathcal{S}_p = \{S \in \mathcal{S} \,|\, p \in S\}, \tag{A.16}$$

$$\mathcal{S}_p^c = \{S \in \mathcal{S} \,|\, p \notin S\}. \tag{A.17}$$

In words, $\mathcal{S}_p$ is the collection of assortments that include product p, while $\mathcal{S}_p^c$ is the collection of assortments that do not include product p. The set Ξ^* is the set of all products p that essentially divide the collection of assortments $\mathcal{S}$ in a "balanced" way, such that the resulting collections of assortments $\mathcal{S}_p$ and $\mathcal{S}_p^c$ contain at least $(1-\beta)$ fraction of the assortments and at most β fraction of the assortments. That is to say, such $p \in \Xi^*$ will be a "good" splitting product, as described in **Proof Strategy**.

With Ξ^* defined, let us define the product p^* as

$$p^* = \begin{cases} \min_{p \in \Xi} p & \text{if } \Xi^* = \emptyset, \\ \min_{p \in \Xi^*} p & \text{if } \Xi^* \neq \emptyset. \end{cases}$$

In words, the product p^* is the lowest index product from Ξ^* if the latter turns out to not be empty, and otherwise it is the lowest index product from Ξ. Both p^* and Ξ^* are random. Note that the definition of p^* when Ξ^* is empty is not important for the proof; it is just needed to ensure that some events we will construct shortly are well-defined.

Having defined p^*, let us now define the following events:

- A: the event that $\Xi^* \neq \emptyset$.
- B_1: the event that there exists a decision forest model $(F_1, \boldsymbol{\lambda}_1)$ of depth at most $d-1$ such that $\mathbf{P}^{(F_1,\boldsymbol{\lambda}_1)}(o \mid S) = v_{o,S}$ for all o and $S \in \mathcal{S}_{p^*}$.
- B_2: the event that there exists a decision forest model $(F_2, \boldsymbol{\lambda}_2)$ of depth at most $d-1$ such that $\mathbf{P}^{(F_2,\boldsymbol{\lambda}_2)}(o \mid S) = v_{o,S}$ for all o and $S \in \mathcal{S}^c_{p^*}$.

Observe that if all three events hold, then Lemma 3 guarantees that there exists a decision forest model $(F, \boldsymbol{\lambda})$ of depth at most $1 + \max\{d-1, d-1\} = d$ such that $\mathbf{P}^{(F,\boldsymbol{\lambda})}(o \mid S) = v_{o,S}$ for all o and $S \in \mathcal{S}$. In other words, if the events A, B_1 and B_2 occur, then the event $R(M', d)$ occurs.

By taking the contrapositive of the above statement, we can bound the probability $\mathbb{P}_F(R(M', d)^c)$ as

$$\begin{aligned}
\mathbb{P}_F(R(M', d)^c) &\leq \mathbb{P}_F(\ (A \cap B_1 \cap B_2)^c\ ) \\
&= \mathbb{P}_F(A^c \cup B_1^c \cup B_2^c) \\
&= \mathbb{P}_F(A^c) + \mathbb{P}_F(A \cap (B_1^c \cup B_2^c)) \\
&\leq \mathbb{P}_F(A^c) + \mathbb{P}_F(B_1^c \cup B_2^c \mid A) \\
&\leq \mathbb{P}_F(A^c) + \mathbb{P}_F(B_1^c \mid A) + \mathbb{P}_F(B_2^c \mid A), \qquad \text{(A.18)}
\end{aligned}$$

where the fourth step follows by the definition of conditional probability and the last step follows by the union bound.

At this stage, we pause to comment on the utility of bounding $\mathbb{P}_F(R(M', d')^c)$ using the events A, B_1 and B_2, as in inequality (A.18). The event A is useful because it is defined in terms of the product set Ξ. Each product p in Ξ is such that p exists in an assortment in $\mathcal{S}$ with probability $1/2$, independently across the M' assortments and independently of any other product; we will see shortly that this allows us to conveniently bound the probability of A^c. Second, the conditional events $B_1^c \mid A$ and $B_2^c \mid A$ are events that bear a resemblance

to $R(M', d)^c$: both $B_1^c \mid A$ and $B_2^c \mid A$ are events in which we fail to find a decision forest model that fits the assortment sets $\mathcal{S}_{p^*}$ and $\mathcal{S}_{p^*}^c$, respectively. The difference is that while the distribution F is such that at least N' products are sampled independently with probability $1/2$ to generate M' assortments, the distribution according to which $\mathcal{S}_{p^*}$ and $\mathcal{S}_{p^*}^c$ are sampled conditional on A is such that at least $N' - q$ products are sampled independently with probability $1/2$ to generate a random number of assortments that is at most $\beta M'$. We will later leverage this similarity to invoke the induction hypothesis and obtain a bound on $\mathbb{P}_F(B_1^c \mid A)$ and $\mathbb{P}_F(B_2^c \mid A)$.

Our goal now will be to bound each of the three terms in the inequality (A.18). For $\mathbb{P}(A^c)$, observe that we can write this event as

$$\mathbb{P}_F(A^c) = \mathbb{P}_F\left(\bigcap_{p \in \Xi} \{|\mathcal{S}_p| > M'\beta \text{ or } |\mathcal{S}_p| < M'(1-\beta)\}\right).$$

By the assumption that $F \in \mathcal{F}(M', N')$ and that Ξ is a subset of those products which are sampled independently, the random variables $\mathbb{I}\{p \in S\}$ for each $p \in \Xi$ and $S \in \mathcal{S}$ are independent Bernoulli$(1/2)$ random variables. The size of the subcollection $\mathcal{S}_p$ can be written as $|\mathcal{S}_p| = \sum_{S \in \mathcal{S}} \mathbb{I}\{p \in S\}$; thus, the random variables $\{|\mathcal{S}_p|\}_{p \in \Xi}$ are distributed as independent Binomial$(M', 1/2)$ random variables. Letting X_p denote each such binomial random variable, we can bound the probability $\mathbb{P}_F(A^c)$ as

$$\begin{aligned}
\mathbb{P}_F(A^c) &= \mathbb{P}_F\left(\bigcap_{p \in \Xi} \{X_p > M'\beta \text{ or } X_p < M'(1-\beta)\}\right) \\
&= \prod_{p \in \Xi} \mathbb{P}\left(X_p > M'\beta \text{ or } X_p < M'(1-\beta)\right) \\
&\leq \prod_{p \in \Xi} (2e^{-M'k}) \\
&= 2^q e^{-M'kq} \\
&\leq 2^{-q(M'k-1)} \\
&\leq 2^{-q(M_0 k-1)}
\end{aligned}$$

where the second step follows by the independence of the X_p random variables; the third step follows by Hoeffding's inequality for a sum of M' Bernoulli random variables and recognizing that $M'\beta - M/2 = M/2 - M'(1-\beta) = M\epsilon/(2(2-\epsilon))$; the fourth step follows by the fact that $|\Xi| = q$; the fifth step follows by the fact that $2 < e$ and algebra; and the last step by the fact that $q > 0$ and $M_0 < M'$. Before continuing, we draw the reader's attention to the dependence of this bound on q: the bound becomes exponentially smaller with q. In words, when we search over a larger set of candidate splitting products, it becomes easier to find a product that splits $\mathcal{S}$ in a balanced way. Ideally, we would search over all N' products, instead of only q products; however, as we discussed earlier under **Notation** when defining the constant q, we have to limit the size of Ξ to ensure that we preserve independent Bernoulli(1/2) products for later stages of splitting.

To bound $\mathbb{P}_F(B_1^c \mid A)$ and $\mathbb{P}_F(B_2^c \mid A)$, let us first define two conditional random variables, Γ and Γ^c, as

$$\Gamma = |\mathcal{S}_{p^*}| \;\Big|\; A, \tag{A.19}$$

$$\Gamma^c = |\mathcal{S}^c_{p^*}| \;\Big|\; A, \tag{A.20}$$

i.e., Γ is the number of assortments containing p^* given that A occurs, while Γ^c is the number of assortments not containing p^* given that A occurs. We can now write $\mathbb{P}_F(B_1^c \mid A)$ and $\mathbb{P}_F(B_2^c \mid A)$ by conditioning and de-conditioning on Γ and Γ^c respectively:

$$\mathbb{P}_F(B_1^c \mid A) = \mathbb{E}_\Gamma\left[\mathbb{P}_F(B_1^c \mid A, \Gamma)\right], \tag{A.21}$$

$$\mathbb{P}_F(B_2^c \mid A) = \mathbb{E}_{\Gamma^c}\left[\mathbb{P}_F(B_2^c \mid A, \Gamma^c)\right], \tag{A.22}$$

where the expectations are taken with respect to the conditional random variables Γ and Γ^c respectively.

To understand how we will next proceed, let us focus on $\mathbb{P}_F(B_1^c \mid A)$. Let $\tilde{M}$ be a given realization of Γ and consider the conditional probability $\mathbb{P}_F(B_1^c \mid A, \Gamma = \tilde{M})$. We now claim

that:

$$\mathbb{P}_F(B_1^c \mid A, \Gamma = \tilde{M}) = \mathbb{P}_{\tilde{F}}(R(\tilde{M}, d-1)^c) \tag{A.23}$$

$$\leq Q(\tilde{M}, N' - q, d-1), \tag{A.24}$$

where $\tilde{F}$ is the distribution over collections of $\tilde{M}$ assortments that is induced by conditioning on the event A and the event $\Gamma = \tilde{M}$. (To actually sample from such a distribution, one can repeatedly sample M' assortments according to F, discard those draws of the M' assortments that do not satisfy both A and $|\mathcal{S}_{p^*}| = \tilde{M}$, and return each collection $\mathcal{S}_{p^*}$ for the remaining draws.) The first step in the above follows from the definition of $\tilde{F}$. The second step follows by the fact that, by definition, $\tilde{F}$ is a member of $\mathcal{F}(\tilde{M}, N' - q)$. To understand why, observe that after conditioning on A, which is an event that involves a set of products of size q from the set of at least N' products that are known to be independent with respect to F, the distribution $\tilde{F}$ may be such that the independence of the q products is no longer guaranteed. However, after conditioning in this way, we know that there still remain at least $N' - q$ products that are independent, because they have not yet been used in any way (we have not conditioned on any event that involves these products). Thus, we obtain the upper bound of $Q(\tilde{M}, N' - q, d-1)$. This observation is critical, because it is what ultimately allows us to link $Q(M', N', d)$ to $Q(M'', N' - q, d-1)$, and bound $Q(M', N', d)$ via the induction hypothesis.

With this insight in hand, let us continue with bounding $\mathbb{P}_F(B_1^c \mid A)$. We have

$$\mathbb{P}_F(B_1^c \mid A) = \mathbb{E}_\Gamma \left[\mathbb{P}(B_1^c \mid A, \Gamma)\right] \tag{A.25}$$

$$\leq \mathbb{E}_\Gamma \left[Q(\Gamma, N' - q, d-1)\right] \tag{A.26}$$

$$\leq g(d-1) \tag{A.27}$$

$$= (2^{d - M_0 - 2} - 1) 2^{-q(kM_0 - 1)}, \tag{A.28}$$

where the first inequality follows by our reasoning above; the second inequality follows by our induction hypothesis (A.13), and the fact that the integer random variable Γ is almost surely bounded by $\lfloor M'\beta \rfloor$; and the last step follows by the definition of g.

Applying the same steps for $\mathbb{P}_F(B_2^c \mid A)$, allows us to also conclude that

$$\mathbb{P}_F(B_2^c \mid A) \leq (2^{d-M_0-2} - 1)2^{-q(kM_0-1)}. \tag{A.29}$$

Now that we have constructed a bound for $\mathbb{P}_F(A)$, $\mathbb{P}_F(B_1^c \mid A)$ and $\mathbb{P}_F(B_2^c \mid A)$, we can return to completing the bound in (A.18). We have:

$$\begin{aligned}
\mathbb{P}_F(R(M', d)^c) &\leq \mathbb{P}_F(A^c) + \mathbb{P}_F(B_1^c \mid A) + \mathbb{P}_F(B_2^c \mid A) && \text{(A.30)} \\
&\leq 2^{-q(kM_0-1)} + 2(2^{d-M_0-2} - 1)2^{-q(kM_0-1)} && \text{(A.31)} \\
&= \left(1 + 2^{d-M_0-1} - 2\right) \cdot 2^{-q(kM_0-1)} && \text{(A.32)} \\
&= (2^{d-M_0-1} - 1) \cdot 2^{-q(kM_0-1)} && \text{(A.33)} \\
&= g(d). && \text{(A.34)}
\end{aligned}$$

Since our choice of the starting distribution F was arbitrary, we have that $\mathbb{P}_F(R(M', d)^c)$ is upper bounded by $g(d)$ for any F in $\mathcal{F}(M', N')$; since $Q(M', N', d)$ is defined as the supremum of this probability over all distributions F in $\mathcal{F}(M', N')$, we thus have

$$Q(M', N', d) \leq g(d), \tag{A.35}$$

as required. □

Having established Lemmas 4 and 5, we put these two results together to establish Theorem 18.

Proof of Theorem 18: For $i = 0, 1, \ldots, \bar{d}$, we define the quantity μ_i as follows:

$$\begin{aligned}
\mu_0 &= M, \\
\mu_i &= \lfloor \beta \cdot \mu_{i-1} \rfloor, \ i = 1, \ldots, \bar{d}.
\end{aligned}$$

It is straightforward to see that $\mu_i \leq M\beta^i$ for $i = 1, \ldots, \bar{d}$.

We now show that the following collection of inequalities holds:

$$
\begin{aligned}
&Q(M', N - \bar{d}q, M_0 + 1) && \leq g(M_0 + 1), \quad \forall M' \leq \mu_{\bar{d}}, && \text{(Q-0)}\\
&Q(M', N - (\bar{d} - 1)q, M_0 + 2) && \leq g(M_0 + 2), \quad \forall M' \leq \mu_{\bar{d}-1}, && \text{(Q-1)}\\
&Q(M', N - (\bar{d} - 2)q, M_0 + 3) && \leq g(M_0 + 3), \quad \forall M' \leq \mu_{\bar{d}-2}, && \text{(Q-2)}\\
&\vdots\\
&Q(M', N - (\bar{d} - i + 1)q, M_0 + i) && \leq g(M_0 + i), \quad \forall M' \leq \mu_{\bar{d}-i+1}, && \text{(Q-}(i-1)\text{)}\\
&Q(M', N - (\bar{d} - i)q, M_0 + i + 1) && \leq g(M_0 + i + 1), \quad \forall M' \leq \mu_{\bar{d}-i}, && \text{(Q-}i\text{)}\\
&\vdots\\
&Q(M', N - q, M_0 + \bar{d}) && \leq g(M_0 + \bar{d}), \quad \forall M' \leq \mu_1, && \text{(Q-}(\bar{d}-1)\text{)}\\
&Q(M', N, M_0 + \bar{d} + 1) && \leq g(M_0 + \bar{d} + 1), \quad \forall M' \leq \mu_0. && \text{(Q-}\bar{d}\text{)}
\end{aligned}
$$

The first inequality (Q-0) holds because $N - \bar{d}q \geq 0$ by the definition of q, so we can invoke Lemma 4 to guarantee that $Q(M', N - \bar{d}q, M_0 + 1) \leq g(M_0 + 1)$ holds for any $M' \leq M_0$. Note that the inequality holds for the range $M' \leq \mu_{\bar{d}}$, because $\mu_{\bar{d}} \leq M_0$ (this is a consequence of $\mu_{\bar{d}} \leq M\beta^{\bar{d}}$ and the definition of $\bar{d}$ as $\lceil \log_{2-\epsilon}(M/M_0) \rceil$).

For $i = 1, \ldots, \bar{d}$, suppose that (Q-$(i-1)$) holds. For any $M' \leq \mu_{\bar{d}-i}$, observe that the inequality of (Q-i) will hold for M' because $\lfloor \beta M' \rfloor \leq \lfloor \beta \mu_{\bar{d}-i} \rfloor = \mu_{\bar{d}-i+1}$ and because of Lemma 5. Thus, (Q-$(i-1)$) implies (Q-i).

Since (Q-0) holds and (Q-$(i-1)$) implies (Q-i) for $i = 1, \ldots, \bar{d}$, it follows by induction that (Q-$\bar{d}$) holds. Theorem 18 follows because the desired inequality (A.11) is contained in the inequalities of (Q-$\bar{d}$). □

Having proved Theorem 18, we are now in a position to prove Theorem 3.

Proof of Theorem 3: The statement of Theorem 18 allows us to use any M_0 and ϵ; let us fix $M_0 = 20$ and $\epsilon = 0.5$.

With these choices for M_0 and ϵ, we will now simplify the probability bound in (A.11).

We define the constant ξ as $\xi = \log_{2-\epsilon} 2 = \log_{1.5} 2$. We observe that

$$\begin{aligned}
\bar{d} &= \lceil \log_{2-\epsilon}(M/M_0) \rceil \\
&= \lceil \log_{1.5} M - \log_{1.5} M_0 \rceil \\
&\leq \log_{1.5} M \\
&= \xi \log_2 M, \qquad \text{(A.36)}
\end{aligned}$$

where the inequality follows because $\log_{1.5} 20 > 1$. Thus, the forest depth, which is at most $M_0 + 1 + \bar{d}$, is of order $O(\log_2 M)$. In addition, we have that

$$(2^{\bar{d}} - 1) \leq 2^{\xi \log_2 M} = M^{\xi} \leq M^2, \qquad \text{(A.37)}$$

which follows because $\xi \leq 2$. Lastly, we have

$$\begin{aligned}
2^{-q(kM_0-1)} &= 2^{-\lfloor N/\bar{d} \rfloor \cdot (0.111)} \\
&\leq 2^{-(N/\bar{d}-1)(0.111)} \\
&\leq 2^{-(N/(\xi \log_2 M)-1)(0.111)} \\
&= 2^{0.111} \cdot 2^{-(0.111/1.710)(N/\log_2 M)} \\
&= 2^{0.111} \cdot 2^{-0.065 N/\log_2 M} \qquad \text{(A.38)}
\end{aligned}$$

which is of order $O(2^{-CN/\log_2 M})$, where C is a positive constant. Putting together (A.37) and (A.38), we thus obtain a bound on $Q(M, N, \bar{d}+M_0+1)$, which is of order $O(M^2 \cdot 2^{-CN/\log_2 M})$. The statement of the theorem thus follows. Finally, the forest may contain trees that violate Requirement 3 in Definition 6. For any such tree, we apply the procedure in Lemma 1 (Algorithm 12) to obtain an equivalent tree without increasing the depth and leaf complexity of the overall forest. □

We note that the probability bound of Theorem 18 may not always be less than or equal to 1; a necessary but not sufficient condition for the bound to be less than or equal to 1 is

that ϵ and M_0 are chosen so that $kM_0 > 1$, ensuring that the coefficient of q in $2^{-q(kM_0-1)}$ is negative.

To obtain the probability bound of Section 3.2.2 with $N = 10000$ and $M = 2000$, we set $M_0 = 20$ and $\epsilon = 0.5$. We then obtain $\bar{d} = \lceil \log_{2-0.5}(2000/20) \rceil = 12$ and $Q(10000, 2000, 40) \leq (2000)^2 \cdot 2^{0.111} \cdot 2^{-0.065 \times 10000/\log_2 2000} \leq 6.2 \times 10^{-12}$. The forest depth is at most $\bar{d} + M_0 + 1 = 12 + 20 + 1 = 33$, as required. Using Theorem 18, we can actually obtain a tighter bound. To set up the bound, note that $q = \lfloor N/\bar{d} \rfloor = 833$ and $kM_0 - 1 = 1/9$, and so we obtain $Q(10000, 2000, 40) \leq (2^{\bar{d}} - 1) \cdot 2^{-q(kM_0-1)} \leq 5.63 \times 10^{-25}$. In Table A.1.4 below, we report values of both the bound in the proof of Theorem 3 as well as the tighter bound of Theorem 18 for a collection of values of M and N. Note that in the table, "Failure Prob. Bound (Theorem 3)" corresponds to the loose bound from the proof of Theorem 3 (the bound $M^2 \cdot 2^{0.111} \cdot 2^{-0.065N/\log_2 M}$). "Failure Prob. Bound (Theorem 18)" corresponds to the tighter bound of Theorem 18 (the bound $(2^{\bar{d}} - 1) \cdot 2^{-q(kM_0-1)}$, with $M_0 = 20$ and $\epsilon = 0.5$). "Depth" is the bound on the depth of the forest that corresponds to the two probability bounds, which in both cases is $M_0 + \bar{d} + 1$. .

N	M	Depth	Failure Prob. Bound (Theorem 3)	Failure Prob. Bound (Theorem. 18)
2000	100	25	0.0139	2.83×10^{-16}
2000	200	27	0.328	4.58×10^{-10}
2000	500	29	11.7	1.11×10^{-6}
2000	1000	31	128	0.000209
2000	2000	33	1.17×10^{3}	0.0115
2000	5000	35	1.77×10^{4}	0.292
2000	10000	37	1.23×10^{5}	4.32
2000	20000	39	7.88×10^{5}	50.8
5000	100	25	2.04×10^{-11}	2.32×10^{-41}
5000	200	27	6.87×10^{-9}	8.66×10^{-27}
5000	500	29	3.31×10^{-6}	3.17×10^{-19}
5000	1000	31	0.000165	1.93×10^{-14}
5000	2000	33	0.00518	4.99×10^{-11}
5000	5000	35	0.295	1.88×10^{-8}
5000	10000	37	4.69	2.4×10^{-6}
5000	20000	39	61.4	0.000142
10000	100	25	3.84×10^{-26}	3.6×10^{-83}
10000	200	27	1.09×10^{-21}	1.19×10^{-54}
10000	500	29	4.06×10^{-17}	3.95×10^{-40}
10000	1000	31	2.52×10^{-14}	3.65×10^{-31}
10000	2000	33	6.21×10^{-12}	5.63×10^{-25}
10000	5000	35	3.22×10^{-9}	2.15×10^{-20}
10000	10000	37	2.04×10^{-7}	8.16×10^{-17}
10000	20000	39	8.74×10^{-6}	7.16×10^{-14}

Table A.1: Probability bound values and corresponding depths for different values of N and M.

A.1.5 Proof of Theorem 4

Before diving into the main elements of the proof, we fix some additional notation. We use $\mathbf{A}_t = (\mathbf{A}_{t,S})_{S \in \mathcal{S}}$ to denote the vector obtained by concatenating the vectors $\mathbf{A}_{t,S}$ over all the

training assortments $S \in \mathcal{S}$. Note that each vector $\mathbf{A}_{t,S}$ is a $N+1$ dimensional "one-hot" vector (i.e., exactly one entry of $\mathbf{A}_{t,S}$ is one, and the remaining entries are zero); therefore, the vector $\mathbf{A}_t$, being the concatenation of M one-hot vectors, will have L_1 norm $\|\mathbf{A}_t\|_1 = M$ and L_2 norm $\|\mathbf{A}_t\|_2 = \sqrt{M}$. We additionally define $\mathbf{v} = (\mathbf{v}_S)_{S \in \mathcal{S}}$ to be the concatenation of the $\mathbf{v}_S$ vectors.

For a collection of trees F and a nonnegative weight vector $\boldsymbol{\mu} = (\mu_t)_{t \in F}$ corresponding to F, we define $\boldsymbol{\psi}(F, \boldsymbol{\mu})$ as

$$\boldsymbol{\psi}(F, \boldsymbol{\mu}) = \sum_{t \in F} \mathbf{A}_t \mu_t. \tag{A.39}$$

When $\boldsymbol{\mu}$ sums up to 1, $\boldsymbol{\mu}$ corresponds to a probability distribution over F, and $(F, \boldsymbol{\mu})$ is a bona fide decision forest model. In that case, recall that $\sum_{t \in F} \mathbf{A}_{t,S} \mu_t$ is the vector of predicted choice probabilities for assortment S given the decision forest model $(F, \boldsymbol{\lambda})$; thus, when $\boldsymbol{\mu}$ is a probability distribution over the forest F, then $\boldsymbol{\psi}(F, \boldsymbol{\mu})$ is the concatenation of all such vectors of predicted choice probabilities, over all of the assortments in $\mathcal{S}$. We refer to a tuple $(F, \boldsymbol{\mu})$ where $\boldsymbol{\mu}$ does not necessarily sum to one as an *extended decision forest model*. Our definition of $\boldsymbol{\psi}$ is intentionally general as we will use it in conjunction with both ordinary (non-extended) and extended decision forest models.

With these additional definitions, we now prove Theorem 4. Our proof relies on two auxiliary results (Lemma 6 and Lemma 8), which we will establish after stating the proof of Theorem 4.

Proof of Theorem 4: Recall that the randomized tree sampling method (Algorithm 3) returns a decision forest model $(\hat{F}, \hat{\boldsymbol{\lambda}})$, where $\hat{F} = \{t_1, \ldots, t_K\}$ is a random sample of K trees drawn i.i.d. from F according to the distribution $\boldsymbol{\xi}$, and $\hat{\boldsymbol{\lambda}}$ is obtained by solving the problem $\textsc{EstLO}(\mathcal{S}, \hat{F})$. Let $\boldsymbol{\lambda}^*$ be a probability distribution over $\Lambda(C, \boldsymbol{\xi})$ that minimizes the empirical risk, that is,

$$\boldsymbol{\lambda}^* \in \arg \min_{\boldsymbol{\lambda} \in \Lambda(C, \boldsymbol{\xi})} R(F, \boldsymbol{\lambda}). \tag{A.40}$$

By Lemma 8, with probability $1-\delta$ over the collection of trees $t_1, \ldots, t_K$, there exists a forest model $(\hat{F}, \boldsymbol{\lambda}')$ such that

$$\|\boldsymbol{\psi}(\hat{F}, \boldsymbol{\lambda}') - \boldsymbol{\psi}(F, \boldsymbol{\lambda}^*)\|_1 \leq \frac{MC}{\sqrt{K}} \cdot \left(\sqrt{N+1} + 3\sqrt{\log(4/\delta)}\right). \tag{A.41}$$

In words, Lemma 8 allows us to assert that with high probability there exists a distribution $\boldsymbol{\lambda}'$ over $\hat{F}$ such that the predicted choice probabilities under $(\hat{F}, \boldsymbol{\lambda}')$ are close to those under $(F, \boldsymbol{\lambda}^*)$.

Thus, with probability at least $1-\delta$, we have the following bound:

$$\begin{aligned}
\mathbf{R}(\hat{F}, \hat{\boldsymbol{\lambda}}) - \min_{\boldsymbol{\lambda} \in \Lambda(C, \boldsymbol{\xi})} \mathbf{R}(F, \boldsymbol{\lambda}) &= \mathbf{R}(\hat{F}, \hat{\boldsymbol{\lambda}}) - \mathbf{R}(F, \boldsymbol{\lambda}^*) \\
&\leq \mathbf{R}(\hat{F}, \boldsymbol{\lambda}') - \mathbf{R}(F, \boldsymbol{\lambda}^*) \\
&\leq \frac{1}{M} \cdot \|\boldsymbol{\psi}(\hat{F}, \boldsymbol{\lambda}') - \boldsymbol{\psi}(F, \boldsymbol{\lambda}^*)\|_1 \\
&\leq \frac{1}{M} \cdot \frac{MC}{\sqrt{K}} \cdot \left(\sqrt{N+1} + 3\sqrt{\log(4/\delta)}\right) \\
&= \frac{C}{\sqrt{K}} \cdot \left(\sqrt{N+1} + 3\sqrt{\log(4/\delta)}\right),
\end{aligned}$$

where the first equality follows by the definition of $\boldsymbol{\lambda}^*$; the first inequality follows since $\hat{\boldsymbol{\lambda}}$ minimizes $\mathbf{R}(\hat{F}, \boldsymbol{\lambda})$ over all probability distributions $\boldsymbol{\lambda}$; the second inequality follows by Lemma 6; and the final inequality by Lemma 8. Re-arranging the inequality to place $\min_{\boldsymbol{\lambda} \in \Lambda(C, \boldsymbol{\xi})} \mathbf{R}(F, \boldsymbol{\lambda})$ to the right-hand side, we obtain the desired result. □

We now establish the auxiliary results used in the proof of Theorem 4. Our first auxiliary result, Lemma 6, states that the difference in training error/empirical risk between two decision forests $(F, \boldsymbol{\lambda})$ and $(F', \boldsymbol{\lambda}')$ can be bounded simply by the L_1 distance between their predicted choice probabilities.

Lemma 6 *For any two forest models $(F, \boldsymbol{\lambda})$ and $(F', \boldsymbol{\lambda}')$,*

$$\mathbf{R}(F, \boldsymbol{\lambda}) - \mathbf{R}(F', \boldsymbol{\lambda}') \leq \frac{\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\lambda}')\|_1}{M}. \tag{A.42}$$

Proof of Lemma 6: By the triangle inequality, we have

$$\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \mathbf{v}\|_1 \leq \|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\lambda}')\|_1 + \|\boldsymbol{\psi}(F', \boldsymbol{\lambda}') - \mathbf{v}\|_1,$$

which we can re-arrange to obtain

$$\mathbf{R}(F, \boldsymbol{\lambda}) - \mathbf{R}(F', \boldsymbol{\lambda}') = \frac{\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \mathbf{v}\|_1}{M} - \frac{\|\boldsymbol{\psi}(F', \boldsymbol{\lambda}') - \mathbf{v}\|_1}{M} \leq \frac{\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\lambda}')\|_1}{M},$$

as required. □

We next turn our attention to Lemma 8. Before we can establish Lemma 8, it is helpful to establish the following general-purpose lemma. For a collection of i.i.d. random vectors, Lemma 7 provides a high probability bound on the L_1 norm of the distance between the average vector and the expectation of the average vector.

Lemma 7 *Let $\mathbf{z}_1, \ldots, \mathbf{z}_K$ be i.i.d. random vectors of size $(N+1)M$ such that $\mathbf{z}_k \geq \mathbf{0}$, $\|\mathbf{z}_k\|_1 \leq A$ and $\|\mathbf{z}_k\|_2 \leq B$ for $k = 1, \ldots, K$, for some positive constants A and B. Let $\bar{\mathbf{z}} = (1/K)\sum_{k=1}^{K} \mathbf{z}_k$ denote their average. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of $\mathbf{z}_1, \ldots, \mathbf{z}_K$, we have that*

$$\|\bar{\mathbf{z}} - \mathbb{E}[\bar{\mathbf{z}}]\|_1 \leq \sqrt{\frac{(N+1)MB^2}{K}} + \sqrt{\frac{2A^2}{K}\log\left(\frac{1}{\delta}\right)} \tag{A.43}$$

Proof of Lemma 7: First, let us define the set $\mathcal{Y}$ from which $\mathbf{z}_1, \ldots, \mathbf{z}_K$ are drawn as

$$\mathcal{Y} = \left\{\mathbf{y} \in \mathbb{R}^{(N+1)M} \mid \mathbf{y} \geq \mathbf{0}, \|\mathbf{y}\|_1 \leq A, \|\mathbf{y}\|_2 \leq B\right\}.$$

Let us also define the scalar function $f : \mathcal{Y}^K \to \mathbb{R}$ as

$$f(\mathbf{y}_1, \ldots, \mathbf{y}_K) = \|\frac{1}{K}\sum_{k=1}^{K} \mathbf{y}_k - \mathbb{E}[\bar{\mathbf{z}}]\|_1.$$

Observe that the random variable $f(\mathbf{z}_1, \ldots, \mathbf{z}_K)$ is equivalent to the random variable on the left-hand side of inequality (A.43); our goal will be to show that $f(\mathbf{z}_1, \ldots, \mathbf{z}_K)$ satisfies this

bound. We will show this by combining a bound on how much $f(\mathbf{z}_1, \ldots, \mathbf{z}_K)$ deviates from its expected value (which we will obtain using McDiarmid's inequality) and a bound on the expected value of $f(\mathbf{z}_1, \ldots, \mathbf{z}_K)$.

To eventually use McDiarmid's inequality, we first show that f possesses the bounded differences property. For any $\mathbf{y}_1, \ldots, \mathbf{y}_K \in \mathcal{Y}$, let $\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_K$ be a collection of vectors in $\mathcal{Y}$ such that $\mathbf{y}_k = \tilde{\mathbf{y}}_k$ for all $k \neq m$, where the index $m \in \{1, \ldots, K\}$ is arbitrary. We then have:

$$\begin{aligned}
|f(\mathbf{y}_1, \ldots, \mathbf{y}_K) - f(\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_K)| &= \left| \|\frac{1}{K}\sum_{k=1}^{K} \mathbf{y}_k - \mathbb{E}[\bar{\mathbf{z}}]\|_1 - \|\frac{1}{K}\sum_{k=1}^{K} \tilde{\mathbf{y}}_k - \mathbb{E}[\bar{\mathbf{z}}]\|_1 \right| \\
&\leq \|\frac{1}{K}\sum_{k=1}^{K} \mathbf{y}_k - \frac{1}{K}\sum_{k=1}^{K} \tilde{\mathbf{y}}_k\|_1 \\
&= \frac{1}{K}\|\mathbf{y}_m - \tilde{\mathbf{y}}_m\|_1 \\
&\leq \frac{2A}{K},
\end{aligned}$$

where both inequalities follow by an application of the triangle inequality.

We next bound $\mathbb{E}[f(\mathbf{z}_1, \ldots, \mathbf{z}_K)]$. To do so, we first derive an auxiliary bound on $\mathbb{E}[\|\bar{\mathbf{z}} - \mathbb{E}[\bar{\mathbf{z}}]\|_2^2]$ by

$$\mathbb{E}[\|\bar{\mathbf{z}} - \mathbb{E}[\bar{\mathbf{z}}]\|_2^2] = \frac{\mathbb{E}[\|\bar{\mathbf{z}}\|_2^2] - \|\mathbb{E}[\bar{\mathbf{z}}]\|_2^2}{K} \leq \frac{\mathbb{E}[\|\bar{\mathbf{z}}\|_2^2]}{K} \leq \frac{B^2}{K}, \tag{A.44}$$

where the first inequality follows since $\|\mathbb{E}[\bar{\mathbf{z}}]\|_2^2 \geq 0$ and the second follows because $\mathcal{Y}$ is convex, so that $\|\bar{\mathbf{z}}\|_2$ is bounded by B almost surely.

Using this bound, we now derive a bound on $\mathbb{E}[f(\mathbf{z}_1, \ldots, \mathbf{z}_K)]$:

$$\begin{aligned}
\mathbb{E}[f(\mathbf{z}_1, \ldots, \mathbf{z}_K)] &= \mathbb{E}[\|\bar{\mathbf{z}} - \mathbb{E}[\bar{\mathbf{z}}]\|_1] \\
&\leq \sqrt{(N+1)M}\mathbb{E}[\|\bar{\mathbf{z}} - \mathbb{E}[\bar{\mathbf{z}}]\|_2] \\
&\leq \sqrt{(N+1)M}\sqrt{\mathbb{E}[\|\bar{\mathbf{z}} - \mathbb{E}[\bar{\mathbf{z}}]\|_2^2]} \\
&\leq \sqrt{\frac{(N+1)MB^2}{K}},
\end{aligned} \tag{A.45}$$

where the first inequality follows by the basic properties of L_1 and L_2 norms, the second inequality follows by using Jensen's inequality and the concavity of the function $h(x) = \sqrt{x}$, and the final inequality by using inequality (A.44).

We now have all the pieces necessary to establish the bound (A.43). For any $\epsilon > 0$, we have:

$$\begin{aligned}\mathbb{P}\left(f(\mathbf{z}_1,\ldots,\mathbf{z}_K) - \sqrt{\frac{(N+1)MB^2}{K}} \geq \epsilon\right) &\leq \mathbb{P}\left(f(\mathbf{z}_1,\ldots,\mathbf{z}_K) - \mathbb{E}[f(\mathbf{z}_1,\ldots,\mathbf{z}_K)] \geq \epsilon\right) \\ &\leq \exp\left(\frac{-2\epsilon^2}{K \cdot \frac{4A^2}{K^2}}\right) \\ &= \exp\left(-\frac{K\epsilon^2}{2A^2}\right),\end{aligned}$$

where the first inequality follows by our bound on the expected value (inequality (A.45)) and the second inequality follows by an application of McDiarmid's inequality. Letting $\epsilon = \sqrt{(2A^2/K)\log(1/\delta)}$, we obtain that

$$f(\mathbf{z}_1,\ldots,\mathbf{z}_K) \leq \sqrt{\frac{(N+1)MB^2}{K}} + \sqrt{\frac{2A^2}{K}\log\left(\frac{1}{\delta}\right)}$$

with probability at least $1-\delta$, which completes the proof. □

Equipped with Lemma 7, we can now turn to proving Lemma 8, which is at the heart of the proof of Theorem 4. Lemma 8 states that for any decision forest model $(F, \boldsymbol{\lambda})$ where $\boldsymbol{\lambda} \in \Lambda(C, \boldsymbol{\xi})$, we can find a new distribution $\boldsymbol{\lambda}'$ over the forest F', which is an i.i.d. sample of K trees from $\boldsymbol{\xi}$, such that the L_1 distance between the predicted choice probabilities of $(F, \boldsymbol{\lambda})$ and $(F', \boldsymbol{\lambda}')$ is bounded with high probability. The distribution $\boldsymbol{\lambda}'$ is constructed by first constructing an appropriate extended forest model $(F', \boldsymbol{\mu}')$, and then normalizing $\boldsymbol{\mu}'$ to sum to 1. The proof then involves two steps: (1) showing that $(F, \boldsymbol{\lambda})$ and $(F', \boldsymbol{\mu}')$ are close in their choice probabilities using Lemma 7; and (2) showing that $(F', \boldsymbol{\mu}')$ and $(F', \boldsymbol{\lambda}')$ are also close in their choice probabilities (using the fact that the normalization constant of $\boldsymbol{\lambda}'$ concentrates to 1). We note that our proof of step (1) resembles a technique used in

the machine learning literature on building classifiers as weighted sums of random feature functions [100]. Specifically, we use a similar procedure as in [100] to construct our extended forest model $(F', \boldsymbol{\mu}')$. However, unlike the weighted sum models in [100], a decision forest model requires that the weight vector $\boldsymbol{\lambda}$ sum up to 1 as it corresponds to a probability distribution; for this reason, we must also establish step (2).

Lemma 8 *Consider a decision forest model $(F, \boldsymbol{\lambda})$ where $\boldsymbol{\lambda} \in \Lambda(C, \boldsymbol{\xi})$. Suppose that $\xi_t > 0$ for all $t \in F$, and that $t_1, \ldots, t_K$ are drawn i.i.d. from F according to the distribution $\boldsymbol{\xi}$. For any $\delta > 0$, with probability at least $1 - \delta$ over the draw of $t_1, \ldots, t_K$, there exists a decision forest model $(F', \boldsymbol{\lambda}')$ such that $F' = \{t_1, \ldots, t_K\}$ and*

$$\|\boldsymbol{\psi}(F', \boldsymbol{\lambda}') - \boldsymbol{\psi}(F, \boldsymbol{\lambda})\|_1 \leq \frac{MC}{\sqrt{K}} \left(\sqrt{N+1} + 3\sqrt{\log\left(\frac{4}{\delta}\right)} \right). \tag{A.46}$$

Proof of Lemma 8: Let $F' = \{t_1, \ldots, t_K\}$. We first consider the extended forest model $(F', \boldsymbol{\mu}')$, where $\boldsymbol{\mu}'$ is defined as

$$\mu'_{t_k} = \frac{1}{K} \left(\frac{\lambda_{t_k}}{\xi_{t_k}} \right).$$

Note that $\boldsymbol{\mu}'$ is not necessarily a probability distribution because it need not add up to 1. We thus define the distribution $\boldsymbol{\lambda}'$ over F' by normalizing $\boldsymbol{\mu}'$:

$$\boldsymbol{\lambda}' = \left(\frac{1}{\sum_{k=1}^{K} \mu'_{t_k}} \right) \boldsymbol{\mu}' \tag{A.47}$$

We claim that $(F', \boldsymbol{\lambda}')$ satisfies the statement of the theorem. To see how, observe that we can bound the quantity $\|\boldsymbol{\psi}(F', \boldsymbol{\lambda}') - \boldsymbol{\psi}(F, \boldsymbol{\lambda})\|_1$ as

$$\|\boldsymbol{\psi}(F', \boldsymbol{\lambda}') - \boldsymbol{\psi}(F, \boldsymbol{\lambda})\|_1 \leq \underbrace{\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\mu}')\|_1}_{\text{Term (a)}} + \underbrace{\|\boldsymbol{\psi}(F', \boldsymbol{\mu}') - \boldsymbol{\psi}(F', \boldsymbol{\lambda}')\|_1}_{\text{Term (b)}}, \tag{A.48}$$

which follows by applying the triangle inequality. We now show that each of the two terms on the right hand side can be bounded with high probability.

Bounding term (a): To bound $\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\mu}')\|_1$, let us define the random vector $\mathbf{z}_k$ as

$$\mathbf{z}_k = \left(\frac{\lambda_{t_k}}{\xi_{t_k}}\right) \mathbf{A}_{t_k}, \tag{A.49}$$

where we recall that $\mathbf{A}_t$ is the concatenation of M one-hot vectors of size $N+1$ corresponding to the choices that tree t makes on the M assortments in the data. Let us also define the random vector $\bar{\mathbf{z}} = (1/K)\sum_{k=1}^{K} \mathbf{z}_k$ as the average of the K random vectors.

The vectors $\mathbf{z}_1, \ldots, \mathbf{z}_K$ have a couple of desirable properties. First, observe that

$$\boldsymbol{\psi}(F', \boldsymbol{\mu}') = \sum_{k=1}^{K} \mu'_{t_k} \mathbf{A}_{t_k} = \sum_{k=1}^{K} \left(\frac{1}{K}\right)\left(\frac{\lambda_{t_k}}{\xi_{t_k}}\right) \mathbf{A}_{t_k} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{z}_k,$$

i.e., the (random) vector of choice probabilities of the extended forest model $(F', \boldsymbol{\mu}')$ is equal to the average of the random vectors $\mathbf{z}_1, \ldots, \mathbf{z}_K$.

Second, observe that for any $k \in \{1, \ldots, K\}$, we have

$$\mathbb{E}[\bar{\mathbf{z}}] = \mathbb{E}[\mathbf{z}_k] = \sum_{t \in F} \xi_t \cdot \frac{\lambda_t}{\xi_t} \mathbf{A}_t = \sum_{t \in F} \lambda_t \mathbf{A}_t = \boldsymbol{\psi}(F, \boldsymbol{\lambda}),$$

i.e., the expected value of $\bar{\mathbf{z}}$ is exactly equal to the vector of choice probabilities of the decision forest model $(F, \boldsymbol{\lambda})$.

We can thus re-write the term $\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\mu}')\|_1$ as

$$\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\mu}')\|_1 = \|\bar{\mathbf{z}} - \mathbb{E}[\bar{\mathbf{z}}]\|_1, \tag{A.50}$$

which is exactly in the form of the bound in Lemma 7. In order to apply the bound, we only need to obtain bounds on the L_1 and L_2 norms of the random vectors $\mathbf{z}_1, \ldots, \mathbf{z}_K$. Since $\mathbf{z}_k = (\lambda_{t_k}/\xi_{t_k})\mathbf{A}_{t_k}$, we can use the fact that $\lambda_t \leq C\xi_t$ for every $t \in F$ (recall that $\boldsymbol{\lambda} \in \Lambda(C, \boldsymbol{\xi})$) and the fact that $\mathbf{A}_t$ consists of M one-hot vectors concatenated together, to obtain the following bounds:

$$\|\mathbf{z}_k\|_1 = \frac{\lambda_{t_k}}{\xi_{t_k}} \cdot \|\mathbf{A}_{t_k}\|_1 \leq C\|\mathbf{A}_{t_k}\|_1 = CM,$$

$$\|\mathbf{z}_k\|_2 = \frac{\lambda_{t_k}}{\xi_{t_k}} \cdot \|\mathbf{A}_{t_k}\|_2 \leq C\|\mathbf{A}_{t_k}\|_2 = C\sqrt{M}.$$

With these bounds in hand, we can invoke Lemma 7 with $\delta/2$ to obtain that

$$\begin{aligned}\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\mu}')\|_1 &= \|\bar{\mathbf{z}} - \mathbb{E}[\bar{\mathbf{z}}]\|_1 \\ &\leq \sqrt{\frac{(N+1)M \cdot C^2 M}{K}} + \sqrt{\frac{2C^2M^2}{K}\log\left(\frac{1}{\delta/2}\right)} \\ &= \frac{CM}{\sqrt{K}}\left(\sqrt{N+1} + \sqrt{2\log\left(\frac{2}{\delta}\right)}\right), \end{aligned} \tag{A.51}$$

with probability at least $1 - \delta/2$.

Bounding term (b): To bound $\|\boldsymbol{\psi}(F', \boldsymbol{\mu}') - \boldsymbol{\psi}(F', \boldsymbol{\lambda}')\|_1$, let us define the random variable s as $s = \sum_{k=1}^{K} \mu'_{t_k}$, which is simply the normalization constant used to define $\boldsymbol{\lambda}'$. Let us also define the random variables $\beta_1, \ldots, \beta_K$ as $\beta_k = \lambda_{t_k}/\xi_{t_k}$. Observe that s can then be written as

$$s = \frac{1}{K}\sum_{k=1}^{K} \beta_k, \tag{A.52}$$

i.e., s is the average of K i.i.d. random variables, $\beta_1, \ldots, \beta_K$. Moreover, each random variable β_k is bounded between 0 (since λ_t is nonnegative and ξ_t is positive for every $t \in F$) and C (since $\lambda_t \leq C\xi_t$, by definition of $\Lambda(C, \boldsymbol{\xi})$). Lastly, observe that for each k,

$$\mathbb{E}[\beta_k] = \mathbb{E}[\lambda_{t_k}/\xi_{t_k}] = \sum_{t \in F} \xi_t \cdot \frac{\lambda_t}{\xi_t} = \sum_{t \in F} \lambda_t = 1,$$

i.e., the expected value of each β_k is 1, and thus, $\mathbb{E}[s]$ will also be 1. We can therefore apply Hoeffding's inequality to bound the deviation of s from 1. For any $\epsilon > 0$, we have

$$\mathbb{P}\left(|s - 1| \geq \epsilon\right) \leq 2\exp\left(-\frac{2K\epsilon^2}{C^2}\right).$$

Set $\epsilon = C\sqrt{\log(4/\delta)/2K}$. We then have, with probability at least $1 - \delta/2$, that

$$|s - 1| \leq C\sqrt{\frac{1}{2K}\log\left(\frac{4}{\delta}\right)}.$$

We can now use this to bound term (b); we have

$$\begin{aligned}
\|\boldsymbol{\psi}(F', \boldsymbol{\lambda}') - \boldsymbol{\psi}(F', \boldsymbol{\mu}')\|_1 &= \|\sum_{k=1}^{K} \lambda'_{t_k} \mathbf{A}_{t_k} - \sum_{k=1}^{K} \mu'_{t_k} \mathbf{A}_{t_k}\|_1 \\
&= \|\sum_{k=1}^{K} \lambda'_{t_k} \mathbf{A}_{t_k} - s \sum_{k=1}^{K} \lambda'_{t_k} \mathbf{A}_{t_k}\|_1 \\
&= |s-1| \cdot \|\sum_{k=1}^{K} \lambda'_{t_k} \mathbf{A}_{t_k}\|_1 \\
&= |s-1| \cdot M \\
&\leq M \cdot C\sqrt{\frac{1}{2K}\log\left(\frac{4}{\delta}\right)}, \qquad \text{(A.53)}
\end{aligned}$$

with probability at least $1 - \delta/2$. (In the above, note that the second step follows by the definition of $\boldsymbol{\lambda}'$ in (A.47).)

Bounding $\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\lambda}')\|_1$: We now complete the proof. Using the bounds (A.51) and (A.53) together with the inequality (A.48) and the union bound, we get:

$$\begin{aligned}
\|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\lambda}')\|_1 &\leq \|\boldsymbol{\psi}(F, \boldsymbol{\lambda}) - \boldsymbol{\psi}(F', \boldsymbol{\mu}')\|_1 + \|\boldsymbol{\psi}(F', \boldsymbol{\lambda}') - \boldsymbol{\psi}(F', \boldsymbol{\mu}')\|_1 \\
&\leq \frac{CM}{\sqrt{K}}\left(\sqrt{N+1} + \sqrt{2\log\left(\frac{2}{\delta}\right)}\right) + \frac{CM}{\sqrt{K}}\sqrt{\frac{1}{2}\log\left(\frac{4}{\delta}\right)} \\
&\leq \frac{CM}{\sqrt{K}}\left(\sqrt{N+1} + 3\sqrt{\log\left(\frac{4}{\delta}\right)}\right),
\end{aligned}$$

with probability at least $1 - \delta$, as required. □

A.2 Additional Results to the Column Generation Method

A.2.1 An Exact Formulation of the Column Generation (CG) Subproblem

We follow the notation in Section 3.3. The subproblem for the column generation approach is to find a decision tree t and the corresponding 0-1 vectors $\mathbf{A}_{t,S}$ for $S \in \mathcal{S}$ such that the

reduced cost of the corresponding λ_t variable, given by $-\sum_{S\in\mathcal{S}} \boldsymbol{\alpha}_S^T \mathbf{A}_{t,S} - \nu$, is minimized. To formulate the subproblem, we introduce a binary decision variable y_o^ℓ for each leaf node $\ell \in \mathbf{leaves}(t)$ and option $o \in \mathcal{N}_+$ that is 1 if the purchase decision of leaf ℓ is option o, and 0 otherwise. Similarly, for each split node $s \in \mathbf{splits}(t)$ and product $p \in \mathcal{N}$, we define the binary decision variable y_p^s to be 1 if product p participates in split node s, and 0 otherwise. For each $\ell \in \mathbf{leaves}(t)$, we define the binary decision variable w_S^ℓ which is 1 if assortment S is mapped to leaf node ℓ under the current purchase decision tree. For each leaf $\ell \in \mathbf{leaves}(t)$, option $o \in \mathcal{N}_+$ and assortment S, we define the binary decision variable $u_{o,S}^\ell$ to be 1 if assortment S is mapped to leaf node ℓ and option o is the resulting purchase decision. Finally, we define the binary decision variable $A_{o,S}$ to indicate whether the tree chooses option o when given assortment S, for each historical assortment $S \in \mathcal{S}$; the vector of these decision variables for a given S is denoted by $\mathbf{A}_S$. With these definitions, we can

formulate the subproblem as the following mixed-integer optimization problem:

$$\underset{\mathbf{A},\mathbf{u},\mathbf{w},\mathbf{y}}{\text{minimize}} \quad -\sum_{S\in\mathcal{S}} \boldsymbol{\alpha}_S^T \mathbf{A}_S - \nu \tag{A.54a}$$

$$\text{subject to} \quad \sum_{p\in\mathcal{N}} y_p^s = 1, \quad \forall\ s \in \mathbf{splits}(t), \tag{A.54b}$$

$$\sum_{o\in\mathcal{N}_+} y_o^\ell = 1, \quad \forall\ \ell \in \mathbf{leaves}(t), \tag{A.54c}$$

$$\sum_{\ell\in\mathbf{leaves}(t)} w_S^\ell = 1, \quad \forall\ m \in \{1,\ldots,M\}, \tag{A.54d}$$

$$\sum_{\ell\in\mathbf{LL}(s)} w_S^\ell \le \sum_{p\in S} y_p^s, \quad \forall\ s \in \mathbf{splits}(t),\ S \in \mathcal{S}, \tag{A.54e}$$

$$\sum_{\ell\in\mathbf{RL}(s)} w_S^\ell \le 1 - \sum_{p\in S} y_p^s, \quad \forall\ s \in \mathbf{splits}(t),\ S \in \mathcal{S}, \tag{A.54f}$$

$$w_S^\ell = \sum_{o\in\mathcal{N}_+} u_{o,S}^\ell, \quad \forall\ \ell \in \mathbf{leaves}(t),\ S \in \mathcal{S}, \tag{A.54g}$$

$$u_{o,S}^\ell \le y_o^\ell, \quad \forall\ \ell \in \mathbf{leaves}(t),\ o \in \mathcal{N}_+,\ S \in \mathcal{S}, \tag{A.54h}$$

$$y_o^\ell \le \sum_{s\in\mathbf{LS}(\ell)} y_o^s, \quad \forall\ \ell \in \mathbf{leaves}(t),\ o \in \mathcal{N}, \tag{A.54i}$$

$$A_{o,S} = \sum_{\ell\in\mathbf{leaves}(t)} u_{o,S}^\ell, \quad \forall\ o \in \mathcal{N}_+,\ S \in \mathcal{S}, \tag{A.54j}$$

$$w_S^\ell \in \{0,1\}, \quad \forall\ \ell \in \mathbf{leaves}(t),\ S \in \mathcal{S}, \tag{A.54k}$$

$$A_{o,S} \in \{0,1\}, \quad \forall\ o \in \mathcal{N}_+,\ S \in \mathcal{S}, \tag{A.54l}$$

$$y_o^\ell \in \{0,1\}, \quad \forall\ \ell \in \mathbf{leaves}(t),\ o \in \mathcal{N}_+, \tag{A.54m}$$

$$y_p^s \in \{0,1\}, \quad \forall\ s \in \mathbf{splits}(t),\ p \in \mathcal{N}, \tag{A.54n}$$

$$u_{o,S}^\ell \in \{0,1\}, \quad \forall\ \ell \in \mathbf{leaves}(t),\ o \in \mathcal{N}_+,\ S \in \mathcal{S}. \tag{A.54o}$$

In order of appearance, the constraints have the following meaning. Constraint (A.54b) requires that exactly one product is chosen for each split in the tree. Constraint (A.54c) similarly requires exactly one option to be selected to serve as the purchase decision for each leaf. Constraint (A.54d) ensures that each assortment is mapped to exactly one leaf.

Constraints (A.54e) and (A.54f) model how the tree maps each of the assortments to a leaf. To understand these constraints, observe that the expression $\sum_{p \in S} y_p^s$ is 1 if any of the products in S is chosen for split s, in other words, it is 1 if the purchase decision process proceeds to the left child of split s, and 0 if it proceeds to the right child. If the expression evaluates to 1, the decision process proceed to the left, and constraint (A.54f) forces all the w_S^ℓ variables of all leaves ℓ that are to the right of split s to zero. Similarly, if the expression evaluates to 0, the decision process proceeds to the right, and constraint (A.54e) forcess all w_S^ℓ variables for leaves to the left of split s to zero. Constraints (A.54g) and (A.54h) ensure that $u_{o,S}^\ell$ is consistent with w_S^ℓ and y_o^ℓ. Constraint (A.54i) ensures that the tree confirms to Requirement 2 in Section 3.1.1, i.e., a product o may only be set as the purchase decision of a leaf ℓ if it participates in at least one split s for which ℓ is to the left of. Constraint (A.54j) ensures that each value $A_{o,S}$ of the tree is properly defined given $u_{o,S}^\ell$. Lastly, constraints (A.54k) to (A.54o) ensure that all of the decision variables are binary. In case that the resulting tree violates Requirement 3 in Definition 6 (Section 3.1.1), one applies the procedure in Lemma 1 (Algorithm 12) to satisfy it.

A.2.2 Numerical Comparison on Solving the CG Subproblems

We now provide a simple numerical comparison of the heuristic column generation approach (Algorithm 1 using Algorithm 2 to solve the subproblem) and the randomized tree sampling approach (Algorithm 3) with the exact column generation approach (Algorithm 1 where the subproblem is solved via the MIO formulation (A.54)). In this experiment, we consider $N \in \{4, 6, 8\}$. For each value of N, we consider an MNL model where the product utilities u_i are defined as $u_i = 0.2i$ for $i \in \{1, \ldots, N\}$. For $N = 4$, we randomly generate $M = 10$ distinct assortments; for $N = 6$, we randomly generate $M = 10, 20, 50$ distinct assortments; and for $N = 8$, we randomly generate $M = 10, 20, 50, 100, 200$ distinct assortments. For each assortment S, we compute the exact choice probability vector $\mathbf{v}_S$ and consider the L_1 estimation problem from Chapter 3.3.1. We consider values of the forest depth d in the set

$\{3, 4, 5\}$.

For each combination of N, M and d, we execute both the heuristic column generation approach and the exact column generation approach, with the average L_1 error as the objective. For the exact column generation approach, we impose a time limit of 6 hours on the whole procedure and a time limit of 30 minutes on each solve of the subproblem; the latter time limit was necessary as in some larger cases, solving just a single instance of the MIO problem (A.54) can exhaust the 6 hour time limit on the overall procedure. For the heuristic column generation approach, we use d as the depth limit for the top-down induction procedure (Algorithm 2). For the randomized tree sampling approach, we sample $K = 5000$ trees randomly from the uniform distribution over all balanced trees of depth d. For all three approaches, we do not use any warm-starting, and initialize each with an empty collection of trees.

Table A.2 compares the average L_1 training error and the overall runtime for the heuristic column generation, the exact column generation and randomized tree sampling approaches; additionally, the table also reports the number of iterations for the two column generation approaches. In terms of runtime, we can see that while the exact column generation approach is manageable for smaller instances, it quickly becomes unmanageable when N, M or d become large; for example, even with $N = 6$, $M = 50$ and $d = 5$, the exact approach does not terminate within the 6 hour time limit. In contrast, the heuristic column generation approach requires no more than 3 minutes to run even in the largest case, while the randomized tree sampling approach requires no more than 6 seconds.

With regard to the training error, we can see that in some small cases where $d = 3$ and the number of assortments M is small, the exact approach does deliver better performance (for example, $(N, M, d) = (4, 10, 3)$). However, in cases involving more products and assortments and/or deeper trees, the heuristic column generation approach is very close to the exact approach (see for example $(N, M, d) = (6, 50, 3)$). In those cases where the exact approach exhausts its time limit, the final forest produced by the exact approach typically achieves a

N	M	d	Training error (10^{-2})			Runtime (s)			Iterations	
			ECG	HCG	RTS	ECG	HCG	RTS	ECG	HCG
4	10	3	5.289	15.577	5.289	8.7	2.2	0.2	16	11
4	10	4	0.000	0.000	0.000	12.7	2.2	0.3	21	22
4	10	5	0.000	0.000	0.000	12.2	2.2	0.6	21	22
6	10	3	2.109	6.489	2.109	21.5	2.2	0.3	37	24
6	10	4	0.000	0.266	0.000	24.8	2.2	0.4	32	29
6	10	5	0.000	0.000	0.000	30.4	2.3	0.7	33	33
6	20	3	13.814	14.111	13.814	38.0	2.2	0.4	31	27
6	20	4	0.000	0.000	0.000	224.2	2.5	0.5	63	84
6	20	5	0.000	0.000	0.000	529.7	2.6	0.8	63	63
6	50	3	20.395	20.690	20.395	125.5	2.3	0.5	37	39
6	50	4	4.717	4.826	4.717	12816.8	2.8	0.9	122	131
6	50	5	1.793	0.355	1.029	21948.6	4.4	1.2	41	209
8	10	3	5.018	11.574	5.634	49.2	2.3	0.2	56	35
8	10	4	0.000	0.000	0.000	69.2	2.4	1.2	46	50
8	10	5	0.000	0.000	0.000	29.7	2.5	0.8	46	45
8	20	3	11.089	13.521	11.089	101.1	2.3	0.4	53	46
8	20	4	0.000	0.000	0.340	1063.1	2.8	1.4	100	109
8	20	5	0.000	0.000	0.000	4476.3	3.1	2.2	97	95
8	50	3	21.680	22.826	21.680	367.1	2.3	0.5	64	41
8	50	4	5.211	5.659	8.105	21648.4	4.2	1.9	163	252
8	50	5	5.397	0.000	8.472	22302.4	7.5	1.3	21	249
8	100	3	29.439	30.161	29.439	973.5	2.4	1.0	58	44
8	100	4	9.799	10.538	12.163	21871.8	3.8	4.7	124	164
8	100	5	12.115	1.647	14.962	22250.8	109.9	3.1	16	1031

Table A.2: Comparison of exact and heuristic column generation approaches.

significantly higher training error than the heuristic column generation approach. Comparing the heuristic column generation and the randomized tree sampling approach, we can see that when d is 3 or 4, the randomized tree sampling approach tends to achieve a lower training error, while the heuristic column generation method tends to do better when $d = 5$. Overall, these results suggest that exact approaches to solving the estimation problem (3.5) are difficult to deploy in practice, and that it is necessary to consider heuristic approaches.

A.2.3 Leaf-Based Heuristic CG Method

As mentioned in Section 3.3.1, one can consider an alternate form of the top-down induction heuristic in which the complexity control is formulated in terms of the number of leaves. In this version of the top-down heuristic, the main termination criterion (beside the reduced cost being locally optimal) is whether or not we reach a user-defined limit L on the number of leaves. We formally define this version of the heuristic as Algorithm 13.

We evaluate the performance of our heuristic column generation approach using the leaf-based top-down induction method (Algorithm 13 described in Section A.2.3), as opposed to the depth-based top-down induction method (Algorithm 2 described in Section 3.3.1). We run the heuristic column generation procedure with the leaf-based top-down induction method, with values for the leaf limit L in $\{4, 8, 16, 32, 64\}$. We additionally run the heuristic column generation procedure with the depth-based top-down induction method, with values for the depth limit d in $\{3, 4, 5, 6, 7\}$. Note that the values for L are chosen to match the maximum number of leaves for each depth limit d; for example, when $d = 4$, the maximum number of leaves that a purchase decision tree may have is 8. We warm start both the depth-based and the leaf-based procedures with the ranking-based model found using the method of [119].

Table A.3 shows the KL divergence (in units of 10^{-2}) of each decision forest model for each product category, averaged over the five folds of each product category. Each column labeled with $d = ...$ corresponds to the depth-based heuristic column generation method, while

Algorithm 13 Leaf-based top-down induction method for heuristically solving column generation subproblem (3.6).

procedure TopDownInduction-Leaf($\boldsymbol{\alpha}$, ν, L)
  Initialize $t \leftarrow t_0$
  Initialize $Z_c \leftarrow \left[-\sum_{S \in \mathcal{S}} \boldsymbol{\alpha}_S^T \mathbf{A}_{t,S} - \nu\right]$
  while $|\mathbf{leaves}(t)| < L$ **do**
    Compute Z_{t,ℓ,p,o_1,o_2} for all $\ell \in \mathbf{leaves}(t)$, $p \in \mathcal{N} \setminus P(\ell)$,
     $o_1 \in \{p, 0\} \cup \{x_s \mid s \in \mathbf{LS}(\ell)\}$, $o_2 \in \{0\} \cup \{x_s \mid s \in \mathbf{LS}(\ell)\}$
    Set $Z^* \leftarrow \min_{\ell,p,o_1,o_2} Z_{t,\ell,p,o_1,o_2}$
    Set $(\ell^*, p^*, o_1^*, o_2^*) \leftarrow \arg\min_{(\ell,p,o_1,o_2)} Z_{t,\ell,p,o_1,o_2}$
    if $Z^* < Z_c$ **then**
      Set $Z_c \leftarrow Z^*$
      Set $t \leftarrow$ GrowTree$(t, \ell^*, p^*, o_1^*, o_2^*)$
    else
      break
  return t, Z_c

each column labeled with $L = ...$ corresponds to the leaf-based heuristic column generation method. In general, for a fixed value of L, the leaf-based heuristic column generation method attains roughly the same or slightly lower KL divergence than the depth d that corresponds to that value of L. The reason for this difference is because the leaf-based procedure can select from a larger set of trees: specifically, a tree of maximum depth d will have at most 2^{d-1} leaves, but a tree with at most 2^{d-1} leaves could have maximum depth greater than d. (For example, a ranking with a consideration set of size 7 corresponds to a tree with 8 leaves and a depth of 8, which is deeper than a balanced tree of depth 4.) In some cases, we observe that for higher values of L the performance of the leaf-based method can deteriorate slightly relative to lower values of L. Overall, these results suggest that heuristic column generation with the leaf-based top-down induction method of Section A.2.3 is also a viable method for learning the decision forest model from data.

A.3 Additional Results on the IRI Dataset

A.3.1 Runtime and model size results

Table A.4 shows the average runtime over the five folds for each of the methods. To simplify the exposition, we focus on the LC-MNL, ranking and decision forest models (see Section 3.4.2 for the details of the estimation for each model); for the ordinary MNL and HALO-MNL models, the average runtime over all product categories was less than 0.01 and 2 seconds, respectively. Note that for the LC-MNL, ranking and DF models, this time includes the time to perform k-fold cross-validation in order to tune the number of classes, the maximum consideration set size and the depth, respectively. For the DF model, we also note that the runtime includes the time required to estimate the ranking model as a warm start.

From this table, we can see that the ranking-based model requires the most amount of time – on average 3128 seconds (almost one hour) – due to the use of k-fold cross-validation to tune the maximum consideration set size and the use of integer programming to solve

Product Category	$d=3$	$L=4$	$d=4$	$L=8$	$d=5$	$L=16$	$d=6$	$L=32$	$d=7$	$L=64$
Beer	1.36	0.79	0.85	2.54	1.98	2.22	2.88	2.63	3.05	2.34
Blades	0.43	0.39	0.36	0.68	0.49	1.05	0.83	0.71	1.18	0.74
Carbonated Beverages	1.60	0.79	0.86	1.72	1.30	1.88	12.53	1.77	1.31	1.77
Cigarettes	1.37	0.72	0.78	0.89	1.06	1.35	0.86	1.58	2.15	1.79
Coffee	1.95	1.71	1.80	1.93	2.34	3.02	2.68	4.00	3.74	6.34
Cold Cereal	0.93	0.67	0.75	0.93	0.82	1.39	1.20	1.22	1.12	1.22
Deodorant	0.44	0.60	0.64	0.56	0.47	1.46	1.22	0.85	1.88	1.29
Diapers	0.82	1.36	1.37	1.58	1.22	1.69	1.40	1.70	1.63	1.70
Facial Tissue	0.78	0.73	0.75	0.84	0.76	0.98	1.05	1.16	0.96	1.15
Frozen Dinners	1.76	1.81	2.37	3.54	4.12	2.88	3.34	1.90	2.34	1.87
Frozen Pizza	1.15	1.01	0.93	1.17	1.05	1.53	1.47	1.66	1.55	1.94
Hotdogs	3.06	2.80	2.81	2.39	2.55	2.63	2.88	3.31	4.35	4.28
Household Cleaners	0.68	0.37	0.55	1.40	1.24	5.57	2.71	5.72	4.10	5.72
Laundry Detergent	2.13	2.27	2.27	2.25	2.54	2.27	3.00	2.74	3.21	3.10
Margarine/Butter	1.37	0.67	0.85	1.07	1.22	1.50	1.54	1.50	1.46	1.50
Mayonnaise	0.84	0.82	0.83	0.89	0.93	0.94	0.88	0.93	1.00	1.01
Milk	1.29	1.27	1.77	1.99	2.37	2.53	2.53	3.25	3.03	3.40
Mustard/Ketchup	0.72	0.81	0.68	0.67	0.62	0.84	0.88	0.99	1.05	0.91
Paper Towels	1.03	1.18	1.45	1.66	1.77	1.76	1.52	1.58	1.82	1.68
Peanut Butter	1.49	1.55	3.09	2.46	4.85	1.69	1.72	1.68	1.82	1.65
Photo	1.29	1.32	1.43	1.38	1.45	1.27	1.41	1.49	1.44	1.35
Salty Snacks	1.72	1.69	1.76	1.67	1.66	1.84	1.89	1.98	2.34	2.13
Shampoo	0.93	0.64	0.66	0.56	0.65	0.72	1.04	0.80	1.29	1.00
Soup	0.96	1.39	1.61	1.23	1.10	1.48	1.39	1.66	1.59	1.66
Spaghetti/Italian Sauce	2.69	3.12	3.93	3.02	2.40	3.32	3.44	2.86	3.88	3.09
Sugar Substitutes	0.77	0.80	0.78	0.72	0.72	0.85	0.66	0.84	0.95	0.93
Toilet Tissue	1.37	1.56	1.63	1.85	1.91	1.99	2.01	1.91	1.90	1.92
Toothbrush	1.00	0.64	0.68	1.01	1.19	1.67	1.66	1.44	1.97	3.67
Toothpaste	0.35	0.36	0.37	0.35	0.40	0.48	0.61	0.55	0.47	0.61
Yogurt	2.78	2.31	2.87	3.53	6.42	3.88	4.85	4.34	3.80	4.19

Table A.3: Comparison of leaf-based and depth-based heuristic column generation for the IRI data set.

Product Category	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{T}\rvert$	LC-MNL	RM	DF (HCG)	DF (RTS)
Beer	55	380,932	577.8	4054.8	207.6	385.2
Blades	57	92,404	651.9	3725.8	112.5	531.5
Carbonated Beverages	31	721,506	680.6	1826.4	88.6	165.4
Cigarettes	68	249,668	1206.3	5188.8	211.5	544.1
Coffee	47	372,536	1365.3	2508.7	159.2	282.8
Cold Cereal	15	577,236	347.3	613.1	23.3	48.2
Deodorant	45	271,286	144.4	3366.2	124.4	204.3
Diapers	18	143,055	353.4	452.5	45.1	112.0
Facial Tissue	43	73,806	867.0	2155.8	107.6	547.8
Frozen Dinners	30	979,936	348.9	1765.3	88.0	128.1
Frozen Pizza	61	292,878	1648.3	4081.2	257.9	529.8
Hotdogs	100	101,624	2013.7	5733.0	361.0	640.6
Household Cleaners	19	282,981	286.5	684.2	32.8	52.6
Laundry Detergent	56	238,163	1500.3	5136.1	240.6	614.5
Margarine/Butter	18	140,969	585.2	1563.2	37.6	58.8
Mayonnaise	48	97,282	741.0	2308.6	111.3	367.3
Milk	49	240,691	1568.0	2570.2	139.8	258.9
Mustard/Ketchup	44	134,800	872.3	2565.3	116.1	295.8
Paper Towels	40	82,636	701.2	2605.3	130.0	284.1
Peanut Butter	51	108,770	1109.4	1839.3	109.1	392.3
Photo	80	17,047	999.6	3298.5	109.3	507.9
Salty Snacks	39	736,148	1047.4	2501.0	114.9	241.6
Shampoo	66	290,429	313.7	3638.0	200.9	363.2
Soup	24	905,541	337.7	1507.7	62.4	114.0
Spaghetti/Italian Sauce	38	276,860	1144.6	3581.3	188.5	273.6
Sugar Substitutes	64	53,834	841.3	3816.8	184.3	511.0
Toilet Tissue	27	112,788	534.7	2333.7	98.2	185.9
Toothbrush	114	197,676	1013.9	13652.5	670.2	850.9
Toothpaste	42	238,271	273.0	1516.2	40.2	173.1
Yogurt	43	499,203	1493.8	3274.7	145.0	310.7
(Mean)			852.3	3128.8	150.6	332.5
(Median)			791.2	2567.7	115.5	290.0
(Maximum)			2013.7	13652.5	670.2	850.9

Table A.4: Runtime (in seconds) for the estimation of each predictive model for each of the thirty product categories in the IRI data set.

the subproblem at each step of the algorithm of [119]. The LC-MNL model is the second highest, requiring just under 15 minutes on average, due to the EM algorithm which requires the estimation problem for MNL to be solved repeatedly and the use of k-fold cross validation. The decision forest model requires on average 150 seconds using the HCG approach and on average 333 seconds using the RTS approach. Note that as mentioned earlier, this includes the time needed to estimate the ranking-based model and for the cross-validation to choose the depth d for the decision forest model (the depth limit for the HCG approach, or the depth of the base forest for the RTS approach). The main takeaway from these results is that the estimation of the decision forest model can be accomplished with manageable computation times.

Lastly, we also compare the size of the models. Table A.5 reports several metrics of model size for the LC-MNL, ranking and decision forest models. For the LC-MNL model, we report the average number of segments K chosen by cross-validation; for the ranking-based model, we report the average number of rankings K and the average maximum consideration set size chosen by cross-validation; and for the decision forest models, we report the average number of trees K, the average cross-validated depth d (either the depth limit for HCG or the depth of the base forest for RTS) and the average number of leaves per tree L. From this table, we can see that the number of trees in the decision forest model obtained via the heuristic column generation method is comparable to the number of rankings in the ranking-based model. (For the randomized tree sampling method, the number of trees is slightly over 2000, as this number includes the 2000 trees that are randomly sampled, and the additional rankings that were used for warm-starting.) For the decision forest model, the model size varies by product category. For the HCG-based decision forest model, the average cross-validated depth can be as low as 3.0 and as high as 5.0, and the average number of leaves per tree varies from 4.0 to 9.5. For the RTS-based decision forest model, the average cross-validated depth similarly ranges between 3.0 and 4.4, while the average number of leaves per tree varies from 4.1 to 18.2 (note that the number of leaves is generally larger

than it is for HCG, as the base forest was specified as the collection of balanced trees of the chosen depth, whereas the HCG method is allowed to estimate unbalanced trees).

Product Category	LC-MNL	— RM —		— DF (HCG) —			— DF (RTS) —		
	K	K	τ	K	d	L	K	d	L
Beer	8.8	202.0	3.4	331.6	3.8	7.7	2208.0	3.8	8.1
Blades	8.4	139.2	4.4	218.6	4.2	8.6	2185.2	3.6	9.6
Carbonated Beverages	10.0	118.4	2.4	274.2	3.8	5.6	2123.2	4.2	11.7
Cigarettes	12.0	162.2	4.4	252.2	3.8	6.8	2169.8	3.6	6.5
Coffee	13.0	142.4	3.4	298.4	3.2	5.6	2145.4	4.2	9.5
Cold Cereal	4.0	70.0	2.8	149.2	3.2	5.7	2072.8	3.2	4.9
Deodorant	6.2	150.8	4.6	200.8	4.2	9.5	2148.0	3.0	4.4
Diapers	5.0	140.8	6.2	219.0	3.6	6.8	2148.2	3.0	4.3
Facial Tissue	13.0	258.4	4.2	319.2	3.4	7.8	2278.4	3.2	5.2
Frozen Dinners	5.4	110.0	2.4	230.4	3.8	7.3	2111.4	4.2	17.8
Frozen Pizza	14.0	210.0	5.4	351.4	3.6	7.0	2216.0	3.2	5.1
Household Cleaners	4.6	65.0	4.8	144.8	3.2	4.6	2065.0	3.2	4.8
Hotdogs	11.6	152.4	3.8	449.8	3.4	5.5	2145.4	4.4	18.2
Laundry Detergent	7.0	157.8	5.2	284.6	3.0	6.0	2156.4	3.6	9.4
Margarine/Butter	9.8	168.8	2.8	240.8	4.4	7.6	2173.8	3.2	5.0
Mayonnaise	9.4	191.0	4.4	263.0	3.0	6.6	2187.6	3.0	4.3
Milk	6.2	89.2	2.0	254.8	3.0	4.0	2116.8	3.2	4.8
Mustard/Ketchup	9.6	170.8	3.4	321.2	3.6	6.5	2176.8	3.4	5.8
Paper Towels	5.6	266.6	3.0	291.4	3.6	7.7	2244.8	3.0	4.4
Peanut Butter	9.4	134.0	5.0	204.4	3.0	6.5	2141.2	3.2	5.0
Photo	2.8	164.6	7.8	184.0	3.2	8.0	2159.8	3.4	5.8
Salty Snacks	7.8	125.8	3.2	276.2	4.0	6.6	2127.0	3.6	7.1
Shampoo	6.6	206.8	4.8	288.0	4.0	9.0	2208.4	4.0	11.0
Soup	8.2	102.2	5.8	223.0	3.6	6.7	2103.4	3.4	5.7
Spaghetti/Italian Sauce	6.2	213.4	3.6	414.4	3.4	6.4	2213.2	4.2	12.3
Sugar Substitutes	11.0	170.2	4.0	244.8	3.6	7.7	2190.6	3.4	5.8
Toilet Tissue	4.6	245.0	2.4	310.0	5.0	8.6	2231.4	3.0	4.4
Toothbrush	10.6	294.4	2.8	429.2	4.2	9.0	2303.6	3.8	10.1
Toothpaste	5.4	84.0	3.0	133.4	3.0	6.5	2083.0	3.0	4.2
Yogurt	14.0	167.2	4.8	267.6	3.2	5.4	2144.0	3.0	4.2

Table A.5: Model size metrics for the LC-MNL, ranking and decision forest models for each of the thirty product categories in the IRI data set.

APPENDIX B

Appendix to Chapter 4

B.1 Benders cuts for integer master solutions

In this section, we provide closed form expressions for the structure of the optimal primal and dual Benders subproblem solutions for integer solutions $\mathbf{x}$, for LeafMIO (Section B.1.1), SplitMIO (Section B.1.2) and ProductMIO (Section B.1.3).

B.1.1 Benders cuts for integer solutions of LeafMIO

Our results in Section 4.2.1 for obtaining primal and dual solutions for the subproblem of LeafMIO apply for any $\mathbf{x} \in [0,1]^n$; in particular, they apply for fractional choices of $\mathbf{x}$, thus allowing us to solve the Benders reformulation of the relaxation of LeafMIO (presented as problem (4.15)).

In the special case that $\mathbf{x}$ is integer, the optimal solutions to the primal and dual subproblems can be obtained more directly than by applying Algorithms 4 and 5; more specifically, they can be obtained in closed form. We formalize this as the following theorem.

Theorem 19 *Fix $t \in F$, and let $\mathbf{x} \in \{0,1\}^n$. Define the primal subproblem solution $\mathbf{y}_t$ as*

$$y_{t,\ell} = \begin{cases} 1 & \text{if } \ell = \ell^*, \\ 0 & \text{if } \ell \neq \ell^*, \end{cases}$$

where ℓ^* denotes the leaf of tree t that the assortment encoded by $\mathbf{x}$ is mapped to. Define the

dual subproblem solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *as*

$$\alpha_{t,s,\ell} = \begin{cases} \max\{0, r_{t,\ell} - r_{t,\ell^*}\} & \text{if } s \in \mathbf{RS}(\ell^*),\ \ell \in \mathbf{left}(s), \\ 0 & \text{otherwise}, \end{cases}$$

$$\beta_{t,s,\ell} = \begin{cases} \max\{0, r_{t,\ell} - r_{t,\ell^*}\} & \text{if } s \in \mathbf{LS}(\ell^*),\ \ell \in \mathbf{right}(s), \\ 0 & \text{otherwise}, \end{cases}$$

$$\gamma_t = r_{t,\ell^*}.$$

Then:

a) $\mathbf{y}_t$ *is a feasible solution to problem* (4.13)*;*

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *is a feasible solution to problem* (4.14)*; and*

c) $\mathbf{y}_t$ *and* $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *are optimal for problems* (4.13) *and* (4.14)*, respectively.*

The significance of Theorem 19 is that it provides a simpler means to checking for violated constraints when $\mathbf{x}$ is binary than applying Algorithms 4 and 5. In particular, for the integer version of LeafMIO, a similar derivation as in Section 4.2.1 leads us to the following Benders reformulation of the integer problem for the LeafMIO formulation:

$$\underset{\mathbf{x},\boldsymbol{\theta}}{\text{maximize}} \quad \sum_{t \in F} \lambda_t \theta_t \tag{B.1a}$$

$$\text{subject to} \quad \theta_t \leq \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} x_{v(t,s)} + \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{right}(s)} \beta_{t,s,\ell}(1 - x_{v(t,s)}) + \gamma_t,$$

$$\forall\ (\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t) \in \mathcal{D}_{t,\text{LeafMIO}}, \tag{B.1b}$$

$$\mathbf{x} \in \{0,1\}^n. \tag{B.1c}$$

To check whether constraint (B.1b) is violated for a particular $\mathbf{x}$ and a tree t, we can simply use Theorem 19 to determine the optimal value of the subproblem, and compare it against θ_t; if the constraint corresponding to the dual solution of Theorem 19 is violated, we add that constraint to the problem. In our implementation of Benders decomposition, we embed the

constraint generation process for the integer problem (B.1) within the branch-and-bound tree, using a technique referred to as *lazy constraint generation*; we discuss this more in Section 4.2.4.

B.1.2 Benders cuts for integer solutions of SplitMIO

We next turn our attention to SplitMIO. In the special case that $\mathbf{x}$ is a candidate integer solution of SplitMIO, we can find optimal solutions to the primal and dual subproblems of SplitMIO in closed form, analogously to Theorem 19 for the primal and dual subproblems of LeafMIO.

Theorem 20 *Fix $t \in F$, and let $\mathbf{x} \in \{0,1\}^n$. Define the primal subproblem solution $\mathbf{y}_t$ as*

$$y_{t,\ell} = \begin{cases} 1 & \text{if } \ell = \ell^*, \\ 0 & \text{if } \ell \neq \ell^*, \end{cases}$$

where ℓ^* denotes the leaf that the assortment encoded by $\mathbf{x}$ is mapped to. Define the dual subproblem solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ as

$$\alpha_{t,s} = \begin{cases} \max\{0, \max_{\ell \in \mathbf{left}(s)} r_{t,\ell} - r_{t,\ell^*}\} & \text{if } s \in \mathbf{RS}(\ell^*), \\ 0 & \text{otherwise}, \end{cases}$$

$$\beta_{t,s} = \begin{cases} \max\{0, \max_{\ell \in \mathbf{right}(s)} r_{t,\ell} - r_{t,\ell^*}\} & \text{if } s \in \mathbf{LS}(\ell^*), \\ 0 & \text{otherwise}, \end{cases}$$

$$\gamma_t = r_{t,\ell^*}.$$

Then:

a) $\mathbf{y}_t$ is a feasible solution to problem (4.17);

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to problem (4.18); and

c) $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for problems (4.17) and (4.18), respectively.

As with Theorem 19, the significance of this theorem is that it provides an even simpler approach than Algorithms 6 and 7 for identifying violated constraints when dealing with integer solutions $\mathbf{x}$.

B.1.3 Benders cuts for integer solutions of ProductMIO

We finally consider ProductMIO. We begin by writing down the dual of the subproblem, for which we need to define several additional sets. We let $\mathbf{LP}(\ell)$ denote the set of "left products" of leaf ℓ (those products that must be included in the assortment for leaf ℓ to be reached), and let $\mathbf{RP}(\ell)$ denote the set of "right products" of leaf ℓ (those products that must be excluded from the assortment for leaf ℓ to be reached). Note that $\ell \in \mathbf{left}(i)$ if and only if $i \in \mathbf{LP}(\ell)$, and similarly $\ell \in \mathbf{right}(i)$ if and only if $i \in \mathbf{RP}(\ell)$.

With these definitions, the dual of the primal subproblem (4.20) is

$$\underset{\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t}{\text{minimize}} \quad \sum_{i \in P(t)} \alpha_{t,i} x_i + \sum_{i \in P(t)} \beta_{t,i}(1 - x_i) + \gamma_t \tag{B.2a}$$

$$\text{subject to} \quad \sum_{i \in \mathbf{LP}(\ell)} \alpha_{t,i} + \sum_{i \in \mathbf{RP}(\ell)} \beta_{t,i} + \gamma_t \geq r_\ell, \quad \forall\, \ell \in \mathbf{leaves}(t), \tag{B.2b}$$

$$\alpha_{t,i} \geq 0, \quad \forall\, i \in P(t), \tag{B.2c}$$

$$\beta_{t,i} \geq 0, \quad \forall\, i \in P(t). \tag{B.2d}$$

In the case that $\mathbf{x}$ is integer, we can obtain optimal solutions to the primal subproblem (4.20) and its dual (B.2) in closed form.

Theorem 21 *Fix $t \in F$ and let $\mathbf{x} \in \{0,1\}^n$. Let $\mathbf{y}_t$ be defined as*

$$y_{t,\ell} = \begin{cases} 1 & \text{if } \ell = \ell^*, \\ 0 & \text{otherwise}, \end{cases} \tag{B.3}$$

and let $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *be defined as*

$$\alpha_{t,i} = \begin{cases} \max\{0, \max_{\ell \in \mathbf{left}(i)} r_{t,\ell} - r_{t,\ell^*}\}, & \text{if } i \in \mathbf{RP}(\ell^*), \\ 0 & \text{otherwise}, \end{cases} \tag{B.4}$$

$$\beta_{t,i} = \begin{cases} \max\{0, \max_{\ell \in \mathbf{right}(i)} r_{t,\ell} - r_{t,\ell^*}\}, & \text{if } i \in \mathbf{LP}(\ell^*), \\ 0 & \text{otherwise}, \end{cases} \tag{B.5}$$

$$\gamma_t = r_{t,\ell^*}. \tag{B.6}$$

Then:

a) $\mathbf{y}_t$ *is a feasible solution for problem* (4.20)*;*

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *is a feasible solution for problem* (B.2)*; and*

c) $\mathbf{y}_t$ *and* $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *are optimal for problems* (4.20) *and* (B.2)*, respectively.*

B.2 Omitted Proofs

B.2.1 Proof of Proposition 2

We prove this by showing that MAX 3SAT problem reduces to the decision forest assortment optimization problem. In the MAX 3SAT problem, one has K binary variables, $x_1, \ldots, x_K$, and is given a Boolean formula of the form $c_1 \wedge c_2 \wedge \cdots \wedge c_M$, where $\wedge$ denotes "and". Each clause is a disjunction involving three literals, where a literal is either one of the binary variables or its negation, and the literals involve distinct binary variables. For example, a clause could be $x_5 \vee \neg x_7 \vee x_8$, where $\vee$ denotes "or". The MAX 3SAT problem is to find the assignment of the variables $x_1, \ldots, x_K$ so as to maximize the number of clauses $c_1, \ldots c_T$ that are true.

Given an instance of the MAX 3SAT problem, we show how the problem can be transformed into an instance of the decision forest assortment optimization problem (4.2).

Consider an instance of problem (4.2) with $N = K+1$ products. Each of the first K products corresponds to one of the binary variables; the last $(K+1)$-th product is necessary to ensure that the revenue of the assortment can correspond to the number of satisfied clauses. Assume that the marginal revenues of the products are set so that $\bar{r}_1 = \cdots = \bar{r}_K = 0$, and $\bar{r}_{K+1} = 1$. For each clause $m \in \{1, \ldots, M\}$, introduce a tree t_m which is constructed by the following procedure:

1. Set the root node of the tree to be a split node involving product $K+1$. The right child of the root node is a leaf node with 0 (the no-purchase option) as the purchase decision. (The left child node will be defined in the next step.)

2. For the first literal, create a split at the left child node of the root, with the corresponding binary variable's index as the product (e.g., if the literal is x_7 or $\neg x_7$, the split node has product 7). If the literal is the binary variable itself (i.e., x_k), we set the left child node of the split to be a leaf node with product $K+1$ as the purchase decision. Otherwise, if the literal is the negation of the binary variable (i.e., $\neg x_k$), we set the right child node of the split to be a leaf node with product $K+1$ as the purchase decision.

3. For the other child node of the split created in Step 2, repeat Step 2 with the second literal.

4. For the other child node of the split created in Step 3, repeat Step 2 with the third literal.

5. Lastly, for the other child node of the third split node in Step 4 corresponding to the third literal, set the purchase decision to be 0.

Figure B.1 visualizes the structure of the tree for the example clause $x_5 \vee \neg x_7 \vee x_8$. Applying

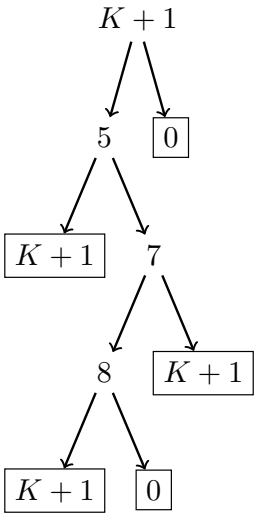


Figure B.1: Example of tree that corresponds to the clause $x_5 \vee \neg x_7 \vee x_8$.

the above procedure for each clause results in a forest F consisting of M trees. Lastly, we set the probability distribution $\boldsymbol{\lambda}$ by setting $\lambda_t = 1/M$ for each tree $t \in F$.

We note that in the resulting instance of problem (4.2), any optimal assortment must include produce $K+1$: due to the structure of the trees, the expected revenue is exactly equal to 0 if $K+1$ is not included in the assortment, but by including $K+1$ and including/excluding products from $\{1, \ldots, K\}$ in accordance with one of the clauses, one can obtain an expected revenue of at least $1/M$. (For example, if the set of clauses includes the clause $x_5 \vee \neg x_7 \vee x_8$ shown in Figure B.1, then any assortment S that includes products 5 and 8 and does not include product 7 automatically has an expected revenue of at least $1/M$.)

Note that given an optimal assortment $S \subseteq \mathcal{N} = \{1, \ldots, K+1\}$, we immediately obtain an assignment $\mathbf{x}$ for the MAX 3SAT problem by setting $x_i = \mathbb{I}\{i \in S\}$. The revenue obtained from each tree t_m corresponding to clause m, which is given by $\sum_{j=1}^{K+1} \bar{r}_j \mathbb{I}\{\hat{A}(t_m, S) = j\}$, is exactly 1 if the assignment $\mathbf{x}$ that corresponds to S satisfies clause m, and 0 otherwise; this holds because the optimal assortment must include product $K+1$. Since each tree t_m has a probability of $1/M$, the expected revenue of the assortment S therefore corresponds to the

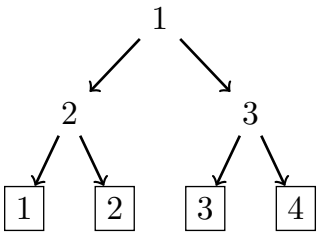


Figure B.2: Purchase decision tree for proof of Proposition 3.

fraction of the M clauses that are satisfied by the assignment $\mathbf{x}$. Thus, it follows that an assignment $\mathbf{x}$ that corresponds to an optimal assortment S is an optimal solution of MAX 3SAT. Since the MAX 3SAT problem is NP-Complete [65], it follows that problem (4.2) is NP-Hard. □

B.2.2 Proof of Proposition 3

Consider a decision forest consisting of only one tree, of the form shown in Figure B.2. The numbers inside the split nodes indicate the split product (i.e., $v(t, s)$). The numbers inside the leaf nodes (enclosed in squares) index the leaves (i.e., the leaves are numbered from 1 to 4). The feasible region of the LO relaxation of problem (4.3) is the set of solutions

$(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^2 \times \mathbb{R}^4$ given by the following family of constraints:

$$y_1 \leq x_1, \tag{B.7a}$$

$$y_1 \leq x_2, \tag{B.7b}$$

$$y_2 \leq x_1, \tag{B.7c}$$

$$y_2 \leq 1 - x_2, \tag{B.7d}$$

$$y_3 \leq 1 - x_1, \tag{B.7e}$$

$$y_3 \leq x_3, \tag{B.7f}$$

$$y_4 \leq 1 - x_1, \tag{B.7g}$$

$$y_4 \leq 1 - x_3, \tag{B.7h}$$

$$y_1 + y_2 + y_3 + y_4 = 1, \tag{B.7i}$$

$$x_1 \leq 1, \tag{B.7j}$$

$$x_2 \leq 1, \tag{B.7k}$$

$$x_3 \leq 1, \tag{B.7l}$$

$$x_1, x_2, x_3, y_1, y_2, y_3, y_4 \geq 0. \tag{B.7m}$$

(Note that for simplicity, we drop the subscript t from all of the y variables.)

It can be verified that the following solution is an extreme point of this polyhedron:

$$x_1 = 0.5, \tag{B.8a}$$

$$x_2 = 0.5, \tag{B.8b}$$

$$x_3 = 0, \tag{B.8c}$$

$$y_1 = 0.5, \tag{B.8d}$$

$$y_2 = 0.5, \tag{B.8e}$$

$$y_3 = 0, \tag{B.8f}$$

$$y_4 = 0. \tag{B.8g}$$

Since this extreme point $(\mathbf{x}, \mathbf{y})$ does not satisfy $\mathbf{x} \in \{0,1\}^3$, we conclude that $\mathcal{F}_{\text{LeafMIO}}$ is not integral in general. $\square$

B.2.3 Proof of Proposition 5

The feasible region $\mathcal{F}_{\text{SplitMIO}}$ of the LO relaxation of problem (4.4) is the set of $(\mathbf{x}, \mathbf{y})$ solutions to the following system of inequalities:

$$\sum_{\ell \in \textbf{leaves}} y_\ell \leq 1, \tag{B.9a}$$

$$\sum_{\ell \in \textbf{leaves}} -y_\ell \leq -1, \tag{B.9b}$$

$$\sum_{\ell \in \textbf{left}(s)} y_\ell - x_{v(s)} \leq 0, \quad \forall\ s \in \textbf{splits}, \tag{B.9c}$$

$$\sum_{\ell \in \textbf{right}(s)} y_\ell + x_{v(s)} \leq 1, \quad \forall\ s \in \textbf{splits}, \tag{B.9d}$$

$$x_i \leq 1, \quad \forall\ i \in \mathcal{N}, \tag{B.9e}$$

$$x_i \geq 0, \quad \forall\ i \in \mathcal{N}, \tag{B.9f}$$

$$y_\ell \geq 0, \quad \forall\ \ell \in \textbf{leaves}. \tag{B.9g}$$

(Since $|F| = 1$, we drop the index t to simplify the notation.) In the above, note that the unit sum constraint on $\mathbf{y}$ has been re-written as a pair of inequalities, and that all constraints from problem (4.4) have been re-arranged to have the variables on one side. This system of inequalities can be further written compactly as

$$\mathbf{A} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \leq \mathbf{b}, \tag{B.10}$$

$$\mathbf{x}, \mathbf{y} \geq \mathbf{0}. \tag{B.11}$$

To show that $\mathcal{F}_{\text{SplitMIO}}$ is integral, we will prove that the matrix $\mathbf{A}$ is totally unimodular. We do so using the following standard characterization of total unimodularity (see [16]):

Proposition 15 (Corollary 3.2 of [16]) *A matrix* $\mathbf{A}$ *is totally unimodular if and only if each collection* Q *of rows of* $\mathbf{A}$ *can be partitioned into two parts so that the sum of the rows in one part minus the sum of the rows in the other part is a vector with entries only 0, +1, and -1.*

To simplify our notation, we will work with algebraic expressions in terms of $\mathbf{x}$ and $\mathbf{y}$ rather than rows of the matrix $\mathbf{A}$. We have the following four primitive expressions:

$$A(s), s \in \mathbf{splits}: \quad \sum_{\ell \in \mathbf{left}(s)} y_\ell - x_{v(s)}, \tag{B.12}$$

$$B(s), s \in \mathbf{splits}: \quad \sum_{\ell \in \mathbf{right}(s)} y_\ell + x_{v(s)}, \tag{B.13}$$

$$C(i), i \in \mathcal{N}: \quad x_i, \tag{B.14}$$

$$D(1): \quad \sum_{\ell \in \mathbf{leaves}} y_\ell, \tag{B.15}$$

$$D(2): \quad \sum_{\ell \in \mathbf{leaves}} -y_\ell. \tag{B.16}$$

Thus, a collection of rows Q of the matrix $\mathbf{A}$ can be viewed as a collection of each of the four types of expressions above:

$$S_A \subseteq \mathbf{splits}, \tag{B.17}$$

$$S_B \subseteq \mathbf{splits}, \tag{B.18}$$

$$S_C \subseteq \mathcal{N}, \tag{B.19}$$

$$S_D \subseteq \{1, 2\}. \tag{B.20}$$

To establish the condition in Proposition 15, we need to show that given S_A, S_B, S_C, S_D, we can partition these expressions into two groups R_+ and R_- such that the difference of the two groups,

$$\sum_{e \in R_+} e - \sum_{e \in R_-} e = \sum_{i \in \mathcal{N}} v_i x_i + \sum_{\ell \in \mathbf{leaves}} w_\ell y_\ell, \tag{B.21}$$

is such that each $v_i \in \{-1, 0, +1\}$ and each $w_\ell \in \{-1, 0, +1\}$. We proceed in several steps.

Step 1. We begin by assigning the A and B expressions. Before doing so, we require some additional notation. Define $d^* = \max_{s \in S_A \cup S_B} d(s)$, where $d(s)$ is the depth of split s and we assume the depth of the root node is 1. Let us define the sets $S_A(d)$ and $S_B(d)$ as

$$S_A(d) = \{s \in S_A \mid d(s) = d\}, \tag{B.22}$$

$$S_B(d) = \{s \in S_B \mid d(s) = d\}, \tag{B.23}$$

for each depth $d \in \{1, \ldots, d^*\}$. These are the sets of splits in S_A and S_B, respectively, that are at a particular depth. Let us also define $\mathbf{LD}(s)$ and $\mathbf{RD}(s)$ to be the sets of splits in $S_A \cup S_B$ that are to the left and right, respectively, of split $s \in S_A \cup S_B$. (The splits in $\mathbf{LD}(s)$ are all those that can be reached by proceeding to the left child of split s; similarly, $\mathbf{RD}(s)$ is the set of splits reachable by going to the right of split s.) Finally, define $\sigma : S_A \cup S_B \to \{-1, +1\}$ to be a mapping that is specified according to the following procedure:

for $d = 1, \ldots, d^*$ **do**
  for $s \in S_A(d)$ **do**
    Set $\sigma(s') = (-1)\sigma(s)$ for $s' \in \mathbf{LD}(s)$
  for $s \in S_B(d)$ **do**
    Set $\sigma(s') = (-1)\sigma(s)$ for $s' \in \mathbf{RD}(s)$

Now, assign the A and B expressions as follows:

- Assign $A(s)$ to R_+ for each $s \in S_A$ with $\sigma(s) = +1$;
- Assign $A(s)$ to R_- for each $s \in S_A$ with $\sigma(s) = -1$;
- Assign $B(s)$ to R_+ for each $s \in S_B$ with $\sigma(s) = +1$;
- Assign $B(s)$ to R_- for each $s \in S_B$ with $\sigma(s) = -1$.

For this assignment of the expressions in S_A and S_B, every y_ℓ coefficient in $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ will be either 0 or $+1$. This follows because the sets of left and right leaves $\mathbf{left}(s)$ and

right(s) are nested. In particular, if $s' \in \mathbf{LD}(s)$, then we will have that **left**$(s') \subseteq$ **left**(s) and **right**$(s') \subseteq$ **left**(s). Similarly, if $s' \in \mathbf{RD}(s)$, then we will have that **left**$(s') \subseteq$ **right**(s) and **right**$(s') \subseteq$ **right**(s).

In addition, by the assumption that there is at most one split s in **splits** such that $v(s) = i$, we are also guaranteed that the coefficient of every x_i in $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ will be $\{-1, 0, +1\}$. In particular, if s is in both S_A and S_B (i.e., we were given the expression $A(s)$ and $B(s)$), then observe that by the procedure for setting σ above, we are guaranteed to assign both $A(s)$ and $B(s)$ to the same set (they cannot be assigned to different sets). This means that the coefficients of $x_{v(s)}$ in $A(s)$ and $B(s)$ will cancel out, leaving $x_{v(s)}$ with a coefficient of 0.

Step 2. We next assign the C expressions. After Step 1, we are guaranteed that the coefficient of each x_i in $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ is 0, -1 or +1. Since each $C(i)$ expression involves only one variable (x_i), it is straightforward to assign these expressions to R_+ and R_- to ensure that every variable's coefficient in $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ is 0, -1 or +1. For completeness, we give the procedure below – for each $i \in S_C$:

- If $v(s) \neq i$ for all splits $s \in$ **splits**, then x_i does not appear in any A or B expressions and its coefficient after Step 1 is just 0; thus, $C(i)$ can be arbitrarily assigned to R_+ or R_-.
- If there exists an $s \in S_A \cup S_B$ such that $v(s) = i$, then:
  - If $s \in S_A \cup S_B$, then $A(s)$ and $B(s)$ were both assigned to R_+ and R_-, and so the coefficient of x_i will be 0 due to cancellation; thus, $C(i)$ can again be arbitrarily assigned to R_+ or R_-.
  - If $s \in S_A$, $s \notin S_B$, and $\sigma(s) = +1$, then the coefficient of x_i is -1 after Step 1; thus, $C(i)$ should be assigned to R_+.

- If $s \in S_A$, $s \notin S_B$, and $\sigma(s) = -1$, then the coefficient of x_i is $+1$ after Step 1; thus, $C(i)$ should be assigned to R_-.
- If $s \notin S_A$, $s \in S_B$, and $\sigma(s) = +1$, then the coefficient of x_i is $+1$ after Step 1; thus, $C(i)$ should be assigned to R_-.
- If $s \notin S_A$, $s \in S_B$, and $\sigma(s) = -1$, then the coefficient of x_i is -1 after Step 1; thus, $C(i)$ should be assigned to R_+.

Step 3. Lastly, we assign the D expressions. This step is also straightforward:

- If $S_D = \{1, 2\}$, then assign $D(1)$ and $D(2)$ to R_+; since $D(1)$ is just the negative of $D(2)$, the two expressions will cancel out, and the expression $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ will remain unchanged.
- If $S_D = \{1\}$, then assign $D(1)$ to R_-; since the coefficient of each y_ℓ is 0 or $+1$ after Step 2, this will ensure that the coefficient of each y_ℓ is either -1 or 0.
- If $S_D = \{2\}$, then assign $D(2)$ to R_-; since the coefficient of each y_ℓ is 0 or $+1$ after Step 2, this will ensure that the coefficient of each y_ℓ is either -1 or 0.

After completing Step 3, we have assigned all of the expressions in S_A, S_B, S_C, S_D to the sets R_+ and R_- in a way that each expression is assigned to exactly one of the two sets, and no expression is unassigned. Moreover, the difference of the two expressions, $\sum_{e \in R_+} e - \sum_{e \in R_-} e$, is such that the coefficient of every x_i and y_ℓ variable is in $\{0, -1, +1\}$. By Proposition 15, this establishes that the matrix $\mathbf{A}$ is totally unimodular. We now employ another standard result:

Proposition 16 (Theorem 3.1(b) of [16]) *Let* $\mathbf{A}$ *be an integer matrix. The matrix* $\mathbf{A}$ *is totally unimodular if and only if the polyhedron* $P(\mathbf{b}) = \{\mathbf{x} \in \mathbb{R}^n_+ \mid \mathbf{Ax} \leq \mathbf{b}\}$ *is integral for all* $\mathbf{b} \in \mathbb{Z}^m$ *for which* $P(\mathbf{b}) \neq \emptyset$.

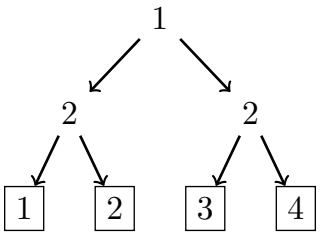


Figure B.3: Purchase decision tree for proof of Proposition 6.

To use this result, we simply have to establish that the feasible region of the polyhedron defined in (B.9) is nonempty.

To do so, we explicitly construct a feasible solution to (B.9). Let r be the root node of the tree. Set $x_i = 0.5$ for all $i \in \mathcal{N}$. Fix any leaf $\ell' \in \mathbf{left}(r)$ and any leaf $\ell'' \in \mathbf{right}(r)$, and set $y_{\ell'} = 0.5$, $y_{\ell''} = 0.5$, and $y_\ell = 0$ for all $\ell \in \mathbf{leaves} \setminus \{\ell', \ell''\}$. It is straightforward to verify that this solution satisfies the system of inequalities (B.9): the y_ℓ's sum to 1 and are nonnegative by construction, and each $x_i \in [0, 1]$ by construction. For constraints (B.9c) and (B.9d), note that since ℓ' and ℓ'' are on opposite sides of the root node, it is impossible for ℓ' and ℓ'' to both belong to $\mathbf{left}(s)$ and $\mathbf{right}(s)$ for any split s; armed with this fact, it is straightforward to establish the two constraints.

Since we have established that $\mathbf{A}$ is totally unimodular and that the set $\mathcal{F}_{\text{SplitMIO}}$ is nonempty, invoking Proposition 16 concludes the proof. □

B.2.4 Proof of Proposition 6

Consider a decision forest consisting of only $|F| = 1$ tree, of the form shown in Figure B.3. The numbers on the split nodes correspond to products that are used for splitting (i.e., $v(t, s)$). The numbers inside the leaf nodes (enclosed in squares) index the leaves (i.e., the leaves are indexed from 1 to 4). The feasible region of the LO relaxation of problem (4.4) is the set of solutions $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^2 \times \mathbb{R}^4$ given by the following family of constraints (where we

again drop the subscript t for simplicity):

$$y_1 \leq x_2, \tag{B.24a}$$

$$y_2 \leq 1 - x_2, \tag{B.24b}$$

$$y_3 \leq x_2, \tag{B.24c}$$

$$y_4 \leq 1 - x_2, \tag{B.24d}$$

$$y_1 + y_2 \leq x_1, \tag{B.24e}$$

$$y_3 + y_4 \leq 1 - x_1, \tag{B.24f}$$

$$y_1 + y_2 + y_3 + y_4 = 1, \tag{B.24g}$$

$$x_1 \leq 1, \tag{B.24h}$$

$$x_2 \leq 1, \tag{B.24i}$$

$$x_1, x_2, y_1, y_2, y_3, y_4 \geq 0. \tag{B.24j}$$

It can be verified that the following solution is an extreme point of this polyhedron:

$$x_1 = 0.5, \tag{B.25a}$$

$$x_2 = 0.5, \tag{B.25b}$$

$$y_1 = 0.5, \tag{B.25c}$$

$$y_2 = 0, \tag{B.25d}$$

$$y_3 = 0, \tag{B.25e}$$

$$y_4 = 0.5. \tag{B.25f}$$

Since this extreme point is not integer, this establishes that even when $|F| = 1$, $\mathcal{F}_{\text{SplitMIO}}$ can be non-integral when a product appears in more than one split. □

B.2.5 Proof of Proposition 8

Define $\Delta^{\mathbf{leaves}}$ to be the $(|\mathbf{leaves}| - 1)$-dimensional unit simplex:

$$\Delta^{\mathbf{leaves}} = \{\mathbf{y} \in \mathbb{R}^{|\mathbf{leaves}|} \mid \sum_{\ell \in \mathbf{leaves}} y_\ell = 1; y_\ell \geq 0, \forall\, \ell \in \mathbf{leaves}\}. \tag{B.26}$$

In addition, for any $S \subseteq \mathbf{leaves}$, define $Q(S) = \{\mathbf{y} \in \Delta^{\mathbf{leaves}} \mid y_\ell \leq 0 \text{ for } \ell \in \mathbf{leaves} \setminus S\}$. We write the combinatorial disjunctive constraint over the ground set **leaves** as

$$\text{CDC}(\mathbf{leaves}) = \bigcup_{\ell \in \mathbf{leaves}} Q(\{\ell\}). \tag{B.27}$$

Consider now the optimization problem

$$\underset{\mathbf{y}}{\text{maximize}} \quad \sum_{\ell \in \mathbf{leaves}} r_\ell y_\ell \tag{B.28a}$$

$$\text{subject to} \quad \mathbf{y} \in \text{CDC}(\mathbf{leaves}). \tag{B.28b}$$

We will re-formulate this problem into a mixed-integer optimization problem. To do this, we claim that CDC(**leaves**) can be written as the following pairwise independent branching scheme:

$$\bigcup_{\ell \in \mathbf{leaves}} Q(\{\ell\}) = \bigcup_{i=1}^{n} \left(Q(L_i) \cup Q(R_i)\right), \tag{B.29}$$

where

$$L_i = \{\ell \in \mathbf{leaves} \mid \ell \in \mathbf{left}(s) \text{ for some } s \text{ with } v(s) = i\}, \tag{B.30}$$

$$R_i = \{\ell \in \mathbf{leaves} \mid \ell \in \mathbf{right}(s) \text{ for some } s \text{ with } v(s) = i\}. \tag{B.31}$$

Note that (B.29) is equivalent to the statement

$$\bigcup_{\ell \in \mathbf{leaves}} \{\ell\} = \bigcup_{i=1}^{n} \left((\mathbf{leaves} \setminus L_i) \cup (\mathbf{leaves} \setminus R_i)\right). \tag{B.32}$$

To establish (B.32), it is sufficient to prove the following equivalence:

$$\{\ell\} = \bigcap_{i \in I(\ell)} (\mathbf{leaves} \setminus R_i) \cap \bigcap_{i \in E(\ell)} (\mathbf{leaves} \setminus L_i) \cap \bigcap_{\substack{i \in \mathcal{N}: \\ i \notin I(\ell) \cup E(\ell)}} (\mathbf{leaves} \setminus L_i), \tag{B.33}$$

where $I(\ell)$ and $E(\ell)$ are defined as

$$I(\ell) = \{i \in \mathcal{N} \mid \ell \in \bigcup_{s:v(s)=i} \textbf{left}(s)\}, \tag{B.34}$$

$$E(\ell) = \{i \in \mathcal{N} \mid \ell \in \bigcup_{s:v(s)=i} \textbf{right}(s)\}. \tag{B.35}$$

We now prove (B.33).

Equation (B.33), $\subseteq$ *direction:* For $i \in I(\ell)$, we have that $\ell \in \bigcup_{s:v(s)=i} \textbf{left}(s)$. This means that there exists $\bar{s}$ such that $\ell \in \textbf{left}(\bar{s})$ and $v(\bar{s}) = i$. Since $\ell \in \textbf{left}(\bar{s})$, this means that $\ell \notin \textbf{right}(\bar{s})$ (a leaf cannot be to the left and to the right of any split). Moreover, ℓ cannot be in $\textbf{right}(s)$ for any other s with $v(s) = i$, because this would mean that product i appears more than once along the path to leaf ℓ, violating Requirement 3 in Definition 6. Therefore, $\ell \in \textbf{leaves} \setminus R_i$ for any $i \in I(\ell)$.

For $i \in E(\ell)$, we have that $\ell \in \bigcup_{s:v(s)=i} \textbf{right}(s)$. This means that there exists a split $\bar{s}$ such that $\ell \in \textbf{right}(\bar{s})$ and $v(\bar{s}) = i$. Since $\ell \in \textbf{right}(\bar{s})$, we have that $\ell \notin \textbf{left}(\bar{s})$. In addition, ℓ cannot be in $\textbf{left}(s)$ for any other s with $v(s) = i$. Therefore, $\ell \in \textbf{leaves} \setminus L_i$ for any $i \in E(\ell)$.

Lastly, for any $i \notin I(\ell) \cup E(\ell)$, note that product i does not appear in any split along the path from the root of the tree to leaf ℓ. Therefore, for any s with $v(s) = i$, it will follow that either $\textbf{left}(s) \subseteq \textbf{left}(s')$ for some s' for which $\ell \in \textbf{right}(s')$, or $\textbf{left}(s) \subseteq \textbf{right}(s')$ for some s' for which $\ell \in \textbf{left}(s')$ – in other words, there is a split s' such that every leaf in $\textbf{left}(s)$ is to one side of s' and ℓ is on the other side of s'. This means that ℓ' cannot be in $\textbf{left}(s)$ for any s with $v(s) = i$, or equivalently, $\ell \in \textbf{leaves} \setminus L_i$ for any $i \notin I(\ell) \cup E(\ell)$.

Equation (B.33), $\supseteq$ *direction:* To prove this, we will prove the contrapositive, which is

$$\{\ell' \in \textbf{leaves} \mid \ell' \neq \ell\} \subseteq \bigcup_{i \in I(\ell)} R_i \cup \bigcup_{i \in E(\ell)} L_i \cup \bigcup_{\substack{i \in \mathcal{N}: \\ i \notin I(\ell) \cup E(\ell)}} L_i.$$

A straightforward result (see Lemma EC.1 from [89]) is that

$$\{\ell' \in \mathbf{leaves} \mid \ell' \neq \ell\} = \bigcup_{s:\ell \in \mathbf{left}(s)} \mathbf{right}(s) \cup \bigcup_{s:\ell \in \mathbf{right}(s)} \mathbf{left}(s).$$

Thus, if $\ell' \neq \ell$, then we have that $\ell' \in \mathbf{right}(s)$ for some s such that $\ell \in \mathbf{left}(s)$, or $\ell' \in \mathbf{left}(s)$ for some s such that $\ell \in \mathbf{right}(s)$. Let $i^* = v(s)$. In the first case, since $\ell \in \mathbf{left}(s)$, we have that $i^* \in I(\ell)$, and we thus have

$$\begin{aligned}
\mathbf{right}(s) &\subseteq \bigcup_{s':v(s')=i^*} \mathbf{right}(s') \\
&= R_{i^*} \\
&\subseteq \bigcup_{i \in I(\ell)} R_i \cup \bigcup_{i \in E(\ell)} L_i \cup \bigcup_{\substack{i' \in \mathcal{N}: \\ i' \notin I(\ell) \cup E(\ell)}} L_i.
\end{aligned}$$

In the second case, since $\ell \in \mathbf{right}(s)$, we have that $i^* \in E(\ell)$, and thus we have

$$\begin{aligned}
\mathbf{left}(s) &\subseteq \bigcup_{s':v(s')=i^*} \mathbf{left}(s') \\
&= L_{i^*} \\
&\subseteq \bigcup_{i \in I(\ell)} R_i \cup \bigcup_{i \in E(\ell)} L_i \cup \bigcup_{\substack{i' \in \mathcal{N}: \\ i' \notin I(\ell) \cup E(\ell)}} L_i.
\end{aligned}$$

This establishes the validity of the pairwise independent branching scheme (B.32). Thus, a valid formulation for problem (B.28) (see formulation (9) in [122], formulation (14) in [123]

and formulation (13) in [71]) is

$$\underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{\ell \in \mathbf{leaves}} r_\ell y_\ell \tag{B.36a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}} y_\ell = 1, \tag{B.36b}$$

$$\sum_{\ell \in L_i} y_\ell \leq x_i, \quad \forall\, i \in \mathcal{N}, \tag{B.36c}$$

$$\sum_{\ell \in R_i} y_\ell \leq 1 - x_i, \quad \forall\, i \in \mathcal{N}, \tag{B.36d}$$

$$y_\ell \geq 0, \quad \forall\, \ell \in \mathbf{leaves}, \tag{B.36e}$$

$$x_i \in \{0, 1\}, \quad \forall\, i \in \mathcal{N}. \tag{B.36f}$$

Observe that, by the definition of L_i and R_i, formulation (B.36) is identical to ProductMIO when $|F| = 1$. By invoking Theorem 1 from [122] with appropriate modifications, we can assert that formulation (B.36) is integral. Therefore, in the special case that $|F| = 1$, formulation ProductMIO is always integral. □

B.2.6 Proof of Theorem 5

Proof of part (a) (feasibility): By definition, the solution produced by Algorithm 4 produces a solution $\mathbf{y}_t$ that never violates constraints (4.13c) and (4.13d). In addition, at each stage of Algorithm (4), the quantities $x_{v(t,s)}$ and $1 - x_{v(t,s)}$ are always nonnegative, and the quantity $1 - \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}$ is never allowed to become negative; thus, the solution $\mathbf{y}_t$ that is produced will satisfy the nonnegativity constraint (4.13e). The only constraint that needs to be verified is the unit sum constraint (4.13b).

Notice that by the definition of Algorithm 4, $\mathbf{y}_t$ will satisfy the unit sum constraint if and only if a C event occurs in Algorithm 4. To show that the unit sum constraint is satisfied, let us assume that it is not. This means that in the execution of Algorithm 4, a C event never occurs, and for every leaf, either a $A_{\ell,s}$ or a $B_{\ell,s}$ event occurs for some s.

For any split node j of a tree t, let $\mathbf{leftchild}(j)$ and $\mathbf{rightchild}(j)$ denote its left and right child nodes respectively, and let $\mathbf{root}(t)$ denote the root node of the tree. Consider the leaf ℓ^* that is obtained by the following procedure:

Procedure 1:

1. Set $j \leftarrow \mathbf{root}(t)$.
2. If $j \in \mathbf{leaves}(t)$, return $\ell^* = j$. Otherwise, go to Step 3.
3. If $x_{v(t,j)} \geq 0.5$, then set $j \leftarrow \mathbf{leftchild}(j)$, and return to Step 2.

   Otherwise, set $j \leftarrow \mathbf{rightchild}(j)$, and return to Step 2.

The leaf ℓ^* that is produced by Procedure 1 is useful for the following reason. Upon termination of Algorithm 4, the hypothesis that a C event never occurs means that we will have

$$y_{t,\ell^*} = \min \left\{ \min_{s:\ell^* \in \mathbf{left}(s)} x_{v(t,s)}, \min_{s:\ell^* \in \mathbf{right}(s)} (1 - x_{v(t,s)}) \right\}. \tag{B.37}$$

Note that in the above, the minimum will be equal to $x_{v(t,s)}$ for some s satisfying $\ell^* \in \mathbf{left}(s)$, or it will be equal to $1 - x_{v(t,s)}$ for some s satisfying $\ell^* \in \mathbf{right}(s)$. We now consider these two cases separately.

Case 1: $y_{t,\ell^*} = x_{v(t,s^*)}$ for some s^* for which $\ell^* \in \mathbf{left}(s^*)$. In this case, consider the following procedure for identifying another leaf ℓ':

Procedure 2.A:

1. Set $j \leftarrow \mathbf{root}(t)$.
2. If $j \in \mathbf{leaves}(t)$, terminate with $\ell' = j$. Otherwise, go to Step 3.
3. If $j = s^*$, set $j \leftarrow \mathbf{rightchild}(j)$, and return to Step 2. Otherwise, go to Step 4.
4. If $x_{v(t,j)} \geq 0.5$, then set $j \leftarrow \mathbf{leftchild}(j)$, and return to Step 2.

   Otherwise, set $j \leftarrow \mathbf{rightchild}(j)$, and return to Step 2.

Procedure 2.A will return a leaf ℓ' for which the following will be true after the termination of Algorithm 4:

$$y_{t,\ell'} = \min \left\{ \underbrace{\min_{s \neq s^*: \ell' \in \mathbf{left}(s^*)} x_{v(t,s)}}_{(a)}, \underbrace{\min_{s: \ell' \in \mathbf{right}(s^*)} (1 - x_{v(t,s)})}_{(b)}, \underbrace{1 - x_{v(t,s^*)}}_{(c)} \right\} \tag{B.38}$$

$$= 1 - x_{v(t,s^*)} \tag{B.39}$$

where the second equality follows because, by the definition of Procedure 1, we know that $x_{v(t,s)} \geq 0.5$ for s such that $\ell^* \in \mathbf{left}(s)$, and $x_{v(t,s)} < 0.5$ or equivalently, $1 - x_{v(t,s)} \geq 0.5$ for s such that $\ell^* \in \mathbf{right}(s)$. Thus, in the above, terms (a) and (b) will both be at least 0.5, while term (c) is strictly less than 0.5. For this reason, $y_{t,\ell'}$ must be equal to $1 - x_{v(t,s^*)}$.

Observe that $y_{t,\ell^*} = x_{v(t,s^*)}$ and $y_{t,\ell'} = 1 - x_{v(t,s^*)}$, which means that $y_{t,\ell^*} + y_{t,\ell'} = 1$. Thus, if Algorithm 4 had encountered leaf ℓ^* followed by leaf ℓ', then by the definition of Algorithm 4, a C event should have been triggered at ℓ', contradicting our assumption that a C event never occurs. Similarly, if ℓ' was encountered before ℓ^*, then a C event should have occurred at ℓ^*, again resulting in a contradiction.

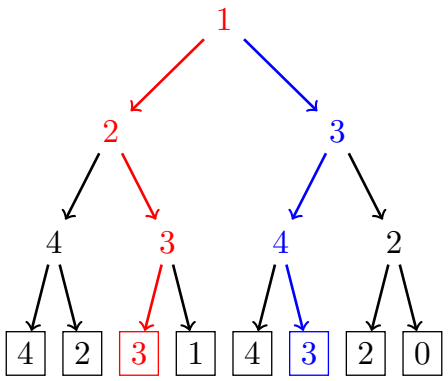


Figure B.4: Example of Procedure 1 and Procedure 2.A for a tree, where $\mathbf{x} = (x_1, x_2, x_3, x_4) = (0.6, 0.3, 0.7, 0.1)$.

Figure B.4 provides an example of Procedure 1 and Procedure 2.A applied to a tree. In this example, $\mathbf{x} = (x_1, x_2, x_3, x_4) = (0.6, 0.3, 0.7, 0.1)$. In this example, the sequence of red nodes is the path traversed by Procedure 1. This results in the leaf ℓ^*, which is the red leaf in the figure, whose value is $y_{t,\ell^*} = \min\{0.6, 1 - 0.3, 0.7\} = 0.6$. This value is exactly equal to $x_{v(t,s)}$ when s is equal to the root node, so we have that s^* is equal to the root node. We now apply Procedure 2.A, which traverses the sequence of nodes indicated in blue, and returns the blue leaf as ℓ', for which $y_{t,\ell'} = \min\{1 - 0.6, 0.7, 1 - 0.1\} = 0.4$.

Case 2: $y_{t,\ell^*} = 1 - x_{v(t,s^*)}$ for some s^* for which $\ell^* \in \textbf{right}(s^*)$. In this case, consider the following procedure that is similar to Procedure 2.A above:

Procedure 2.B:

1. Set $j \leftarrow \mathbf{root}(t)$.
2. If $j \in \mathbf{leaves}(t)$, terminate with $\ell' = j$. Otherwise, go to Step 3.
3. If $j = s^*$, set $j \leftarrow \mathbf{leftchild}(j)$, and return to Step 2. Otherwise, go to Step 4.
4. If $x_{v(t,j)} \geq 0.5$, then set $j \leftarrow \mathbf{leftchild}(j)$, and return to Step 2.

   Otherwise, set $j \leftarrow \mathbf{rightchild}(j)$, and return to Step 2.

In the same way as for Case 1, we can show that

$$y_{t,\ell'} = \min \left\{ \underbrace{\min_{s:\ell' \in \mathbf{left}(s^*)} x_{v(t,s)}}_{(a)}, \underbrace{\min_{s \neq s^*:\ell' \in \mathbf{right}(s^*)} (1 - x_{v(t,s)})}_{(b)}, \underbrace{x_{v(t,s^*)}}_{(c)} \right\}$$
$$= x_{v(t,s^*)}$$

Similarly to Case 1, we can again see that $y_{t,\ell^*} + y_{t,\ell'} = 1 - x_{v(t,s^*)} + x_{v(t,s^*)} = 1$, which implies that a C event must have occurred when either ℓ' or ℓ^* was checked. Thus, we again reach a contradiction.

These two cases establish that $\mathbf{y}_t$ must satisfy the unit sum constraint and therefore, that $\mathbf{y}_t$ is a feasible solution of problem (4.13).

Proof of part (b) (extreme point): To show that $\mathbf{y}_t$ is an extreme point, let us assume that $\mathbf{y}_t$ is not an extreme point. Then, there exist solutions $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$ different from $\mathbf{y}_t$ and a weight $\theta \in (0, 1)$ such that $\mathbf{y}_t = \theta \mathbf{y}_t^1 + (1 - \theta)\mathbf{y}_t^2$.

Let ℓ^* be the first leaf checked by Algorithm 4 at which $y_{t,\ell^*} \neq y_{t,\ell^*}^1$ and $y_{t,\ell^*} \neq y_{t,\ell^*}^2$. Such

a leaf must exist because $\mathbf{y}_t \neq \mathbf{y}_t^1$ and $\mathbf{y}_t \neq \mathbf{y}_t^2$, and because $\mathbf{y}_t$ is the convex combination of $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$. Without loss of generality, let us further assume that

$$y_{t,\ell^*}^1 < y_{t,\ell^*} < y_{t,\ell^*}^2.$$

By definition, Algorithm 4 sets each $y_{t,\ell}$ to the largest it can be without violating the left split constraints (4.4c) and the right split constraints (4.4d), and ensuring that $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}$ does not exceed 1. Since $y_{t,\ell^*}^2 > y_{t,\ell^*}$, and since $\mathbf{y}_t^2$ and $\mathbf{y}_t$ are equal for all leaves checked before ℓ^*, this implies that $\mathbf{y}_t^2$ either violates constraint (4.3c), violates constraint (4.3d), or is such that $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} > 1$. This implies that $\mathbf{y}_t^2$ cannot be a feasible solution, which contradicts the assumption that $\mathbf{y}_t^2$ is a feasible solution, and ultimately contradicts $\mathbf{y}_t$ not being an extreme point. □

B.2.7 Proof of Theorem 6

Proof of part (a) (feasibility): We first establish constraint (4.14c). First, observe that when for any split s and leaf ℓ such that $A_{s,\ell} \notin \mathcal{E}$, then $\alpha_{t,s,\ell} = 0$, which automatically satisfies the constraint. For any (s, ℓ) such that $A_{t,s,\ell} \in \mathcal{E}$, observe that it must be the case that the leaf $f(A_{s,\ell})$ that is checked when $A_{s,\ell}$ occurs was checked before the leaf $f(C)$ that is checked when the C event occurs; this is because Algorithm 4 terminates when the C event occurs. As a result, it must be that $r_{t,f(A_{s,\ell})} \geq r_{t,f(C)}$, since the leaves are checked in decreasing order of revenue. As a result, $\alpha_{t,s,\ell} = r_{t,f(A_{s,\ell})} - \gamma_t = r_{t,f(A_{s,\ell})} - r_{t,f(C)} \geq 0$, which establishes constraint (4.14c).

Constraint (4.14d) (that each $\beta_{t,s,\ell}$ is nonnegative) follows by similar reasoning as constraint (4.14c); for brevity, we omit the steps.

This leaves constraint (4.14b). Let ℓ be a leaf. There are three collectively exhaustive cases to consider: (1) ℓ is a leaf such that $A_{s,\ell} \in \mathcal{E}$ for some $s \in \mathbf{splits}(t)$; (2) ℓ is a leaf such that $B_{s,\ell} \in \mathcal{E}$ for some $s \in \mathbf{splits}(t)$; and (3) ℓ is a leaf such that $r_{t,\ell} \leq r_{t,f(C)}$. Note that these are collectively exhaustive because every leaf ℓ with $r_{t,\ell} > r_{t,f(C)}$ is a leaf that is

checked before the final leaf $f(C)$, and thus by the definition of Algorithm 4, either an $A_{s,\ell}$ or a $B_{s,\ell}$ event occurs for some split s.

In the first case, if ℓ is a leaf such that $A_{s,\ell} \in \mathcal{E}$ for some $s \in \mathbf{splits}(t)$, then let $s_{A,\ell}$ be that split. We then have

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t \geq \alpha_{t,s_{A,\ell},\ell} + \gamma_t = r_{t,\ell} - \gamma_t + \gamma_t = r_{t,\ell}$$

where the first step follows by the nonnegativity of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ and the fact that $s_{A,\ell} \in \mathbf{LS}(\ell)$; and the second step follows by how the dual procedure (Algorithm 5) sets $\alpha_{t,s,\ell}$ when $A_{s,\ell} \in \mathcal{E}$.

Similarly, in the second case, if ℓ is a leaf such that $B_{s,\ell} \in \mathcal{E}$ for some $s \in \mathbf{splits}(t)$, then letting $s_{B,\ell}$ be that split, we similarly have

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t \geq \beta_{t,s_{B,\ell},\ell} + \gamma_t = r_{t,\ell} - \gamma_t + \gamma_t = r_{t,\ell}.$$

Finally, in the third case, if ℓ is such that $r_{t,\ell} \leq r_{t,f(C)}$, then the nonnegativity of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ immediately gives us that

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t \geq \gamma_t = r_{t,f(C)} \geq r_{t,\ell}.$$

This establishes that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to problem (4.14).

Proof of part (b) (basic feasible solution): To establish this, we will use the equivalence between extreme points and basic feasible solutions. A feasible solution $\mathbf{z}$ in a polyhedron $P = \{\mathbf{z} \in \mathbb{R}^m \mid \mathbf{A}\mathbf{z} \leq \mathbf{b}\}$ is a basic feasible solution if there are m linearly independent active constraints at $\mathbf{z}$.

First, let us define the sets L_A and L_B as follows:

$$L_A = \{\ell \in \mathbf{leaves}(t) \mid A_{s,\ell} \in \mathcal{E} \text{ for some } s \in \mathbf{splits}(t)\},$$
$$L_B = \{\ell \in \mathbf{leaves}(t) \mid B_{s,\ell} \in \mathcal{E} \text{ for some } s \in \mathbf{splits}(t)\}.$$

Additionally, let us define $s_{A,\ell}$ to be the split for which an $A_{s,\ell}$ event occurs for leaf $\ell \in L_A$, and $s_{B,\ell}$ to be the split for which an $B_{s,\ell}$ event occurs for leaf $\ell \in L_B$. We note that for each leaf ℓ, by the definition of Algorithm 4, there is at most one event of the form $A_{s,\ell}$, $B_{s,\ell}$ or C that can occur. Thus, an immediate identity is

$$|L_A| + |L_B| + 1 = |\mathcal{E}|,$$

where on the left hand side, the first term counts the number of $A_{s,\ell}$ events, the second counts the number of $B_{s,\ell}$ events that have occurred, and the 1 corresponds to the single C event that must occur.

Now, consider the following system of equations:

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t = r_{t,\ell}, \quad \forall \ell \in L_A, \tag{B.40}$$

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t = r_{t,\ell}, \quad \forall \ell \in L_B, \tag{B.41}$$

$$\sum_{s \in \mathbf{LS}(f(C))} \alpha_{t,s,f(C)} + \sum_{s \in \mathbf{RS}(f(C))} \beta_{t,s,f(C)} + \gamma_t = r_{t,f(C)}, \tag{B.42}$$

$$\alpha_{t,s,\ell} = 0, \quad \forall s \in \mathbf{splits}(t), \ell \in \mathbf{left}(s) \text{ such that } A_{s,\ell} \notin \mathcal{E}, \tag{B.43}$$

$$\beta_{t,s,\ell} = 0, \quad \forall s \in \mathbf{splits}(t), \ell \in \mathbf{right}(t) \text{ such that } B_{s,\ell} \notin \mathcal{E}. \tag{B.44}$$

Observe that each equation corresponds to one of the constraints of problem (4.14) being made to hold at equality. Observe that there are in total

$$\sum_{s \in \mathbf{splits}(t)} |\mathbf{left}(s)| + \sum_{s \in \mathbf{splits}(t)} |\mathbf{right}(s)| + 1$$

variables. There are

$$\begin{aligned} &|L_A| + |L_B| + 1 + \left[\sum_{s \in \mathbf{splits}(t)} |\mathbf{left}(s)| - |L_A|\right] + \left[\sum_{s \in \mathbf{splits}(t)} |\mathbf{right}(s)| - |L_B|\right] \\ &= \sum_{s \in \mathbf{splits}(t)} |\mathbf{left}(s)| + \sum_{s \in \mathbf{splits}(t)} |\mathbf{right}(s)| + 1 \end{aligned}$$

equations. We now show that the only solution to this system of equations is exactly the solution produced by Algorithm 5.

First, observe that by the property that at most one event of the form $A_{s,\ell}$, $B_{s,\ell}$ or C can occur for the leaf $\ell = f(C)$, equations (B.42) - (B.44) imply that

$$\sum_{s \in \mathbf{LS}(f(C))} \alpha_{t,s,f(C)} + \sum_{s \in \mathbf{RS}(f(C))} \beta_{t,s,f(C)} + \gamma_t = 0 + 0 + \gamma_t = \gamma_t = r_{t,f(C)},$$

which is exactly how Algorithm 5 sets γ_t.

Second, observe again that by the property that at most one event of the form $A_{s,\ell}$, $B_{s,\ell}$ or C can occur for a leaf $\ell \in L_A$, equations (B.40), (B.43) and (B.44) imply that for any leaf $\ell \in L_A$,

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t = \alpha_{t,s_{A,\ell},\ell} + \gamma_t = r_{t,\ell},$$

or equivalently, that $\alpha_{t,s_{A,\ell},\ell} = r_{t,\ell} - \gamma_t$, which is exactly how Algorithm 5 sets $\alpha_{t,s_{A,\ell},\ell}$.

Similarly, for any leaf $\ell \in L_B$, equations (B.41), (B.43) and (B.44) imply

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t = \beta_{t,s_{B,\ell},\ell} + \gamma_t = r_{t,\ell},$$

or equivalently, $\beta_{t,s_{B,\ell},\ell} = r_{t,\ell} - \gamma_t$, which again exactly agrees with Algorithm 5.

Finally, for any s, ℓ such that $A_{s,\ell} \notin \mathcal{E}$, equation (B.43) exactly matches how Algorithm 5 sets $\alpha_{t,s,\ell}$ for such (s,ℓ) combinations. Similarly, equation (B.44) exactly matches how Algorithm 5 sets $\beta_{t,s,\ell}$ for (s,ℓ) pairs for which $B_{s,\ell} \notin \mathcal{E}$.

Since the solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ that is completely determined by equations (B.40) - (B.44) is identical to the one produced by Algorithm (5), it follows that this solution is an extreme point. □

B.2.8 Proof of Theorem 7

We will prove this by showing that the two solutions $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ satisfy complementary slackness. The complementary slackness conditions for this problem are

$$(x_{v(t,s)} - y_{t,\ell}) \cdot \alpha_{t,s,\ell} = 0, \quad \forall\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{left}(s), \tag{B.45}$$

$$(1 - x_{v(t,s)} - y_{t,\ell}) \cdot \beta_{t,s,\ell} = 0, \quad \forall\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{right}(s), \tag{B.46}$$

$$y_{t,\ell} \cdot \Big(\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t - r_{t,\ell} \Big) = 0, \quad \forall\ \ell \in \mathbf{leaves}(t). \tag{B.47}$$

We now verify each of these conditions.

Condition (B.45)**:** For this condition, observe that if $A_{s,\ell} \notin \mathcal{E}$, then by the definition of Algorithm 5, $\alpha_{t,s,\ell}$ will be equal to the default value of zero, and the condition will automatically hold. If $A_{s,\ell} \in \mathcal{E}$, then by the definition of the primal procedure (Algorithm 4) $y_{t,\ell}$ will be set to q^* which is equal to $x_{v(t,s)}$. Thus, we will have that $x_{v(t,s)} - y_{t,\ell} = 0$ and the condition is again satisfied.

Condition (B.46)**:** This condition follows by analogous reasoning as condition (B.45).

Condition (B.47)**:** For this condition, we can see that if $y_{t,\ell} = 0$, then the condition is immediately satisfied; thus, let us assume that $y_{t,\ell} > 0$. In this case, it must be that when ℓ was checked by Algorithm 4, that either an $A_{s,\ell}$ event occurred for some split s, a $B_{s,\ell}$ event occurred for some split s or a C event occurred. As shown in the proof of Theorem B.2.7 (see equations (B.40) - (B.42)), for any leaf for which such an event occurs, the constraint (4.14b) is satisfied at equality and thus condition (B.47) must hold as well.

Since all three conditions hold, it follows that the solutions $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ produced by Algorithms 4 and 5 are optimal for their respective problems. □

B.2.9 Proof of Theorem 19

We prove that LeafMIO closed form for integer $\mathbf{x}$.

Proof of part (a) (primal feasibility): Observe that by construction, $\mathbf{y}_t$ automatically satisfies the nonnegativity constraint (4.13e) and the unit sum constraint (4.13b). For constraints (4.13c) and (4.13d), observe that for any $\ell \neq \ell^*$, these constraints are automatically satisfied because $y_{t,\ell} = 0$ whereas $x_{v(t,s)}$ and $1 - x_{v(t,s)}$ can only be either 0 or 1. For the case that $\ell = \ell^*$, observe that if $\ell^* \in \mathbf{left}(s)$ for some split s, then it must be that $x_{v(t,s)} = 1$, so the constraint $y_{t,\ell^*} \leq x_{v(t,s)}$ is satisfied. Similarly, if $\ell^* \in \mathbf{right}(s)$ for some split s, then it must be that $x_{v(t,s)} = 0$, so the constraint $y_{t,\ell^*} \leq 1 - x_{v(t,s)}$ is satisfied. Thus, $\mathbf{y}_t$ is a feasible solution of problem (4.13).

Proof of part (b) (dual feasibility): First, observe that by the definition of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$, they are automatically nonnegative, and so constraints (4.14c) and (4.14d) are satisfied. To verify constraint (4.14b), observe that for any $\ell \neq \ell^*$, it is either the case that $\ell \in \mathbf{right}(s')$ for some $s' \in \mathbf{LS}(\ell^*)$, or $\ell \in \mathbf{left}(s')$ for some $s' \in \mathbf{RS}(\ell^*)$. In the first case, we have:

$$\sum_{s:\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} + \sum_{s:\ell \in \mathbf{right}(s)} \beta_{t,s,\ell} + \gamma_t \geq \beta_{t,s,\ell} + \gamma_t \geq r_{t,\ell} - r_{t,\ell^*} + r_{t,\ell^*} = r_{t,\ell}$$

where the first inequality follows by the nonnegativity of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ and the fact that $\ell \in \mathbf{right}(s')$, and the second inequality follows by the definition of $\boldsymbol{\beta}_t$ and the fact that $s' \in \mathbf{LS}(\ell^*)$. In the second case, where $\ell \in \mathbf{left}(s')$ for some $s' \in \mathbf{RS}(\ell^*)$, similar logic lets us establish that

$$\sum_{s:\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} + \sum_{s:\ell \in \mathbf{right}(s)} \beta_{t,s,\ell} + \gamma_t \geq \alpha_{t,s',\ell} + \gamma_t \geq r_{t,\ell} - r_{t,\ell^*} + r_{t,\ell^*} = r_{t,\ell}.$$

This establishes that constraint (4.14b) holds for any leaf ℓ other than ℓ^*. When $\ell = \ell^*$, we

very simply have

$$\sum_{s:\ell^* \in \mathbf{left}(s)} \alpha_{t,s,\ell^*} + \sum_{s:\ell^* \in \mathbf{right}(s)} \beta_{t,s,\ell} + \gamma_t = 0 + 0 + \gamma_t = r_{t,\ell^*},$$

where the first step follows from the definition of $\alpha_{t,s,\ell}$ and $\beta_{t,s,\ell}$ (note that from the statement of Theorem 19, α_{t,s,ℓ^*} will be zero for any $s \in \mathbf{LS}(\ell^*)$, or equivalently, any s such that $\ell^* \in \mathbf{left}(s)$, and similarly, β_{t,s,ℓ^*} will be zero for any $s \in \mathbf{RS}(\ell^*)$). Thus, this establishes that constraint (4.14b) holds for every leaf ℓ, and thus, that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution of the dual subproblem (4.14).

Proof of part (c) (optimality): To establish optimality, we simply need to check that the primal and dual solutions obtain the same objective; by weak duality, we will thus establish that the two solutions are optimal for their respective problems. For the primal solution $\mathbf{y}_t$, it is immediately clear that its objective is r_{t,ℓ^*}. For the dual solution, we have

$$\begin{aligned}
&\sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} x_{v(t,s)} + \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{right}(s)} \beta_{t,s,\ell}(1 - x_{v(t,s)}) + \gamma_t \\
&= \sum_{s \in \mathbf{RS}(\ell^*)} \sum_{\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} x_{v(t,s)} + \sum_{s \in \mathbf{RS}(\ell^*)} \sum_{\ell \in \mathbf{right}(s)} \beta_{t,s,\ell}(1 - x_{v(t,s)}) + \gamma_t \\
&= 0 + 0 + \gamma_t \\
&= r_{t,\ell^*},
\end{aligned}$$

where the first step follows because $\alpha_{t,s,\ell}$ is automatically zero for any $s \notin \mathbf{RS}(\ell^*)$ and $\beta_{t,s,\ell}$ is automatically zero for any $s \notin \mathbf{LS}(\ell^*)$; the second step follows because for any $s \in \mathbf{RS}(\ell^*)$, $x_{v(t,s)}$ will be 0 (recall that $x_{v(t,s)} = 0$ means that the product is not in the assortment, which implies that we must proceed to the right of any split $s \in \mathbf{RS}(\ell^*)$), and similarly, for any $s \in \mathbf{LS}(\ell^*)$, $x_{v(t,s)}$ will be 1; and the final step follows by the definition of γ_t. This establishes that $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for problems (4.13) and (4.14) respectively. □

B.2.10 Proof of Theorem 8

Proof of part (a) (feasibility): By definition, the solution produced by Algorithm 6 produces a solution $\mathbf{y}_t$ that satisfies the left and right split constraints (4.17c) and (4.17d). With regard to the nonnegativity constraint (4.17e), we can see that at each stage of Algorithm 6, the quantities $x_{v(t,s)} - \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell}$, $1 - x_{v(t,s)} - \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell}$ and $1 - \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}$ never become negative; thus, the solution $\mathbf{y}_t$ produced upon termination satisfies the nonnegativity constraint (4.17e).

The only constraint that remains to be verified is constraint (4.17b), which requires that $\mathbf{y}_t$ adds up to 1. Observe that it is sufficient for a C event to occur during the execution of Algorithm 6 to ensure that constraint (4.17b) is satisfied. We will show that a C event must occur during the execution of Algorithm 6.

We proceed by contradiction. For the sake of a contradiction, let us suppose that a C event does not occur during the execution of the algorithm. Note that under this assumption, for any split s, it is impossible that the solution $\mathbf{y}_t$ produced by Algorithm 6 satisfies

$$x_{v(t,s)} = \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell},$$
$$1 - x_{v(t,s)} = \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell},$$

as this would imply that

$$\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} + \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} = x_{v(t,s)} + 1 - x_{v(t,s)} = 1;$$

by the definition of Algorithm 6, this would have triggered a C event at one of the leaves in $\mathbf{left}(s) \cup \mathbf{right}(s)$.

Thus, this means that at every split, either $\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} < x_{v(t,s)}$ or $\sum_{\ell \in \mathbf{right}(s)} < 1 - x_{v(t,s)}$. Using this property, let us identify a leaf ℓ^* using the following procedure:

Procedure 3:

1. Set $j \leftarrow \mathbf{root}(t)$.
2. If $j \in \mathbf{leaves}(t)$, terminate with $\ell^* = j$.

   Otherwise, proceed to Step 3.
3. If $\sum_{\ell \in \mathbf{left}(j)} y_{t,\ell} < x_{v(t,j)}$, set $j \leftarrow \mathbf{leftchild}(j)$.

   Otherwise, set $j \leftarrow \mathbf{rightchild}(j)$.
4. Repeat Step 2.

Note that by our observation that at most one of the left or right split constraints can be satisfied at equality for any split s, Procedure 3 above is guaranteed to terminate with a leaf ℓ^* such that:

$$y_{t,\ell^*} \leq \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} < x_{v(t,s)}, \quad \forall\ s \in \mathbf{splits}(t) \text{ such that } \ell^* \in \mathbf{left}(s),$$

$$y_{t,\ell^*} \leq \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} < 1 - x_{v(t,s)}, \quad \forall\ s \in \mathbf{splits}(t) \text{ such that } \ell^* \in \mathbf{right}(s).$$

However, this is impossible, because Algorithm 6 always sets each leaf $y_{t,\ell}$ to the highest value it can be without violating any of the left or right split constraints; the above conditions imply that y_{t,ℓ^*} could have been set higher, which is not possible. We thus have a contradiction, and it must be the case that a C event occurs.

Proof of part (b) (extreme point): To show that $\mathbf{y}_t$ is an extreme point, let us assume that $\mathbf{y}_t$ is not an extreme point. Then, there exist feasible solutions $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$ different from $\mathbf{y}_t$ and a weight $\theta \in (0, 1)$ such that $\mathbf{y}_t = \theta \mathbf{y}_t^1 + (1 - \theta)\mathbf{y}_t^2$.

Let ℓ^* be the first leaf checked by Algorithm 6 at which $y_{t,\ell^*} \neq y_{t,\ell^*}^1$ and $y_{t,\ell^*} \neq y_{t,\ell^*}^2$. Such

a leaf must exist because $\mathbf{y}_t \neq \mathbf{y}_t^1$ and $\mathbf{y}_t \neq \mathbf{y}_t^2$, and because $\mathbf{y}_t$ is the convex combination of $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$. Without loss of generality, let us further assume that

$$y_{t,\ell^*}^1 < y_{t,\ell^*} < y_{t,\ell^*}^2.$$

By definition, Algorithm 6 sets $y_{t,\ell}$ at each iteration to the largest it can be without violating the left split constraints (4.17c) and the right split constraints (4.17d), and ensuring that $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}$ does not exceed 1. Since $y_{t,\ell^*}^2 > y_{t,\ell^*}$, and since $\mathbf{y}_t^2$ and $\mathbf{y}_t$ are equal for all leaves checked before ℓ^*, this implies that $\mathbf{y}_t^2$ either violates constraint (4.17c), violates constraint (4.17d), or is such that $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} > 1$. This implies that $\mathbf{y}_t^2$ cannot be a feasible solution, which contradicts the assumption that $\mathbf{y}_t^2$ is a feasible solution. □

B.2.11 Proof of Theorem 9

We prove that SplitMIO dual is BFS.

Proof of part (a) (feasibility): Before we prove the result, we first establish a helpful property of the events that are triggered during the execution of Algorithm 6.

Lemma 9 *Let $s_1, s_2 \in \mathbf{splits}(t)$, $s_1 \neq s_2$, such that s_2 is a descendant of s_1. Suppose that $e_1 = A_{s_1}$ or $e_1 = B_{s_1}$, and that $e_2 = A_{s_2}$ or $e_2 = B_{s_2}$. If e_1 and e_2 occur during the execution of Algorithm 6, then $r_{t,f(e_1)} \leq r_{t,f(e_2)}$.*

Proof: We will prove this by contradiction. Suppose that we have two splits s_1 and s_2 and events e_1 and e_2 as in the statement of the lemma, and that $r_{t,f(e_2)} < r_{t,f(e_1)}$. This implies that leaf $f(e_1)$ is checked before leaf $f(e_2)$. When leaf $f(e_1)$ is checked, the event e_1 occurs, which implies that either the left split constraint (4.17c) becomes tight (if $e_1 = A_{s_1}$) or the right split constraint (4.17d) becomes tight (if $e_1 = B_{s_1}$) at s_1. In either case, since s_2 is a descendant of s_1, the leaf $f(e_2)$ must be contained in the left leaves of split s_1 (if $e = A_{s_1}$) or the right leaves of split s_1 (if $e_1 = B_{s_1}$). Thus, when leaf $f(e_2)$ is checked, the event e_2

cannot occur, because q_{s_1} in Algorithm 6 will be zero (implying that $q_{A,B} = 0$), and so s^* cannot be equal to s_2 because s_1 is a shallower split that attains the minimum of $q_{A,B} = 0$. $\square$

To establish that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is feasible for the SplitMIO dual subproblem (4.18), we will first show that the $\alpha_{t,s}$ variables are nonnegative.

Fix $s \in \mathbf{splits}(t)$. If $A_s \notin \mathcal{E}$, then $\alpha_{t,s} = 0$, and constraint (4.18c) is satisfied. If $A_s \in \mathcal{E}$, then consider the split $\tilde{s}$ defined as follows: let Set $1 = \{s' \mid \mathbf{LS}(f(A_s)),\ d(s') < d,\ A_{s'} \in \mathcal{E}\}$ and Set $2 = \{s' \mid \mathbf{RS}(f(A_s)),\ d(s') < d,\ B_{s'} \in \mathcal{E}\}$, and

$$\tilde{s} = \arg\min_{s'} \{d(s') \mid s' \in \text{Set } 1 \cup \text{Set } 2\}$$

where we recall that $d = d(s)$ is the depth of split s. In words, $\tilde{s}$ is the shallowest split (i.e., closest to the root) along the path of splits from the root node to split s such that either an $A_{\tilde{s}}$ event occurs or a $B_{\tilde{s}}$ event occurs for split $\tilde{s}$. There are three possible cases that can occur here, which we now handle.

Case 1: $\tilde{s} \in \mathbf{LS}(f(A_s))$. In this case, $A_{\tilde{s}} \in \mathcal{E}$, and we have

$$\begin{aligned}
\alpha_{t,s} &= r_{t,f(A_s)} - \left[\sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ d(s') < d, \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_s)): \\ d(s') < d, \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - \left[\alpha_{t,\tilde{s}} + \sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ d(s') < d(\tilde{s}), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_s)): \\ d(s') < d(\tilde{s}), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - \left[\alpha_{t,\tilde{s}} + \sum_{\substack{s' \in \mathbf{LS}(f(A_{\tilde{s}})): \\ d(s') < d(\tilde{s}), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_{\tilde{s}})): \\ d(s') < d(\tilde{s}), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - r_{t,f(A_{\tilde{s}})} \\
&\geq 0,
\end{aligned}$$

where the first step follows by the definition of $\alpha_{t,s}$ in Algorithm 7; the second step follows by the definition of $\alpha_{t,\tilde{s}}$ as the deepest split for which an A or B event occurs that is at a depth lower than s; the third step by the fact that the left splits and right splits of $f(A_{\tilde{s}})$ at a depth below $d(\tilde{s})$ are the same as the left and right splits of $f(A_s)$ at a depth below $d(\tilde{s})$; and the fourth step follows from the definition of $\alpha_{t,\tilde{s}}$ in Algorithm 7. The inequality follows by Lemma 9.

Case 2: $\tilde{s} \in \mathbf{RS}(f(A_s))$. In this case, $B_{\tilde{s}} \in \mathcal{E}$, and analogously to Case 1, we have:

$$\begin{aligned}
\alpha_{t,s} &= r_{t,f(A_s)} - \left[\sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ d(s') < d, \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_s)): \\ d(s') < d, B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - \left[\beta_{t,\tilde{s}} + \sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ d(s') < d(\tilde{s}), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_s)): \\ d(s') < d(\tilde{s}), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - \left[\beta_{t,\tilde{s}} + \sum_{\substack{s' \in \mathbf{LS}(f(B_{\tilde{s}})): \\ d(s') < d(\tilde{s}), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(B_{\tilde{s}})): \\ d(s') < d(\tilde{s}), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - r_{t,f(B_{\tilde{s}})} \\
&\geq 0.
\end{aligned}$$

Case 3: $\tilde{s}$ is undefined because the underlying sets are empty. In this case, $\alpha_{t,s} = r_{t,f(A_s)} - \gamma_t$. In this case, we have

$$\alpha_{t,s} = r_{t,f(A_s)} - \gamma_t = r_{t,f(A_s)} - r_{t,f(C)} \geq 0,$$

where the inequality follows because $f(C)$ is the last leaf to be checked before Algorithm 6 terminates, and thus it must be that $r_{t,f(A_s)} \geq r_{t,f(C)}$.

This establishes that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ satisfy constraint (4.18c). Constraint (4.18d) can be shown in an almost identical fashion; for brevity, we omit the steps.

We thus only need to verify constraint (4.18b). Let $\ell \in \mathbf{leaves}(t)$. Here, there are four

mutually exclusive and collectively exhaustive cases to consider.

Case 1: $r_{t,\ell} \leq r_{t,f(C)}$. In this case we have

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t \geq \gamma_t = r_{t,f(C)} \geq r_{t,\ell}.$$

Case 2: $r_{t,\ell} > r_{t,f(C)}$ and $\ell = f(A_s)$ for some $s \in \mathbf{splits}(t)$. In this case, we have

$$\sum_{s' \in \mathbf{LS}(\ell)} \alpha_{t,s'} + \sum_{s' \in \mathbf{RS}(\ell)} \beta_{t,s'} + \gamma_t \geq \alpha_{t,s} + \sum_{\substack{s' \in \mathbf{LS}(\ell): \\ d(s') < d(s), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(\ell): \\ d(s') < d(s), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t = r_{t,f(A_s)} = r_{t,\ell},$$

where the first step follows by the nonnegativity of $\alpha_{t,s'}$ and $\beta_{t,s'}$ for all s', and the second step by the definition of $\alpha_{t,s}$ in Algorithm 7.

Case 3: $r_{t,\ell} > r_{t,f(C)}$ and $\ell = f(B_s)$ for some $s \in \mathbf{splits}(t)$. By similar logic as case 2, we have

$$\sum_{s' \in \mathbf{LS}(\ell)} \alpha_{t,s'} + \sum_{s' \in \mathbf{RS}(\ell)} \beta_{t,s'} + \gamma_t \geq \beta_{t,s} + \sum_{\substack{s' \in \mathbf{LS}(\ell): \\ d(s') < d(s), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(\ell): \\ d(s') < d(s), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t = r_{t,f(B_s)} = r_{t,\ell}.$$

Case 4: $r_{t,\ell} > r_{t,f(C)}$ and ℓ is not equal to $f(A_s)$ or $f(B_s)$ for any split s. In this case, when leaf ℓ is checked by Algorithm 6, the algorithm reaches line 17 where s^* is determined and e is set to either A_{s^*} or B_{s^*}, and it turns out that e is already in $\mathcal{E}$. If $e = A_{s^*}$, then this

means that leaf $f(A_{s^*})$ was checked before leaf ℓ, and that $r_{t,\ell} \leq r_{t,f(A_{s^*})}$. We thus have

$$\begin{aligned}
&\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t \\
&\geq \alpha_{t,s^*} + \sum_{\substack{s \in \mathbf{LS}(\ell): \\ d(s) < d(s^*), \\ A_s \in \mathcal{E}}} \alpha_{t,s} + \sum_{\substack{s \in \mathbf{RS}(\ell): \\ d(s) < d(s^*), \\ B_s \in \mathcal{E}}} \beta_{t,s} + \gamma_t \\
&= \alpha_{t,s^*} + \sum_{\substack{s \in \mathbf{LS}(f(A_{s^*})): \\ d(s) < d(s^*), \\ A_s \in \mathcal{E}}} \alpha_{t,s} + \sum_{\substack{s \in \mathbf{RS}(f(A_{s^*})): \\ d(s) < d(s^*), \\ B_s \in \mathcal{E}}} \beta_{t,s} + \gamma_t \\
&= r_{t,f(A_{s^*})} \\
&\geq r_{t,\ell},
\end{aligned}$$

where the first equality follows because ℓ and $f(A_{s^*})$, by virtue of being to the left of s^*, share the same left and right splits at depths lower than $d(s^*)$. Similarly, if $e = B_{s^*}$, then the leaf $f(B_{s^*})$ was checked before ℓ, which means that $r_{t,\ell} \leq r_{t,f(B_{s^*})}$; in this case, we have

$$\begin{aligned}
&\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t \\
&\geq \beta_{t,s^*} + \sum_{\substack{s \in \mathbf{LS}(\ell): \\ d(s) < d(s^*), \\ A_s \in \mathcal{E}}} \alpha_{t,s} + \sum_{\substack{s \in \mathbf{RS}(\ell): \\ d(s) < d(s^*), \\ B_s \in \mathcal{E}}} \beta_{t,s} + \gamma_t \\
&\geq \beta_{t,s^*} + \sum_{\substack{s \in \mathbf{LS}(f(B_{s^*})): \\ d(s) < d(s^*), \\ A_s \in \mathcal{E}}} \alpha_{t,s} + \sum_{\substack{s \in \mathbf{RS}(f(B_{s^*})): \\ d(s) < d(s^*), \\ B_s \in \mathcal{E}}} \beta_{t,s} + \gamma_t \\
&= r_{t,f(B_{s^*})} \\
&\geq r_{t,\ell}.
\end{aligned}$$

We have thus shown that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to the SplitMIO dual subproblem (4.18).

Proof of part (b) (extreme point): To prove this, we will use the equivalence between

extreme points and basic feasible solutions, and show that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a basic feasible solution of problem (4.18).

Define the sets L_A and L_B as

$$L_A = \{\ell \in \mathbf{leaves}(t) \mid \ell = f(A_s) \text{ for some } s \in \mathbf{splits}(t)\},$$
$$L_B = \{\ell \in \mathbf{leaves}(t) \mid \ell = f(B_s) \text{ for some } s \in \mathbf{splits}(t)\}.$$

Consider the following system of equations:

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t = r_{t,\ell}, \quad \forall \ell \in L_A, \tag{B.48}$$

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t = r_{t,\ell}, \quad \forall \ell \in L_B, \tag{B.49}$$

$$\sum_{s \in \mathbf{LS}(f(C))} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(f(C))} \beta_{t,s} + \gamma_t = r_{t,f(C)}, \tag{B.50}$$

$$\alpha_{t,s} = 0, \quad \forall\ s \text{ such that } A_s \notin \mathcal{E}, \tag{B.51}$$

$$\beta_{t,s} = 0, \quad \forall\ s \text{ such that } B_s \notin \mathcal{E}. \tag{B.52}$$

Observe that each equation corresponds to a constraint from problem (4.18) made to hold at equality. In addition, we note that there are $|L_A| + |L_B| + 1 + (|\mathbf{splits}(t)| - |L_A|) + (|\mathbf{splits}(t)| - |L_B|) = 2|\mathbf{splits}(t)| + 1$ equations, which is exactly the number of variables. We will show that the unique solution implied by this system of equations is exactly the solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ that is produced by Algorithm 7.

In order to establish this, we first establish a couple of useful results.

Lemma 10 *Suppose that $e \in \mathcal{E}$, $\ell = f(e)$ and $e = A_s$ or $e = B_s$ for some $s \in \mathbf{splits}(t)$. Then:*

a) $A_{s'} \notin \mathcal{E}$ for all $s' \in \mathbf{LS}(\ell)$ such that $d(s') > d(s)$; and

b) $B_{s'} \notin \mathcal{E}$ for all $s' \in \mathbf{RS}(\ell)$ such that $d(s') > d(s)$.

Proof of Lemma 10: We will prove this by contradiction. Without loss of generality, let us suppose that there exists an $A_{s'}$ event in $\mathcal{E}$ where $s' \in \mathbf{LS}(\ell)$ and $d(s') > d(s)$. (The case where there exists an $B_{s'}$ event in $\mathcal{E}$ where $s' \in \mathbf{RS}(\ell)$ and $d(s') > d(s)$ can be shown almost identically.)

Since $A_{s'} \in \mathcal{E}$, consider the leaf $\ell' = f(A_{s'})$. There are now two possibilities for when Algorithm 6 checks leaf ℓ':

1. **Case 1**: Leaf ℓ' is checked after leaf ℓ. In this case, in the iteration of Algorithm 6 corresponding to leaf ℓ', it will be the case that $q_s = 0$ because the left constraint (4.17c) at split s (if $e = A_s$) or the right constraint (4.17d) at split s (if $e = B_s$) became tight when leaf ℓ was checked. As a result, $q_{A,B} = 0$ in the iteration for leaf ℓ'. This implies that s' cannot be the lowest depth split that attains the minimum q_s value of $q_{A,B}$, because $q_s = 0$, and s has a depth lower than s', which contradicts the fact that the $A_{s'}$ event occurred.

2. **Case 2**: Leaf ℓ' is checked before leaf ℓ. In this case, consider the value of q_s when leaf ℓ is checked in Algorithm 6.

   If $q_s > 0$, then there is immediately a contradiction, because $q_{s'} = 0$ when leaf ℓ is checked (this is true because the left split constraint (4.17c) at s' became tight after leaf ℓ' was checked), and thus it is impossible that $s^* = s$.

   If $q_s = 0$, then this implies that $x_{v(t,s)} = 0$. This would imply that $q_s = 0$ when leaf ℓ' was checked, which would imply that s^* cannot be s' when leaf ℓ' is checked because s is at a lower depth than s'.

Thus, in either case, we arrive at a contradiction, which completes the proof. □

Lemma 11 *Suppose that $\ell = f(C)$. Then:*

a) $A_{s'} \notin \mathcal{E}$ *for all* $s' \in \mathbf{LS}(\ell)$*; and*

b) $B_{s'} \notin \mathcal{E}$ *for all* $s' \in \mathbf{RS}(\ell)$.

Proof of Lemma 11: We proceed by contradiction. Suppose that A_s occurs for some $s \in \mathbf{LS}(\ell)$ or that B_s occurs for some $s \in \mathbf{LS}(\ell)$; in the former case, let $e = A_s$, and in the latter case, let $e = B_s$. Let $\ell' = f(e)$. Then ℓ' must be checked before ℓ by Algorithm 6, since the algorithm always terminates after a C event occurs. Consider what happens when Algorithm 6 checks leaf ℓ:

1. **Case 1**: $q_C > 0$. This is impossible, because if e occurs, then q_s when leaf ℓ is checked would have to be 0, which would imply that $q_{A,B} < q_C$ and that a C event could not have occurred when ℓ was checked.

2. **Case 2**: $q_C = 0$. This is also impossible, because it implies that the unit sum constraint (4.17b) was satisfied at an earlier iteration, which would have triggered the C event at a leaf that was checked before ℓ.

We thus have that $A_{s'}$ does not occur for any $s' \in \mathbf{LS}(\ell)$ and $B_{s'}$ does not occur for any $s' \in \mathbf{RS}(\ell)$, as required. □

With these two lemmas in hand, we now return to the proof of Theorem B.2.11 (b). Observe now that by using Lemmas 10 and 11 and using equations (B.56) and (B.57), the

system of equations (B.48)-(B.52) is equivalent to

$$\alpha_{t,s} + \sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ d(s') < d(s), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(\ell): \\ d(s') < d(s), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t = r_{t,f(A_s)}, \quad \forall s \text{ such that } A_s \in \mathcal{E}, \tag{B.53}$$

$$\beta_{t,s} + \sum_{\substack{s' \in \mathbf{LS}(\ell): \\ d(s') < d(s), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(\ell): \\ d(s') < d(s), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t = r_{t,f(B_s)}, \quad \forall s \text{ such that } B_s \in \mathcal{E}, \tag{B.54}$$

$$\gamma_t = r_{t,f(C)}, \tag{B.55}$$

$$\alpha_{t,s} = 0, \quad \forall\ s \text{ such that } A_s \notin \mathcal{E}, \tag{B.56}$$

$$\beta_{t,s} = 0, \quad \forall\ s \text{ such that } B_s \notin \mathcal{E}.. \tag{B.57}$$

We now observe that the solution implied by this system of equations is exactly the solution produced by Algorithm 7. We thus establish that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a basic feasible solution of problem (4.18), and thus an extreme point. □

B.2.12 Proof of Theorem 10

To prove that the $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ produced by Algorithms 6 and 7 are optimal for their respective problems, we show that they satisfy complementary slackness. The complementary slackness conditions for problems (4.17) and (4.18) are

$$\alpha_{t,s} \cdot \left(x_{v(t,s)} - \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \right) = 0, \quad \forall\ s \in \mathbf{splits}(t), \tag{B.58}$$

$$\beta_{t,s} \cdot \left(1 - x_{v(t,s)} - \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \right) = 0, \quad \forall\ s \in \mathbf{splits}(t), \tag{B.59}$$

$$y_{t,\ell} \cdot \left(\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t - r_{t,\ell} \right) = 0, \quad \forall\ \ell \in \mathbf{leaves}(t). \tag{B.60}$$

Condition (B.58): If $\alpha_{t,s} = 0$, then the condition is trivially satisfied. If $\alpha_{t,s} > 0$, then this implies that $A_s \in \mathcal{E}$. This means that the left split constraint (4.17c) at s became tight

after leaves $f(A_s)$ was checked, which implies that $\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} = x_{v(t,s)}$ or equivalently, that $x_{v(t,s)} - \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} = 0$, which again implies that the condition is satisfied.

Condition (B.59): This follows along similar logic to condition (B.58), only that we use the fact that $\beta_{t,s} > 0$ implies that a B_s event occurred and that the right split constraint (4.17d) at s became tight.

Condition (B.60): If $y_{t,\ell} = 0$, then the condition is trivially satisfied. If $y_{t,\ell} > 0$, then either $\ell = f(C)$, or $\ell = f(A_s)$ for some split $s \in \mathbf{LS}(\ell)$, or $\ell = f(B_s)$ for some split $s \in \mathbf{RS}(\ell)$. In any of these three cases, as shown in the proof of part (b) of Theorem 9, the dual constraint (4.18b) holds with equality for any such leaf ℓ. Thus, we have that $\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t - r_{t,\ell} = 0$, and the condition is again satisfied.

Since complementary slackness holds, $\mathbf{y}_t$ is feasible for the primal problem (4.17) (by Theorem B.2.10), and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is feasible for the dual problem (4.18) (by Theorem 9, it follows that $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for their respective problems. □

B.2.13 Proof of Theorem 20

We prove that SplitMIO primal and dual are closed form solvable for binary $\mathbf{x}$

Proof of part (a): Observe that by construction, $\mathbf{y}_t$ automatically satisfies the unit sum constraint (4.17b) and the nonnegativity constraint (4.17e). We thus need to verify constraints (4.17c) and (4.17d).

For constraint (4.17c), observe that for any split $s \notin \mathbf{LS}(\ell^*)$, it must be that $\ell^* \notin \mathbf{left}(s)$. Thus, we will have

$$\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} = 0,$$

which means that constraint (4.17c) is automatically satisfied, because the right hand side $x_{v(t,s)}$ is always at least 0. On the other hand, for any split $s \in \mathbf{LS}(\ell^*)$, we will have that

$x_{v(t,s)} = 1$, and that

$$\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} = \sum_{\ell \in \mathbf{left}(s): \ell \neq \ell^*} y_{t,\ell} + y_{t,\ell^*} = 1,$$

which implies that constraint (4.17c) is satisfied. Similar reasoning can be used to establish that constraint (4.17d) holds. This establishes that $\mathbf{y}_t$ is indeed a feasible solution of problem (4.17).

Proof of part (b): By construction, $\alpha_{t,s} \geq 0$ and $\beta_{t,s} \geq 0$ for all $s \in \mathbf{splits}(t)$, so constraints (4.18c) and (4.18d) are satisfied. To verify constraint (4.18b), fix a leaf $\ell \in \mathbf{leaves}(t)$. If $\ell \neq \ell^*$, then either $\ell \in \mathbf{left}(s')$ for some $s' \in \mathbf{RS}(\ell^*)$ or $\ell \in \mathbf{right}(s')$ for some $s' \in \mathbf{LS}(\ell^*)$. If $\ell \in \mathbf{left}(s')$ for some $s' \in \mathbf{RS}(\ell^*)$, then

$$\sum_{s: \ell \in \mathbf{left}(s)} \alpha_{t,s} + \sum_{s: \ell \in \mathbf{right}(s)} \beta_{t,s} + \gamma_t \geq \alpha_{t,s'} + \gamma_t \geq \max_{\ell' \in \mathbf{left}(s')} r_{t,\ell'} - r_{t,\ell^*} + r_{t,\ell^*} \geq r_{t,\ell}$$

where the first inequality follows because $\ell \in \mathbf{left}(s')$ and the fact that all $\alpha_{t,s}$ and $\beta_{t,s}$ variables are nonnegative; the second follows by how the dual solution is defined in the statement of the theorem; and the third by the definition of the maximum. Similarly, if $\ell \in \mathbf{right}(s')$ for some $s' \in \mathbf{LS}(\ell^*)$, then by similar reasoning we have

$$\sum_{s: \ell \in \mathbf{left}(s)} \alpha_{t,s} + \sum_{s: \ell \in \mathbf{right}(s)} \beta_{t,s} + \gamma_t \geq \beta_{t,s'} + \gamma_t \geq \max_{\ell' \in \mathbf{right}(s')} r_{t,\ell'} - r_{t,\ell^*} + r_{t,\ell^*} \geq r_\ell - r_{t,\ell^*} + r_{t,\ell^*} = r_{t,\ell}.$$

Lastly, if $\ell = \ell^*$, then we automatically have

$$\sum_{s: \ell^* \in \mathbf{left}(s)} \alpha_{t,s} + \sum_{s: \ell^* \in \mathbf{right}(s)} \beta_{t,s} + \gamma_t \geq \gamma_t = r_{t,\ell^*}.$$

Thus, we have established that constraint (4.18b) is satisfied for all leaves ℓ, and thus $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ as defined in the statement of the theorem is a feasible solution of the dual (4.18).

Proof of part (c): To establish that the two solutions are optimal, by weak duality it is sufficient to show that the two solutions attain the same objective values in their respective

problems. For the primal solution $\mathbf{y}_t$, it is immediately clear that its objective is r_{t,ℓ^*}. For the dual solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$, we have

$$\begin{aligned}
&\sum_{s \in \mathbf{splits}(t)} \alpha_{t,s} x_{v(t,s)} + \sum_{s \in \mathbf{splits}(t)} \beta_{t,s}(1 - x_{v(t,s)}) + \gamma_t \\
&= \sum_{s \in \mathbf{RS}(\ell^*)} \alpha_{t,s} x_{v(t,s)} + \sum_{s \in \mathbf{LS}(\ell^*)} \beta_{t,s}(1 - x_{v(t,s)}) + \gamma_t \\
&= 0 + 0 + \gamma_t \\
&= r_{t,\ell^*}
\end{aligned}$$

where the first step follows because $\alpha_{t,s} = 0$ for $s \notin \mathbf{RS}(\ell^*)$ and $\beta_{t,s} = 0$ for $s \notin \mathbf{LS}(\ell^*)$; the second step follows by the fact that $x_{v(t,s)} = 0$ for $s \in \mathbf{RS}(\ell^*)$ and $x_{v(t,s)} = 1$ for $s \in \mathbf{LS}(\ell^*)$; and the final step follows just by the definition of γ_t. This establishes that $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for their respective problems, which concludes the proof. □

B.2.14 Proof of Proposition 9

To see that the ProductMIO primal subproblem (4.20) is not greedy solvable, consider an instance where $\mathcal{N} = \{1, 2, 3\}$, and the revenues of these products are $\bar{r}_1 = 20$, $\bar{r}_2 = 19$ and $\bar{r}_3 = 18$. Consider the tree shown in Figure B.5. Labeling the leaves as 1, 2, 3, 4, 5 and 6

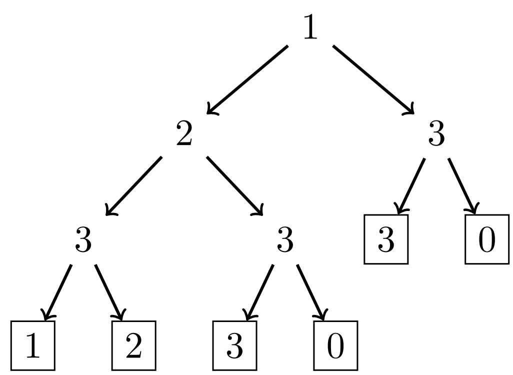


Figure B.5: Structure of tree for which the ProductMIO primal subproblem is not solvable via a greedy algorithm.

from left to right, the primal subproblem (4.20) can be explicitly written as

$$\begin{aligned}
\underset{\mathbf{y}}{\text{maximize}} \quad & 20y_1 + 19y_2 + 18y_3 + 18y_5 && \text{(B.61a)}\\
\text{subject to} \quad & y_1 + y_2 + y_3 + y_4 \leq 0.5 \quad (= x_1) && \text{(B.61b)}\\
& y_5 + y_6 \leq 0.5 \quad (= 1 - x_1) && \text{(B.61c)}\\
& y_1 + y_2 \leq 0.5 \quad (= x_2) && \text{(B.61d)}\\
& y_3 + y_4 \leq 0.5 \quad (= 1 - x_2) && \text{(B.61e)}\\
& y_1 + y_3 + y_5 \leq 0.5 \quad (= x_3) && \text{(B.61f)}\\
& y_2 + y_4 + y_6 \leq 0.5 \quad (= 1 - x_3) && \text{(B.61g)}\\
& y_1, \ldots, y_6 \geq 0, && \text{(B.61h)}
\end{aligned}$$

where we omit the subscript t to simplify notation. When we apply the greedy algorithm to solve this LO problem, we can see that there are multiple orderings of the leaves in decreasing

revenue:

$$1, 2, 3, 5, 4, 6$$

$$1, 2, 5, 3, 4, 6$$

$$1, 2, 3, 5, 6, 4$$

$$1, 2, 5, 3, 6, 4$$

For any of these orderings, the greedy solution will turn out to be

$$y_1 = 0.5,$$

$$y_2 = 0,$$

$$y_3 = 0,$$

$$y_4 = 0,$$

$$y_5 = 0,$$

$$y_6 = 0.5,$$

resulting in an objective value of $y_1 \times \bar{r}_1 + y_6 \times 0 = (0.5)(20) = 10$. However, the actual optimal solution of problem (B.61) turns out to be

$$y_1^* = 0,$$

$$y_2^* = 0.5,$$

$$y_3^* = 0,$$

$$y_4^* = 0,$$

$$y_5^* = 0.5,$$

$$y_6^* = 0,$$

for which the objective value is $y_2^* \times \bar{r}_2 + y_5^* \times \bar{r}_3 = (0.5)(19) + (0.5)(18) = 18.5$. This shows that in general, the ProductMIO primal subproblem cannot be solved to optimality via the same type of greedy algorithm as for LeafMIO and SplitMIO. □

B.2.15 Proof of Theorem 21

We prove that PRODUCTMIO primal and dual are closed form solvable for binary $\mathbf{x}$.

Proof of part (a) (primal feasibility): The solution $\mathbf{y}_t$ clearly satisfies the unit sum constraint (4.20b). For $i \in \mathbf{LP}(\ell^*)$, we know that $x_i = 1$, and by the definition of ℓ^*, we have that

$$\sum_{\ell \in \mathbf{left}(i)} y_{t,\ell} = y_{t,\ell^*} + \sum_{\ell \in \mathbf{left}(i): \ell \neq \ell^*} y_{t,\ell} = 1,$$

which implies that the left split constraint (4.20c) is satisfied for product i. Similarly, for $i \in \mathbf{RP}(\ell^*)$, we know that $x_i = 0$ (or equivalently, $1 - x_i = 1$, and we again have that $\sum_{\ell \in \mathbf{right}(i)} y_{t,\ell} = 1$, which implies that the right split constraint (4.20d) is satisfied at product i.

For $i \notin \mathbf{LP}(\ell^*)$, we know that $\ell^* \notin \mathbf{left}(i)$, and thus $\sum_{\ell \in \mathbf{left}(i)} y_{t,\ell} = 0$, which implies that constraint (4.20c) is automatically satisfied (the right hand side is x_i which can only be 0 or 1). Similarly, for $i \notin \mathbf{RP}(\ell^*)$, we know that $\ell^* \notin \mathbf{right}(i)$ and that $\sum_{\ell \in \mathbf{right}(i)} y_{t,\ell} = 0$, which similarly implies that constraint (4.20d) is satisfied (the right hand side is $1 - x_i$, which can only be 0 or 1). This establishes that $\mathbf{y}_t$ is a feasible solution of problem (4.20).

Proof of part (b) (dual feasibility): By construction, $\alpha_{t,i} \geq 0$ and $\beta_{t,i} \geq 0$ for all products i. Thus, we only need to verify the dual constraint (B.2b). If $\ell = \ell^*$, then the constraint is immediately satisfied, because $\gamma_t = r_{t,\ell^*}$ and $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ are nonnegative. If $\ell \neq \ell^*$, then either $\ell \in \mathbf{left}(i')$ for some $i' \in \mathbf{RP}(\ell^*)$ or $\ell \in \mathbf{right}(i')$ for some $i' \in \mathbf{LP}(\ell^*)$. In the former case, using the definition of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ and the nonnegativity of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$, we have

$$\sum_{i \in \mathbf{LP}(\ell)} \alpha_{t,i} + \sum_{i \in \mathbf{RP}(\ell)} \beta_{t,i} + \gamma_t \geq \alpha_{t,i'} + \gamma_t \geq \max_{\ell' \in \mathbf{left}(i')} r_{t,\ell'} - r_{\ell^*} + r_{\ell^*} \geq r_{t,\ell}$$

In the latter case, we similarly have

$$\sum_{i \in \mathbf{LP}(\ell)} \alpha_{t,i} + \sum_{i \in \mathbf{RP}(\ell)} \beta_{t,i} + \gamma_t \geq \beta_{t,i'} + \gamma_t \geq \max_{\ell' \in \mathbf{right}(i')} r_{t,\ell'} - r_{\ell^*} + r_{\ell^*} \geq r_{t,\ell}.$$

This establishes that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution of problem (B.2).

Proof of part (c) (optimality): We prove this by showing that the two solutions have the same objective value. For the primal solution, it is clear that its objective value is r_{t,ℓ^*}. For the dual solution, we have:

$$\sum_{i \in P(t)} \alpha_{t,i} x_i + \sum_{i \in P(t)} \beta_{t,i}(1 - x_i) + \gamma_t = \sum_{i \in \mathbf{RP}(\ell^*)} \alpha_{t,i} x_i + \sum_{i \in \mathbf{LP}(\ell^*)} \beta_{t,i}(1 - x_i) + \gamma_t = r_{t,\ell^*},$$

where the first step follows because $\alpha_{t,i}$ is defined to be zero when $i \notin \mathbf{RP}(\ell^*)$ and $\beta_{t,i}$ is defined to be zero when $i \notin \mathbf{LP}(\ell^*)$; and the second step follows because $x_i = 0$ when $i \in \mathbf{RP}(\ell^*)$ and $x_i = 1$ when $i \in \mathbf{LP}(\ell^*)$. This establishes that $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for their respective problems. □

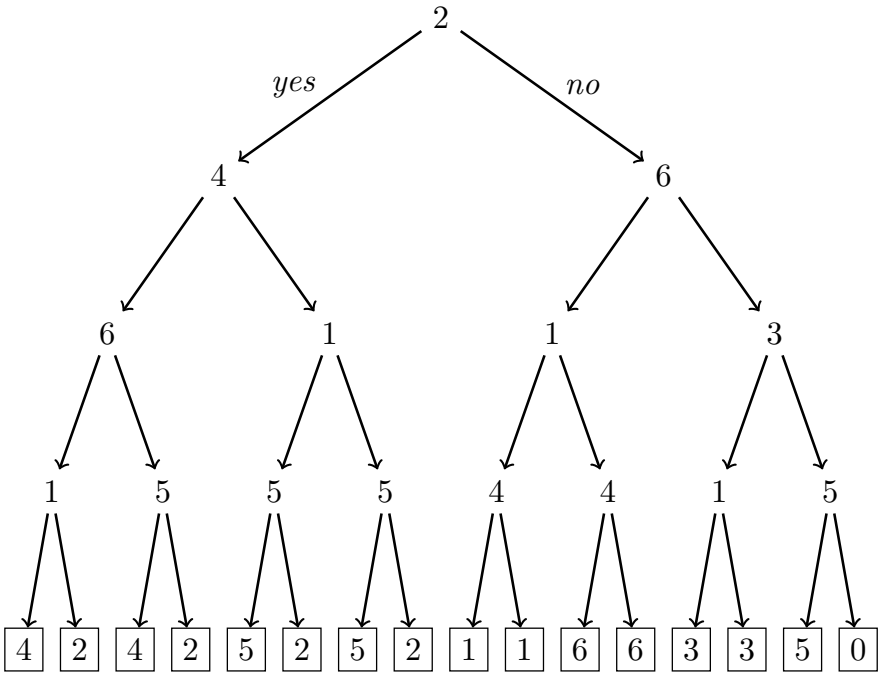
B.3 Additional Numerical Results

B.3.1 Example of Benders algorithms for SplitMIO

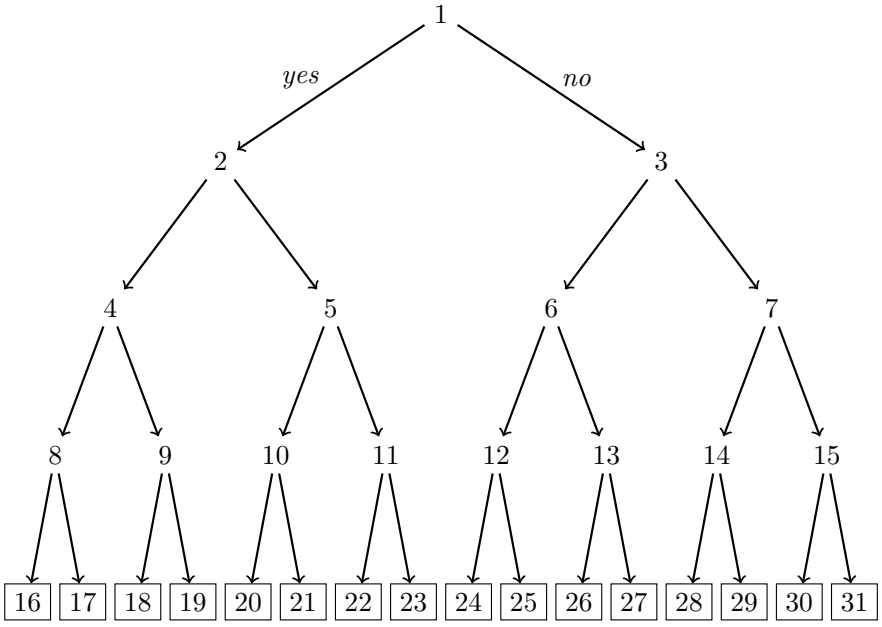
In this section, we provide a small example of the primal-dual procedure (Algorithms 6 and 7) for solving the SplitMIO subproblem.

Suppose that $n = 6$, and that $\mathbf{x} = (x_1, \ldots, x_6) = (0.62, 0.45, 0.32, 0.86, 0.05, 0.35)$. Suppose that $\bar{\mathbf{r}} = (\bar{r}_1, \ldots, \bar{r}_6) = (97, 72, 89, 50, 100, 68)$. Suppose that the purchase decision tree t has the form given in Figure B.6a; in addition, suppose that the splits and leaves are indexed as in Figure B.6b. Note that the top figure shows the purchase decision tree, in terms of the products at each split, and the purchase decision at each leaf. The bottom figure shows the indexing of nodes (for example, 8 corresponds to the split node that is furthest to the bottom and to the left, while 30 corresponds to the second leaf from the right).

We first run Algorithm 6 on the problem, which carries out the steps shown below in Table B.1. For this execution of the procedure, we assume that the following ordering of



(a) Purchase decision tree.



(b) Indexing of nodes in tree.

Figure B.6: Tree used in example of SplitMIO primal-dual algorithms.

leaves (encoded by τ) is used:

$$20, 22, 30, 24, 25, 28, 29, 17, 19, 21, 23, 26, 27, 16, 18, 31.$$

Iteration	Values of q_C and $q_{A,B}$	Steps
$\ell = 20$	$q_C = 1.0$, $q_{A,B} = 0.05$	Set $y_{20} \leftarrow 0.05$ A_{10} event
$\ell = 22$	$q_C = 0.95$, $q_{A,B} = 0.05$	Set $y_{22} \leftarrow 0.05$ A_{11} event
$\ell = 30$	$q_C = 0.90$, $q_{A,B} = 0.05$	Set $y_{30} \leftarrow 0.05$ A_{15} event
$\ell = 24$	$q_C = 0.85$, $q_{A,B} = 0.35$	Set $y_{24} \leftarrow 0.35$ A_3 event
$\ell = 25$	$q_C = 0.5$, $q_{A,B} = 0.0$	Set $y_{25} \leftarrow 0.0$
$\ell = 28$	$q_C = 0.5$, $q_{A,B} = 0.15$	Set $y_{28} \leftarrow 0.15$ B_1 event
$\ell = 29$	$q_C = 0.35$, $q_{A,B} = 0.0$	Set $y_{29} \leftarrow 0.0$
$\ell = 17$	$q_C = 0.35$, $q_{A,B} = 0.35$	Set $y_{17} \leftarrow 0.35$ C event **break**

Table B.1: Steps of primal procedure (Algorithm 6).

After running the procedure, the primal solution $\mathbf{y}$ is

$$\begin{aligned} y_{16} &= 0.0 & y_{17} &= 0.35 \\ y_{18} &= 0.0 & y_{19} &= 0.0 \\ y_{20} &= 0.05 & y_{21} &= 0.0 \\ y_{22} &= 0.05 & y_{23} &= 0.0 \\ y_{24} &= 0.35 & y_{25} &= 0.0 \\ y_{26} &= 0.0 & y_{27} &= 0.0 \\ y_{28} &= 0.15 & y_{29} &= 0.0 \\ y_{30} &= 0.05 & y_{31} &= 0.0 \end{aligned}$$

The event set is $\mathcal{E} = \{A_{10}, A_{11}, A_{15}, A_3, B_1, C\}$, and the function $f : \mathcal{E} \to \textbf{leaves}$ is defined as

$$\begin{aligned} f(A_{10}) &= 20, \\ f(A_{11}) &= 22, \\ f(A_{15}) &= 30, \\ f(A_3) &= 24, \\ f(B_1) &= 28, \\ f(C) &= 17. \end{aligned}$$

We now run Algorithm 7, which carries out the steps shown below in Table B.2.

Phase	Calculation
Initialization	$\alpha_s \leftarrow 0$, $\beta_s \leftarrow 0$ for all s
Set γ	$\gamma \leftarrow r_{17} = 72$
Loop: $d = 1$	$\beta_1 \leftarrow r_{28} - \gamma = 89 - 72 = 17$
Loop: $d = 2$	$\alpha_3 \leftarrow r_{24} - \gamma - \beta_1 = 97 - 72 - 17 = 8$
Loop: $d = 4$	$\alpha_{10} \leftarrow r_{20} - \gamma = 100 - 72 = 28$
	$\alpha_{11} \leftarrow r_{22} - \gamma = 100 - 72 = 28$
	$\alpha_{15} \leftarrow r_{30} - \gamma - \beta_1 = 100 - 72 - 17 = 11$

Table B.2: Steps of dual procedure (Algorithm 7).

After running the procedure, the dual solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ is

$$
\begin{aligned}
&\gamma = 72 \\
&\alpha_1 = 0 \quad \beta_1 = 17 \\
&\alpha_2 = 0 \quad \beta_2 = 0 \\
&\alpha_3 = 8 \quad \beta_3 = 0 \\
&\alpha_4 = 0 \quad \beta_4 = 0 \\
&\alpha_5 = 0 \quad \beta_5 = 0 \\
&\alpha_6 = 0 \quad \beta_6 = 0 \\
&\alpha_7 = 0 \quad \beta_7 = 0 \\
&\alpha_8 = 0 \quad \beta_8 = 0 \\
&\alpha_9 = 0 \quad \beta_9 = 0 \\
&\alpha_{10} = 28 \quad \beta_{10} = 0 \\
&\alpha_{11} = 28 \quad \beta_{11} = 0 \\
&\alpha_{12} = 0 \quad \beta_{12} = 0 \\
&\alpha_{13} = 0 \quad \beta_{13} = 0 \\
&\alpha_{14} = 0 \quad \beta_{14} = 0 \\
&\alpha_{15} = 11 \quad \beta_{15} = 0
\end{aligned}
$$

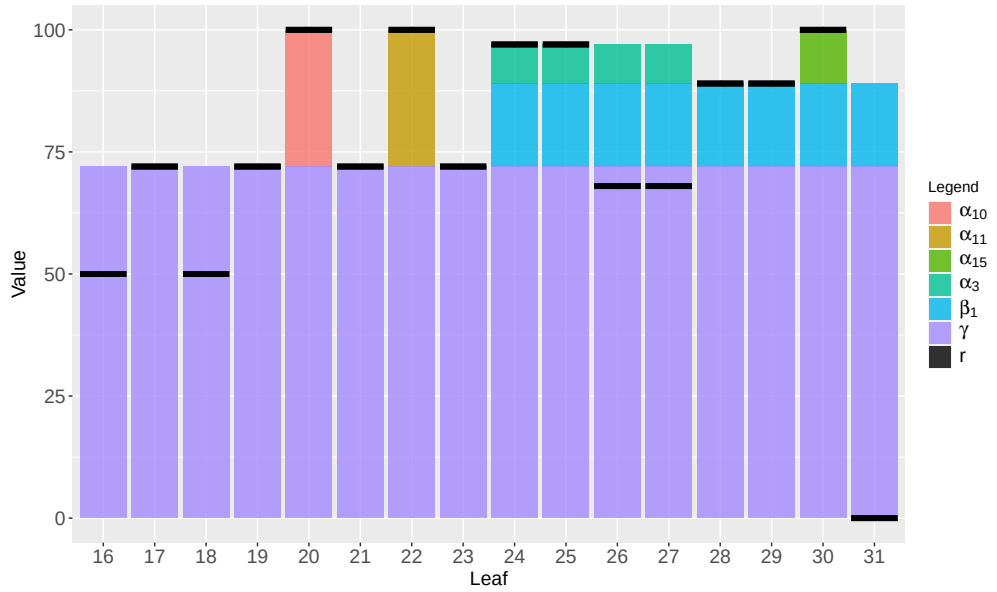


Figure B.7: Visualization of feasibility of dual solution in SplitMIO algorithm example.

The feasibility of the dual solution is visualized in Figure B.7. The colored bars correspond to the different dual variables; a colored bar appears multiple times when the variable participates in multiple dual constraints. The height of the black lines for each leaf indicates the value of r_ℓ, while the total height of the colored bars at a leaf corresponds to the value $\gamma + \sum_{s \in \mathbf{LS}(\ell)} \alpha_s + \sum_{s \in \mathbf{RS}(\ell)} \beta_s$ (the left hand side of the dual constraint (4.18b). For each leaf, the total height of the colored bars exceeds the black line, which indicates that all dual constraints are satisfied.

The objective value of the primal solution is

$$\begin{aligned} & r_{20} \times y_{20} + r_{22} \times y_{22} + r_{30} \times y_{30} + r_{24} \times y_{24} + r_{28} \times y_{28} + r_{17} \times y_{17} \\ &= 100 \times 0.05 + 100 \times 0.05 + 100 \times 0.05 + 97 \times 0.35 + 89 \times 0.15 + 72 \times 0.35 \\ &= 87.5 \end{aligned}$$

The objective value of the dual solution is

$$\begin{aligned}
&\gamma + (1 - x_2) \times \beta_1 + x_6 \times \alpha_3 + x_5 \times \alpha_{10} + x_5 \times \alpha_{11} + x_5 \times \alpha_{15} \\
&= 72.0 + 0.55 \times 17 + 0.35 \times 8 + 0.05 \times 28 + 0.05 \times 28 + 0.05 \times 11 \\
&= 87.5
\end{aligned}$$

B.3.2 Additional Results for Section 4.3.3

Table B.3 provides the same results as Table 4.2 for the T1 and T2 instances.

B.3.3 Comparison to Heuristic Approaches

In this experiment, we compare the performance of the three formulations to heuristic approaches. We will consider three different heuristic approaches:

1. LS: A local search heuristic, which starts from the empty assortment, and in each iteration moves to the neighboring assortment which improves the expected revenue the most. The neighborhood of assortments consists of those assortments obtained by adding a new product to the current assortment, or removing one of the existing products from the assortment. The heuristic terminates when there is no assortment in the neighborhood of the current one that provides an improvement.

2. LS10: This heuristic involves running LS from ten randomly chosen starting assortments. Each assortment is chosen uniformly at random from the set of 2^n possible assortments. After the ten repetitions, the assortment with the best expected revenue is retained.

3. ROA: This heuristic involves finding the optimal revenue ordered assortment. More formally, we define $S_k = \{i_1, \ldots, i_k\}$, where $i_1, \ldots, i_n$ corresponds to an ordering of the products so that $r_{i_1} \geq r_{i_2} \geq \cdots \geq r_{i_n}$, and we find $\arg\max_{S \in \{S_1, \ldots, S_n\}} R^{(F,\boldsymbol{\lambda})}(S)$.

We compare these heuristics against the best integer solution obtained by each of our three MIO formulations, leading to a total of six methods for each instance. We measure the performance of the solution corresponding to approach $\mathcal{M}$ using the metric $\bar{G}_{\mathcal{M}}$, which is defined as

$$\bar{G}_{\mathcal{M}} = 100\% \times \frac{Z_{BestUB} - Z_{\mathcal{M}}}{Z_{BestUB}},$$

where Z_{BestUB} is the best (lowest) upper bound obtained from among the three MIO formulations, and $Z_{\mathcal{M}}$ is the objective value of the solution returned by approach $\mathcal{M}$.

Table B.4 shows the performance of the six approaches – LeafMIO, SplitMIO, ProductMIO, LS, LS10 and ROA – for each family of instances. The gaps are averaged over the twenty instances for each combination of instance type, $|F|$ and $|\mathbf{leaves}(t)|$. We can see from this table that in general, for the small instances, the solutions obtained by the MIO formulations are either optimal or near optimal, while the solutions produced by the heuristic approaches are quite suboptimal. For cases where the gap is zero for the MIO formulations, the gap of LS ranges from 1.5% to 14.4%; the LS10 heuristic improves on this, due to its use of restarting and randomization, but still does not perform as well as the MIO solutions (gaps ranging from 0.5 to 7.4%). For the larger instances, where the gap of the MIO solutions is larger, LS and LS10 still tend to perform worse. Across all of the instances, ROA achieves much higher gaps than all of the other approaches (ranging from 14.6 to 40.6%). Overall, these results suggest that the very general structure of the decision forest model poses significant difficulty to standard heuristic approaches, and highlight the value of using exact approaches over inexact/heuristic approaches to the assortment optimization problem.

Type	$\lvert F\rvert$	$\lvert\mathbf{leaves}(t)\rvert$	$\tilde{G}_{\text{LeafMIO}}$	$\tilde{G}_{\text{SplitMIO}}$	$\tilde{G}_{\text{ProductMIO}}$	T_{LeafMIO}	T_{SplitMIO}	$T_{\text{ProductMIO}}$
T1	50	8	0.0	0.0	0.0	0.1	0.0	0.0
T1	50	16	0.0	0.0	0.0	0.4	0.2	0.0
T1	50	32	0.0	0.0	0.0	2.4	0.8	0.2
T1	50	64	0.0	0.0	0.0	40.9	7.5	1.4
T1	100	8	0.0	0.0	0.0	0.3	0.1	0.0
T1	100	16	0.0	0.0	0.0	3.7	1.5	0.3
T1	100	32	0.0	0.0	0.0	158.1	61.8	5.9
T1	100	64	1.8	0.4	0.0	3071.6	2700.6	178.7
T1	200	8	0.0	0.0	0.0	1.2	0.6	0.1
T1	200	16	0.0	0.0	0.0	334.2	380.5	13.8
T1	200	32	4.4	4.9	0.2	3600.1	3600.1	2093.5
T1	200	64	11.7	10.8	7.7	3600.1	3600.1	3600.0
T1	500	8	0.0	0.0	0.0	130.9	241.9	6.3
T1	500	16	8.1	12.0	5.0	3600.1	3600.2	3600.1
T1	500	32	17.9	19.6	14.7	3602.5	3600.1	3600.0
T1	500	64	22.9	22.6	19.8	3600.6	3601.4	3600.1
T2	50	8	0.0	0.0	0.0	0.1	0.0	0.0
T2	50	16	0.0	0.0	0.0	0.5	0.2	0.2
T2	50	32	0.0	0.0	0.0	5.4	0.8	0.8
T2	50	64	0.0	0.0	0.0	859.9	84.7	56.8
T2	100	8	0.0	0.0	0.0	0.3	0.1	0.1
T2	100	16	0.0	0.0	0.0	23.4	3.0	3.3
T2	100	32	2.9	0.0	0.0	3334.3	1436.0	767.4
T2	100	64	9.1	6.9	6.6	3600.3	3600.1	3600.0
T2	200	8	0.0	0.0	0.0	3.4	0.6	0.6
T2	200	16	6.1	1.8	1.4	3566.4	2683.8	2347.2
T2	200	32	15.0	12.9	12.6	3600.2	3600.1	3600.1
T2	200	64	20.5	18.5	18.5	3600.2	3600.2	3600.1
T2	500	8	1.9	0.0	0.0	3000.0	981.1	591.6
T2	500	16	20.8	19.1	18.8	3600.1	3600.3	3600.1
T2	500	32	27.9	25.4	25.0	3600.2	3600.1	3600.1
T2	500	64	33.5	30.7	30.4	3601.4	3600.1	3600.1

Table B.3: Comparison of final optimality gaps and computation times for LeafMIO, SplitMIO and ProductMIO, for T1 and T2 instances.

Type	$\lvert F\rvert$	$\lvert\mathbf{leaves}(t)\rvert$	$\bar{G}_{\text{LeafMIO}}$	$\bar{G}_{\text{SplitMIO}}$	$\bar{G}_{\text{ProductMIO}}$	$\bar{G}_{\text{LS}}$	$\bar{G}_{\text{LS10}}$	$\bar{G}_{\text{ROA}}$
T1	50	8	0.0	0.0	0.0	8.1	2.3	26.8
T1	50	16	0.0	0.0	0.0	9.8	4.0	31.2
T1	50	32	0.0	0.0	0.0	9.0	4.8	27.5
T1	50	64	0.0	0.0	0.0	10.0	6.4	28.6
T1	100	8	0.0	0.0	0.0	3.9	2.2	26.0
T1	100	16	0.0	0.0	0.0	6.6	3.5	26.2
T1	100	32	0.0	0.0	0.0	8.5	6.2	25.6
T1	100	64	0.0	0.0	0.0	9.9	6.8	24.5
T1	200	8	0.0	0.0	0.0	3.5	1.2	19.9
T1	200	16	0.0	0.0	0.0	5.5	2.7	21.7
T1	200	32	0.2	0.2	0.2	7.9	5.1	22.4
T1	200	64	8.2	7.8	7.7	14.6	13.7	25.6
T1	500	8	0.0	0.0	0.0	1.5	0.5	14.6
T1	500	16	5.0	5.2	5.0	8.5	6.6	19.5
T1	500	32	15.1	15.0	14.7	19.0	17.1	28.0
T1	500	64	20.3	20.3	19.8	23.3	22.2	30.4
T2	50	8	0.0	0.0	0.0	13.8	2.7	31.5
T2	50	16	0.0	0.0	0.0	11.6	4.3	32.2
T2	50	32	0.0	0.0	0.0	10.0	4.9	31.1
T2	50	64	0.0	0.0	0.0	11.6	6.8	30.4
T2	100	8	0.0	0.0	0.0	5.5	1.6	28.1
T2	100	16	0.0	0.0	0.0	8.2	3.6	30.8
T2	100	32	0.0	0.0	0.0	8.9	5.7	31.3
T2	100	64	7.1	6.6	6.6	17.3	12.2	31.8
T2	200	8	0.0	0.0	0.0	3.8	1.0	23.1
T2	200	16	1.4	1.4	1.4	6.8	3.9	25.3
T2	200	32	12.9	12.6	12.5	18.5	16.3	34.0
T2	200	64	19.3	18.3	18.5	24.4	21.2	37.0
T2	500	8	0.0	0.0	0.0	2.0	0.5	15.6
T2	500	16	18.9	18.8	18.7	21.5	19.6	31.7
T2	500	32	26.3	25.2	25.0	28.2	25.8	35.9
T2	500	64	32.4	30.5	30.4	31.4	29.5	38.4
T3	50	8	0.0	0.0	0.0	13.2	3.3	33.1
T3	50	16	0.0	0.0	0.0	14.4	5.9	34.9
T3	50	32	0.0	0.0	0.0	12.6	6.1	33.7
T3	50	64	0.0	0.0	0.0	13.9	7.4	33.0
T3	100	8	0.0	0.0	0.0	8.0	1.4	30.2
T3	100	16	0.0	0.0	0.0	10.0	3.4	32.6
T3	100	32	0.0	0.0	0.0	10.4	3.9	32.0
T3	100	64	3.6	3.5	3.5	13.1	8.9	33.4
T3	200	8	0.0	0.0	0.0	4.0	0.8	25.8
T3	200	16	0.0	0.0	0.0	6.5	2.2	26.5
T3	200	32	8.7	8.6	8.6	15.4	11.1	33.1
T3	200	64	16.0	15.7	15.6	23.0	18.3	35.6
T3	500	8	0.0	0.0	0.0	2.6	0.5	15.8
T3	500	16	13.7	13.7	13.8	17.4	14.4	27.9
T3	500	32	23.7	22.9	23.0	26.5	23.4	34.8
T3	500	64	31.7	30.9	30.8	33.0	30.2	40.6

Table B.4: Comparison of integer solutions from LeafMIO, SplitMIO and ProductMIO against heuristic solutions from LS, LS10 and ROA.

APPENDIX C

Appendix to Chapter 5

C.1 Omitted Proofs

C.1.1 Proof of Theorem 11 and 12

C.1.1.1 Preliminary Results and Lemmas

Lemma 12 and 13 bound the distance between the sample mean and the expected value of a collection of i.i.d. vectors, in terms of ℓ_2 norm and ℓ_1 norm, respectively. Lemma 12 is Lemma 4 from [100], which utilizes McDiarmid's inequality to show that the scalar function $\|\bar{\mathbf{w}} - \mathbb{E}[\bar{\mathbf{w}}]\|_2$, where $\bar{\mathbf{w}}$ is the mean of K i.i.d. vectors $\mathbf{w}_1, \ldots, \mathbf{w}_K$, concentrates to zero with rate $O\left(1/\sqrt{K}\right)$.

Lemma 12 *[100] Let $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$ be i.i.d. random vectors such that $\|\mathbf{w}_k\|_2 \leq C$ for $k = 1, \ldots, K$. Let $\bar{\mathbf{w}} = (1/K) \cdot \sum_{k=1}^{K} \mathbf{w}_k$. Then for any $\delta \in (0, 1)$, we have, with probability at least $1 - \delta$,*

$$\|\bar{\mathbf{w}} - \mathbb{E}[\bar{\mathbf{w}}]\|_2 \leq \frac{C}{\sqrt{K}} \cdot \left(1 + \sqrt{2 \log \frac{1}{\delta}}\right).$$

Lemma 13 *Let $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$ be i.i.d. random vectors of size m such that $\|\mathbf{w}_k\|_\infty \leq C$ for $k = 1, \ldots, K$. Let $\bar{\mathbf{w}} = (1/K) \cdot \sum_{k=1}^{K} \mathbf{w}_k$. Then for any $\delta \in (0, 1)$, we have, with probability at least $1 - \delta$,*

$$\|\bar{\mathbf{w}} - \mathbb{E}[\bar{\mathbf{w}}]\|_1 \leq \frac{mC}{\sqrt{K}} \cdot \left(1 + \sqrt{2 \log \frac{1}{\delta}}\right).$$

Proof: Since $\|\mathbf{w}_k\|_2 \leq \sqrt{m}\|\mathbf{w}_k\|_\infty \leq \sqrt{m}C$, we apply Lemma 12 and obtain that with probability $1-\delta$, $\|\bar{\mathbf{w}} - \mathbb{E}[\bar{\mathbf{w}}]\|_2 \leq \sqrt{m} \cdot C/\sqrt{K} \cdot \left(1 + \sqrt{2\log\frac{1}{\delta}}\right)$. Combining this with the fact that $\|\bar{\mathbf{w}} - \mathbb{E}[\bar{\mathbf{w}}]\|_1 \leq \sqrt{m} \cdot \|\bar{\mathbf{w}} - \mathbb{E}[\bar{\mathbf{w}}]\|_2$, we obtain the desired result. □

Lemma 14 is a standard result of sensitivity analysis of linear programming; see Chapter 5 of [14]. In fact, one can view the optimal objective value of problem P as a convex function in $\mathbf{b}$ and show that the dual solution $\mathbf{p}$ is a subgradient at $\mathbf{b}$.

Lemma 14 *Let* $z(\mathbf{b}) = \min\left\{\mathbf{c}_0^T\mathbf{y} \mid \mathbf{A}_0\mathbf{y} = \mathbf{b}, \mathbf{y} \geq \mathbf{0}\right\}$ *and* $z(\mathbf{b}') = \min\left\{\mathbf{c}_0^T\mathbf{y} \mid \mathbf{A}_0\mathbf{y} = \mathbf{b}', \mathbf{y} \geq \mathbf{0}\right\}$. *Then* $z(\mathbf{b}) - z(\mathbf{b}') \leq \mathbf{p}^T(\mathbf{b} - \mathbf{b}')$, *where* $\mathbf{p}$ *is an optimal dual solution of the former problem.*

C.1.1.2 Proofs of Main Theorems

We first establish a useful result.

Proposition 17 *Let* C *be a nonnegative constant and define the linear program* P_{distr} *as in Theorem 11, i.e.,* P_{distr} : $\min\left\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{0} \leq \mathbf{x} \leq C\boldsymbol{\xi}\right\}$. *Let* P_J *be the column-randomized LP solved by Algorithm 8. For any* $\delta \in (0,1)$, *with probability at least* $1-\delta$ *over the sample* J, *the following holds: if* P_J *is feasible, then*

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + \frac{C}{\sqrt{K}} \cdot (1 + \|\mathbf{p}\|_\infty \cdot m \cdot \|\mathbf{A}\|_{\max}) \cdot \left(1 + \sqrt{2\log\frac{2}{\delta}}\right)$$

for any optimal solution $\mathbf{p}$ *of problem* D_J *(the dual of problem* P_J*).*

Proof: Let $j_1, \ldots, j_K$ be the set of indices sampled according to the distribution $\boldsymbol{\xi}$ by the randomization scheme ρ. Let $\mathbf{x}^{*0}$ be an optimal solution of the distributional counterpart problem P_{distr}. Consider the solution $\mathbf{x}'$ that is defined as

$$\mathbf{x}' \equiv \frac{1}{K}\sum_{k=1}^{K} \frac{x_{j_k}^{*0}}{\xi_{j_k}} \cdot \mathbf{e}_{j_k},$$

where we use $\mathbf{e}_j$ to denote the jth standard basis vector for $\mathbb{R}^n$. In addition, define the vector $\mathbf{b}'$ as

$$\mathbf{b}' \equiv \mathbf{A}\mathbf{x}'.$$

To prove our result, we proceed in three steps. In the first step, we show how we can probabilistically bound $\|\mathbf{x}' - \mathbf{x}^{*0}\|_2$. In the second step, we show how we can probabilistically bound $\|\mathbf{b}' - \mathbf{b}\|_1$. In the last step, we use the results of our first two steps, together with sensitivity results for linear programs, to derive the required bound. In what follows, we use I_+ to denote the support of $\boldsymbol{\xi}$, that is, $I_+ = \{j \in [n] \mid \xi_j > 0\}$.

Step 1: Bounding $\|\mathbf{x}' - \mathbf{x}^{*0}\|_2$. To show that $\mathbf{x}'$ will be close to $\mathbf{x}^{*0}$, let us first define the vector $\mathbf{w}_k$ as

$$\mathbf{w}_k = \frac{x^{*0}_{j_k}}{\xi_{j_k}} \cdot \mathbf{e}_{j_k}$$

for each $k \in [K]$. The vectors $\mathbf{w}_1, \ldots, \mathbf{w}_K$ constitute an i.i.d. collection of vectors, and possess three special properties. First, observe that $\mathbf{x}'$ is just the sample mean of $\mathbf{w}_1, \ldots, \mathbf{w}_K$. Second, observe that the expected value of each $\mathbf{w}_k$ can be calculated as

$$\mathbb{E}[\mathbf{w}] = \sum_{j \in I_+} \xi_j \cdot \frac{x^{*0}_j}{\xi_j} \cdot \mathbf{e}_j = \sum_{j \in I_+} x^{*0}_j \mathbf{e}_j = \sum_{j \in [n]} x^{*0}_j \mathbf{e}_j = \mathbf{x}^{*0}$$

where we use $\mathbf{w}$ to denote a random vector following the same distribution as each $\mathbf{w}_k$. In the above, we note that the third step follows because the distributional counterpart P_{distr} includes the constraint $\mathbf{x} \leq C\boldsymbol{\xi}$, so $j \notin I_+$ automatically implies that $x^{*0}_j = 0$.

Finally, observe that the ℓ_2 norm of each $\mathbf{w}_k$ can be bounded as

$$\|\mathbf{w}_k\|_2 = \left| \frac{x^{*0}_{j_k}}{\xi_{j_k}} \right| \cdot \|\mathbf{e}_{j_k}\|_2 \leq C \cdot 1 = C,$$

where the inequality follows because $\mathbf{x}^{*0}$ satisfies the constraint $\mathbf{0} \leq \mathbf{x} \leq C\boldsymbol{\xi}$. With these three properties in hand, and recognizing that $\|\mathbf{x}' - \mathbf{x}^{*0}\|_2 = \|(1/K)\sum_{k=1}^K \mathbf{w}_k - \mathbb{E}[\mathbf{w}]\|_2$, we can invoke Lemma 12 to assert that, with probability at least $1 - \delta/2$,

$$\|\mathbf{x}' - \mathbf{x}^{*0}\|_2 \leq \frac{C}{\sqrt{K}} \cdot \left(1 + \sqrt{2 \log \frac{2}{\delta}}\right). \tag{C.1}$$

Step 2: Bounding $\|\mathbf{b}' - \mathbf{b}\|_1$. To show that $\mathbf{b}'$ will be close $\mathbf{b}$, we proceed similarly to Step 1. In particular, we define $\mathbf{b}_k$ for each $k \in [K]$ as

$$\mathbf{b}_k \equiv \mathbf{A}\mathbf{w}_k = \frac{x^{*0}_{j_k}}{\xi_{j_k}} \cdot \mathbf{A}\mathbf{e}_{j_k} = \frac{x^{*0}_{j_k}}{\xi_{j_k}} \mathbf{A}_{j_k}.$$

Observe that by definition of $\mathbf{b}_k$, we have that the sample mean of $\mathbf{b}_1, \ldots, \mathbf{b}_K$ is equal to $\mathbf{b}'$:

$$\frac{1}{K}\sum_{k=1}^{K} \mathbf{b}_k = \frac{1}{K}\sum_{k=1}^{K} \mathbf{A}\mathbf{w}_k = \mathbf{A}\left(\frac{1}{K}\sum_{k=1}^{K} \mathbf{w}_k\right) = \mathbf{A}\mathbf{x}' \equiv \mathbf{b}'. \tag{C.2}$$

In addition, the expected value of each $\mathbf{b}_k$ can be calculated; letting $\tilde{\mathbf{b}}$ denote a random variable with the same distribution as each $\mathbf{b}_k$, we have

$$\mathbb{E}[\tilde{\mathbf{b}}] = \mathbf{A}\mathbb{E}[\mathbf{w}_k] = \mathbf{A}\mathbf{x}^{*0} = \mathbf{b}.$$

Lastly, we can bound the ℓ_∞ norm of each vector $\mathbf{b}_k$ as

$$\|\mathbf{b}_k\|_\infty = \left\| \frac{x^{*0}_{j_k}}{\xi_{j_k}} \mathbf{A}_{j_k} \right\|_\infty = \left| \frac{x^{*0}_{j_k}}{\xi_{j_k}} \right| \cdot \|\mathbf{A}_{j_k}\|_\infty \leq C\|\mathbf{A}\|_{\max},$$

where the inequality follows by the definition of $\|\mathbf{A}\|_{\max}$ and the fact that $\mathbf{x}^{*0}$ satisfies $\mathbf{0} \leq \mathbf{x} \leq C\boldsymbol{\xi}$.

With these observations in hand, we now recognize that $\|\mathbf{b}' - \mathbf{b}\|_1 = \|(1/K)\sum_{k=1}^{K} \mathbf{b}_k - \mathbb{E}[\tilde{\mathbf{b}}]\|_1$, i.e., $\|\mathbf{b}' - \mathbf{b}\|_1$ is just the ℓ_1 norm of the deviation of a sample mean from its true expectation; we can therefore invoke Lemma 13 to assert that, with probability at least $1 - \delta/2$,

$$\|\mathbf{b}' - \mathbf{b}\|_1 \leq \frac{m \cdot C \cdot \|\mathbf{A}\|_{\max}}{\sqrt{K}} \cdot \left(1 + \sqrt{2\log\frac{2}{\delta}}\right). \tag{C.3}$$

Step 3: Completing the proof. With Steps 1 and 2 complete, we are now ready to bound the optimality gap. For any vector $\mathbf{b}'' \in \mathbb{R}^m$, we define the linear program $P_J(\mathbf{b}'')$ as

$$P_J(\mathbf{b}''): \quad \min\left\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}'', \mathbf{x} \geq \mathbf{0},\ \ x_j = 0\ \forall j \notin J\right\}. \tag{C.4}$$

Then $v(P_J(\mathbf{b}')) \leq \mathbf{c}^T\mathbf{x}'$; this follows because $\mathbf{A}\mathbf{x}' = \mathbf{b}'$ and $\mathbf{x}' \geq \mathbf{0}$, which means that $\mathbf{x}'$ is a feasible solution to problem $P_J(\mathbf{b}')$. In addition, since $\mathbf{c}^T\mathbf{x}^{*0} = v(P_{\text{distr}})$, we have

$$v(P_J(\mathbf{b}')) \leq \mathbf{c}^T\mathbf{x}' = \mathbf{c}^T\left(\mathbf{x}^{*0} + (\mathbf{x}' - \mathbf{x}^{*0})\right) = v(P_{\text{distr}}) + \mathbf{c}^T(\mathbf{x}' - \mathbf{x}^{*0}). \tag{C.5}$$

If the column-randomized problem P_J is feasible, then by letting $\mathbf{p}$ be any optimal solution of the dual of P_J and applying Lemma 14, we have

$$v(P_J) = v(P_J(\mathbf{b})) \leq v(P_J(\mathbf{b}')) + \mathbf{p}^T(\mathbf{b} - \mathbf{b}') \tag{C.6}$$

$$\leq v(P_{\text{distr}}) + \mathbf{c}^T(\mathbf{x}' - \mathbf{x}^{*0}) + \mathbf{p}^T(\mathbf{b} - \mathbf{b}') \tag{C.7}$$

$$\leq v(P_{\text{distr}}) + \|\mathbf{c}\|_2 \cdot \|\mathbf{x}' - \mathbf{x}^{*0}\| + \|\mathbf{p}\|_\infty \cdot \|\mathbf{b}' - \mathbf{b}\|_1 \tag{C.8}$$

$$= v(P_{\text{distr}}) + \|\mathbf{x}' - \mathbf{x}^{*0}\|_2 + \|\mathbf{p}\|_\infty \cdot \|\mathbf{b}' - \mathbf{b}\|_1, \tag{C.9}$$

where the first inequality comes from Lemma 14, the second inequality comes from (C.5), the third inequality comes from the Cauchy-Schwarz inequality and Hölder's inequality, and the last equality comes from the assumption that $\|\mathbf{c}\|_2 = 1$.

We now bound expression (C.9) by applying the inequalities (C.1) and (C.3), each of which hold with probability at least $1 - \delta/2$, and combining them using the union bound. We thus obtain that, with probability at least $1 - \delta$,

$$v(P_J) \leq v(P_{\text{distr}}) + \frac{C}{\sqrt{K}} \cdot (1 + \|\mathbf{p}\|_\infty \cdot m \cdot \mathbf{A}_{\max}) \cdot \left(1 + \sqrt{2 \log \frac{2}{\delta}}\right). \tag{C.10}$$

Subtracting $v(P)$ from both sides gives us the required inequality. □

With Proposition 17, we can smoothly prove Theorem 11 as follows.

Proof of Theorem 11:

By invoking Proposition 17, we obtain that with probability at least $1-\delta$, if P_J is feasible, then

$$\Delta v(P_J) \leq \Delta v(P_{\text{distr}}) + \frac{C}{\sqrt{K}} \cdot (1 + \|\mathbf{p}\|_\infty \cdot m \cdot \mathbf{A}_{\max}) \cdot \left(1 + \sqrt{2 \log \frac{2}{\delta}}\right),$$

for any dual optimal solution $\mathbf{p}$ of D_J. To prove the theorem, let us set $\mathbf{p}$ to an optimal basic feasible solution of the problem D_J. Note that such a dual optimal solution is guaranteed to exist by the assumption that $\text{rank}(\mathbf{A}_J) = m$. Since $\mathbf{p}$ is a basic feasible solution of D_J, it

is automatically a basic (but not necessarily feasible) solution of the complete dual problem D. By the definition of γ in the theorem, we have that $\|\mathbf{p}\|_\infty \leq \gamma$, and the theorem follows. $\square$

To prove Theorem 12, we prove a complementary result to Proposition 17.

Proposition 18 *Let C, P_J and P_{distr} be defined as in the statement of Proposition 17. For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sample J, the following holds: if P_J is feasible, then*

$$\Delta v(P_J) \leq \Delta v(P_{distr}) + \frac{C}{\sqrt{K}} \cdot \|\mathbf{c}^T - \mathbf{p}^T\mathbf{A}\|_2 \cdot \left(1 + \sqrt{2\log\frac{1}{\delta}}\right)$$

for any optimal solution $\mathbf{p}$ of problem D_J (the dual of problem P_J).

Proof: We follow the proof of Proposition 17 until inequality (C.7) and continue as follows:

$$\begin{aligned} v(P_J) = v(P_J(\mathbf{b})) &\leq v(P_J(\mathbf{b}')) + \mathbf{p}^T(\mathbf{b}-\mathbf{b}') \\ &\leq v(P_{\text{distr}}) + \mathbf{c}^T(\mathbf{x}'-\mathbf{x}^{*0}) + \mathbf{p}^T(\mathbf{b}-\mathbf{b}') \\ &= v(P_{\text{distr}}) + \mathbf{c}^T(\mathbf{x}'-\mathbf{x}^{*0}) + \mathbf{p}^T\mathbf{A}(\mathbf{x}^{*0}-\mathbf{x}') \\ &= v(P_{\text{distr}}) + \left(\mathbf{c}^T - \mathbf{p}^T\mathbf{A}\right)(\mathbf{x}'-\mathbf{x}^{*0}) \\ &\leq v(P_{\text{distr}}) + \|\mathbf{c}^T - \mathbf{p}^T\mathbf{A}\|_2 \cdot \|\mathbf{x}'-\mathbf{x}^{*0}\|_2, \end{aligned} \tag{C.11}$$

where the bound holds for any optimal solution $\mathbf{p}$ of the sampled dual problem D_J. By invoking Lemma 12 with δ to bound $\|\mathbf{x}'-\mathbf{x}^{*0}\|_2$, and subtracting $v(P)$ from both sides, we obtain the desired result. $\square$

Using Proposition 18, we now prove Theorem 12.

Proof of Theorem 12: We invoke Proposition 18 and set $\mathbf{p}$ to be an optimal basic feasible solution of the sampled dual problem D_J; then $\mathbf{p}^T = \mathbf{c}_B^T\mathbf{A}_B^{-1}$ for some set of basic variables $B \subset [n]$. In this case, we observe that the dual slack vector $\mathbf{c}^T - \mathbf{p}^T\mathbf{A}$ becomes $\mathbf{c}^T - \mathbf{c}_B^T\mathbf{A}_B^{-1}\mathbf{A}$,

which is exactly the reduced cost vector $\bar{\mathbf{c}}$ associated with the basis B within the full problem P. By using the hypothesis that any such reduced cost vector satisfies $\|\bar{\mathbf{c}}\|_2 \leq \chi$, we obtain the desired result. $\square$

C.1.2 Proof of Proposition 10

Let $\mathbf{x}^{*0}$ be an optimal solution of $P^{\text{feas}}_{\text{distr}}$. Define the solution $\mathbf{x}'$ as

$$\mathbf{x}' = \frac{1}{K} \sum_{k=1}^{K} \frac{x^{*0}_{j_k}}{\xi_{j_k}} \cdot \mathbf{e}_{j_k}.$$

With $\mathbf{x}'$, we can bound the objective value of P^{feas}_J as follows:

$$\begin{aligned} v(P^{\text{feas}}_J) &\leq \|\mathbf{A}\mathbf{x}' - \mathbf{b}\|_1 \\ &= \|\mathbf{A}\mathbf{x}' - \mathbf{A}\mathbf{x}^{*0} + \mathbf{A}\mathbf{x}^{*0} - \mathbf{b}\|_1 \\ &\leq \|\mathbf{A}\mathbf{x}' - \mathbf{A}\mathbf{x}^{*0}\|_1 + \|\mathbf{A}\mathbf{x}^{*0} - \mathbf{b}\|_1 \\ &= \|\mathbf{A}\mathbf{x}' - \mathbf{A}\mathbf{x}^{*0}\|_1 + v(P^{\text{feas}}_{\text{distr}}) \end{aligned} \tag{C.12}$$

where the first step follows by the fact that $\mathbf{x}'$, when restricted to the indices in J, is a feasible solution of P^{feas}_J; the third step follows by the triangle inequality; and the fourth follows by the definition of $\mathbf{x}^{*0}$ as an optimal solution of $P^{\text{feas}}_{\text{distr}}$.

The only remaining step is to bound $\|\mathbf{A}\mathbf{x}' - \mathbf{A}\mathbf{x}^{*0}\|_1$. To do this, let us define the vector $\mathbf{v}_k$ as

$$\mathbf{v}_k = \frac{x^{*0}_{j_k}}{\xi_{j_k}} \mathbf{A}_{j_k}$$

for each $k \in [K]$. The vectors $\mathbf{v}_1, \ldots, \mathbf{v}_K$ are special for three reasons. First, their sample mean is exactly

$$\frac{1}{K} \sum_{k=1}^{K} \mathbf{v}_k = \frac{1}{K} \sum_{k=1}^{K} \frac{x^{*0}_{j_k}}{\xi_{j_k}} \mathbf{A}_{j_k} = \frac{1}{K} \sum_{k=1}^{K} \frac{x^{*0}_{j_k}}{\xi_{j_k}} \mathbf{A}\mathbf{e}_{j_k} = \mathbf{A}\mathbf{x}'.$$

Second, letting $\mathbf{v}$ denote a random variable following the same distribution as each $\mathbf{v}_k$, the

expected value of each $\mathbf{v}_k$ is

$$\mathbb{E}[\mathbf{v}] = \sum_{j \in I_+} \xi_j \cdot \frac{x_j^{*0}}{\xi_j} \mathbf{A}_j = \sum_{j \in I_+} x_j^{*0} \mathbf{A}_j = \sum_{j \in [n]} x^{*0_j} \mathbf{A}_j = \mathbf{A}\mathbf{x}^{*0}$$

where I_+ is the subset of indices in $[n]$ such that $\xi_j > 0$. Note that the third step is justified by observing that $\xi_j^{*0} = 0$ whenever $j \notin I_+$ (this is because of the constraint $\mathbf{0} \leq \mathbf{x} \leq C\boldsymbol{\xi}$ in the definition of $P_{\text{distr}}^{\text{feas}}$).

Lastly, observe that each $\mathbf{v}_k$ is bounded as

$$\|\mathbf{v}_k\|_\infty = \frac{x_{j_k}^{*0}}{\xi_{j_k}} \cdot \|\mathbf{A}_{j_k}\|_\infty \leq C \cdot H,$$

where we use the hypothesis that $\|\mathbf{A}_j\|_\infty \leq \|\mathbf{A}\|_{\max}$ and the fact that $\mathbf{x}^{*0}$ satisfies $\mathbf{0} \leq \mathbf{x}^{*0} \leq C\boldsymbol{\xi}$.

With all of these properties, the quantity $\|\mathbf{A}\mathbf{x}' - \mathbf{A}\mathbf{x}^{*0}\|_1$ can be re-written as $\|(1/K)\sum_{k=1}^K \mathbf{v}_k - \mathbb{E}[\mathbf{v}]\|_1$, which we can bound using Lemma 13 (see Section C.1.1). Invoking Lemma 13, we get that

$$\begin{aligned}\|\mathbf{A}\mathbf{x}' - \mathbf{A}\mathbf{x}^{*0}\|_1 &= \|\frac{1}{K}\sum_{k=1}^K \mathbf{v}_k - \mathbb{E}[\mathbf{v}]\|_1 \\ &\leq \frac{mC\|\mathbf{A}\|_{\max}}{\sqrt{K}}\left(1 + \sqrt{2\log\frac{1}{\delta}}\right).\end{aligned}$$

with probability at least $1 - \delta$. Using this within the bound (C.12), we obtain that

$$\begin{aligned}v(P_J^{\text{feas}}) &\leq v(P_{\text{distr}}^{\text{feas}}) + \|\mathbf{A}\mathbf{x}' - \mathbf{A}\mathbf{x}^{*0}\|_1 \\ &\leq v(P_{\text{distr}}^{\text{feas}}) + \frac{C}{\sqrt{K}} \cdot m \cdot \|\mathbf{A}\|_{\max} \cdot \left(1 + \sqrt{2\log\frac{1}{\delta}}\right)\end{aligned}$$

holds with probability at least $1 - \delta$, which completes the proof. □

C.1.3 Proof of Theorem 14

We prove the result by showing that the bound U^{covering} is a valid bound on $\|\mathbf{p}\|_\infty$ for any feasible solution of the dual D_J^{covering}, no matter what the sample of columns J is, and then

invoking Proposition 12. Fix an $i \in [m]$, and consider the LP

$$D_J^{\text{B-covering}} \ : \ \max\{p_i \mid \mathbf{p}^T \mathbf{A}_J \leq \mathbf{c}_J^T, \ \mathbf{p} \geq \mathbf{0}\}. \tag{C.13}$$

The optimal objective value of this problem, $v(D_J^{\text{B-covering}})$, is an upper bound on p_i for any feasible solution $\mathbf{p}$ of D_J^{covering} (and thus, it is an upper bound on p_i for any optimal solution $\mathbf{p}$ of D_J^{covering}). Consider the dual of this problem:

$$P_J^{\text{B-covering}} \ : \ \min\{\mathbf{c}_J^T \tilde{\mathbf{x}} \mid \mathbf{A}_J \tilde{\mathbf{x}} \geq \mathbf{e}_i, \ \tilde{\mathbf{x}} \geq \mathbf{0}\}, \tag{C.14}$$

where $\mathbf{e}_i$ is the ith standard basis vector for $\mathbb{R}^m$. By weak duality, the objective value of any feasible solution of $P_J^{\text{B-covering}}$ is an upper bound on $v(D_J^{\text{B-covering}})$.

We now construct a particular feasible solution. Let j' be any column in J such that $A_{i,j'} > 0$; such a column is guaranteed to exist by our assumption on the matrix $\mathbf{A}$. Define a solution $\tilde{\mathbf{x}}$ as

$$\tilde{x}_j = \begin{cases} 1/A_{i,j'} & \text{if } j = j', \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that $\tilde{\mathbf{x}}$ is a feasible solution of $P_J^{\text{B-covering}}$, and that its objective value is $\mathbf{c}_J^T \tilde{\mathbf{x}} = c_{j'}/A_{i,j'}$. Since this objective value is bounded by U^{covering}, it follows that $U^{\text{covering}} \geq \max\{p_i \mid \mathbf{p}^T \mathbf{A}_J \leq \mathbf{c}_J^T, \ \mathbf{p} \geq \mathbf{0}\}$.

Since our choice of i was arbitrary, it follows that $\|\mathbf{p}\|_\infty \leq U^{\text{covering}}$ for any feasible solution of D_J^{covering}. The result then follows by invoking Proposition 12. □

C.1.4 Proof of Theorem 15

As with Theorem 14, we will prove the result by showing that U^{packing} is a valid upper bound on $\|\mathbf{p}\|_\infty$ for any optimal solution of the dual problem D_J^{packing}, no matter what J is, and then invoking Proposition 13.

We first establish a useful property of W: the quantity W is actually an upper bound on

$v(P)$. To see this, define the solution $\tilde{\mathbf{x}}^{(i)}$ for each i as

$$\tilde{\mathbf{x}}^{(i)} = \frac{b_i}{A_{i,j_i^*}} \cdot \mathbf{e}_{j_i^*},$$

and define $\tilde{\mathbf{x}} = \sum_{i=1}^{m} \tilde{\mathbf{x}}^{(i)}$. Let $\mathbf{x}$ be any feasible solution of the complete problem P^{packing}. Note that for each $\tilde{\mathbf{x}}^{(i)}$, we have:

$$\begin{aligned}
\mathbf{c}^T \tilde{\mathbf{x}}^{(i)} &= \frac{c_{j_i^*} b_i}{A_{i,j_i^*}} \\
&\geq \frac{c_{j_i^*}}{A_{i,j_i^*}} \left[\sum_{j=1}^{n} A_{i,j} x_j \right] \\
&= \frac{c_{j_i^*}}{A_{i,j_i^*}} \left[\sum_{j : A_{i,j} > 0} A_{i,j} x_j \right] \\
&\geq \sum_{j : A_{i,j} > 0} A_{i,j} \cdot \frac{c_j}{A_{i,j}} \cdot x_j \\
&= \sum_{j : A_{i,j} > 0} c_j x_j.
\end{aligned}$$

where the first inequality follows because $\mathbf{x}$ satisfies $\mathbf{A}\mathbf{x} \leq \mathbf{b}$, and the second follows by the definition of j_i^*. Using this bound, we have

$$\begin{aligned}
\mathbf{c}^T \tilde{\mathbf{x}} &= \sum_{i=1}^{m} \mathbf{c}^T \tilde{\mathbf{x}}^{(i)} \\
&\geq \sum_{i=1}^{m} \left[\sum_{j : A_{i,j} > 0} c_j x_j \right] \\
&\geq \sum_{j=1}^{n} c_j x_j \\
&= \mathbf{c}^T \mathbf{x},
\end{aligned}$$

where the second inequality follows by our assumption that for each j, there exists an i such that $A_{i,j} > 0$.

Now, let us fix an $i \in [m]$. We wish to bound $|p_i|$ for an optimal solution $\mathbf{p}$ of D_J^{packing}. We can compute a bound on $|p_i|$ by solving the following LP:

$$D_J^{\text{B-packing}} \ : \ \max\{p_i \mid \mathbf{p}^T \mathbf{b} \leq v(P_J^{\text{packing}}),\ \mathbf{p}^T \mathbf{A}_J \geq \mathbf{c}_J^T,\ \mathbf{p} \geq \mathbf{0}\}.$$

Note that by weak duality, the feasible region of $D_J^{\text{B-covering}}$ is exactly the set of all optimal solutions to the sampled dual problem, D_J^{packing}. Observe that for any J, $v(P_J^{\text{packing}}) \leq v(P^{\text{packing}}) \leq W$. Thus, a valid upper bound on $v(D_J^{\text{B-packing}})$ can be obtained by solving the following relaxation of $D_J^{\text{B-packing}}$:

$$D_J^{\text{B-packing-rlx}} \ : \ \max\{p_i \mid \mathbf{p}^T\mathbf{b} \leq W, \ \mathbf{p} \geq \mathbf{0}\}.$$

This problem is a valid relaxation, because we have simply removed the constraint $\mathbf{p}^T\mathbf{A}_J \geq \mathbf{c}_J^T$, and we have replaced the value $v(P_J^{\text{packing}})$ with the larger value of W. The optimal objective value of this relaxation is simply W/b_i. Therefore, we obtain that for any dual optimal solution $\mathbf{p}$ of D_J^{packing}, $|p_i| \leq W/b_i$. It follows that $\|\mathbf{p}\|_\infty \leq \max_{i\in[m]}(W/b_i) \equiv U^{\text{packing}}$, for any optimal solution $\mathbf{p}$ of D_J^{packing}. Invoking Proposition 13 with this bound gives the desired result. □

C.1.5 Proof of Proposition 14

Let $(\mathbf{x}^{*0}, \mathbf{r}^{*0})$ be an optimal solution of $P_{\text{distr}}^{\text{portfolio}}$. Consider the solution $(\mathbf{x}', \mathbf{r}')$ defined relative to the sample J:

$$\mathbf{x}' = \frac{1}{K}\sum_{k=1}^{K} \frac{x_{j_k}^{*0}}{\xi_{j_k}} \mathbf{e}_{j_k}, \tag{C.15}$$

$$\mathbf{r}' = \sum_{j\in[n]} \boldsymbol{\alpha}_j x_j' = \frac{1}{K}\sum_{k=1}^{K} (x_{j_k}^{*0}/\xi_{j_k})\boldsymbol{\alpha}_{j_k}. \tag{C.16}$$

The significance of $(\mathbf{x}', \mathbf{r}')$ is that we will be able to show that $\mathbf{r}'$ will be close to $\mathbf{r}^{*0}$, and that $f(\mathbf{r}')$ will be close to $f(\mathbf{r}^{*0}) = F_{\text{distr}}$. However, $(\mathbf{x}', \mathbf{r}')$ is not necessarily a feasible solution to problem $P^{\text{portfolio}}$, because $\mathbf{x}'$ will in general not satisfy the unit sum constraint. To turn it into a feasible solution for problem $P^{\text{portfolio}}$, we consider the solution $(\mathbf{x}'', \mathbf{r}'')$ obtained by normalizing $\mathbf{x}'$ by its sum:

$$\mathbf{x}'' = \frac{\mathbf{x}'}{\mathbf{1}^T\mathbf{x}'}, \tag{C.17}$$

$$\mathbf{r}'' = \frac{\mathbf{r}'}{\mathbf{1}^T\mathbf{x}'}. \tag{C.18}$$

Note that $(\mathbf{x}'', \mathbf{r}'')$ is a feasible solution of $P_J^{\text{portfolio}}$.

To understand why we consider $(\mathbf{x}', \mathbf{r}')$ and $(\mathbf{x}'', \mathbf{r}'')$, we show how these two solutions can be used to bound the difference between F_J and F_{distr}. Let $(\mathbf{x}, \mathbf{r})$ be an optimal solution of $P_J^{\text{portfolio}}$. We now bound $F_J - F_{\text{distr}}$ as follows:

$$\begin{aligned} F_J - F_{\text{distr}} &= f(\mathbf{r}) - f(\mathbf{r}^{*0}) \\ &\leq f(\mathbf{r}'') - f(\mathbf{r}^{*0}) \\ &= f(\mathbf{r}'') - f(\mathbf{r}') + f(\mathbf{r}') - f(\mathbf{r}^{*0}) \\ &\leq |f(\mathbf{r}'') - f(\mathbf{r}')| + |f(\mathbf{r}') - f(\mathbf{r}^{*0})| \\ &\leq L\|\mathbf{r}'' - \mathbf{r}'\|_2 + L\|\mathbf{r}' - \mathbf{r}^{*0}\|_2 \end{aligned} \tag{C.19}$$

where the first step follows by the definitions of $(\mathbf{x}, \mathbf{r})$ and $(\mathbf{x}^{*0}, \mathbf{r}^{*0})$; the second step follows because $(\mathbf{x}'', \mathbf{r}'')$ is a feasible solution of $P_J^{\text{portfolio}}$; the third and fourth step follow by algebra and basic properties of absolute values; and the last step follows by the fact that f is Lipschitz continuous with constant L.

We now proceed to show that $\|\mathbf{r}' - \mathbf{r}^{*0}\|_2$ and $\|\mathbf{r}'' - \mathbf{r}'\|_2$ can be bounded with high probability.

Bounding $\|\mathbf{r}' - \mathbf{r}^{*0}\|_2$: To bound this term, let us define for each $k \in [K]$ the random vector $\mathbf{r}_{j_k}$ as

$$\mathbf{r}_{j_k} = \frac{x_{j_k}^{*0}}{\xi_{j_k}} \boldsymbol{\alpha}_{j_k}.$$

We make three important observations about $\mathbf{r}_{j_1}, \ldots, \mathbf{r}_{j_K}$. First, for each k, the norm of $\mathbf{r}_{j_k}$ is bounded as

$$\|\mathbf{r}_{j_k}\|_2 = \left\| \frac{x_{j_k}^{*0}}{\xi_{j_k}} \boldsymbol{\alpha}_{j_k} \right\|_2 \leq \frac{x_{j_k}^{*0}}{\xi_{j_k}} \cdot \|\boldsymbol{\alpha}_{j_k}\|_2 \leq \frac{C\xi_{j_k}}{\xi_{j_k}} \cdot H = CH.$$

Second, observe that $\mathbf{r}'$ is just the sample mean of $\mathbf{r}_{j_1}, \ldots, \mathbf{r}_{j_K}$, i.e., $\mathbf{r}' = (1/K) \sum_{k=1}^{K} \mathbf{r}_{j_k}$.

Lastly, we observe that the expected value of each $\mathbf{r}_{j_k}$ is

$$\mathbb{E}[\mathbf{r}_{j_k}] = \sum_{j \in [n]: \xi_j > 0} \xi_j \cdot \frac{x_j^{*0}}{\xi_j} \boldsymbol{\alpha}_j = \sum_{j \in [n]: \xi_j > 0} x_j^{*0} \boldsymbol{\alpha}_j = \sum_{j \in [n]} x_j^{*0} \boldsymbol{\alpha}_j = \mathbf{r}^{*0},$$

where the third step uses the fact that $x_j^{*0} = 0$ when $\xi_j = 0$ (by virtue of the constraint $\mathbf{0} \leq \mathbf{x} \leq C\boldsymbol{\xi}$). Therefore, the term $\|\mathbf{r}' - \mathbf{r}^{*0}\|_2$ is just the distance between the sample mean of an i.i.d. collection of random vectors from its expected value, where the ℓ_2 norm of each random vector is bounded. We can therefore invoke Lemma 12 to assert that

$$\|\mathbf{r}' - \mathbf{r}^{*0}\|_2 \leq \frac{CH}{\sqrt{K}} \left(1 + \sqrt{2 \log \frac{2}{\delta}}\right) \tag{C.20}$$

with probability at least $1 - \delta/2$.

Bounding $\|\mathbf{r}'' - \mathbf{r}'\|_2$: For this term, observe first that since $\mathbf{r}'' = \mathbf{r}'/(\mathbf{1}^T\mathbf{x}')$, we can re-arrange this to obtain that $\mathbf{r}' = (\mathbf{1}^T\mathbf{x}')\mathbf{r}''$. Let us use s to denote the normalization constant, i.e., $s = \mathbf{1}^T\mathbf{x}'$. We can now bound $\|\mathbf{r}'' - \mathbf{r}'\|_2$ in the following way:

$$\|\mathbf{r}'' - \mathbf{r}'\|_2 = \|\mathbf{r}'' - s\mathbf{r}''\|_2 = |s - 1| \cdot \|\mathbf{r}''\|_2 .$$

We now bound $|s - 1|$. Note that s can be written as

$$s = \mathbf{1}^T\mathbf{x}' = \frac{1}{K} \sum_{k=1}^{K} \frac{x_{j_k}^{*0}}{\xi_{j_k}} \mathbf{1}^T \mathbf{e}_{j_k} = \frac{1}{K} \sum_{k=1}^{K} \frac{x_{j_k}^{*0}}{\xi_{j_k}}.$$

Letting $w_k = (x_{j_k}^{*0}/\xi_{j_k})$, we obtain $s = (1/K)\sum_{k=1}^{K} w_k$; in other words, s is the average of K i.i.d. random variables, $w_1, \ldots, w_K$. Note that each w_k has expected value $\mathbb{E}[w_k] = \sum_{j \in [n]: \xi_j > 0} (x_j^{*0}/\xi_j) \cdot \xi_j = \sum_{j \in [n]} x_j^{*0} = 1$; therefore, the term $|s - 1|$ represents how much the sample mean s deviates from its expected value of 1. We also observe that each w_k is contained in the interval $[0, C]$. Therefore, using Hoeffding's inequality, we obtain that

$$\mathbf{Pr}[|s - 1| > \epsilon] = \mathbf{Pr}[|s - \mathbb{E}[s]| > \epsilon] \leq 2 \cdot \exp\left(-\frac{2K\epsilon^2}{C^2}\right), \tag{C.21}$$

for any $\epsilon > 0$; by setting $\epsilon = C\sqrt{\log(4/\delta)/(2K)}$, we obtain that

$$|s - 1| \leq C\sqrt{\frac{1}{2K}\log\frac{4}{\delta}}, \tag{C.22}$$

with probability at least $1 - \delta/2$.

With this bound in hand, let us now bound $\|\mathbf{r}''\|_2$. Observe that

$$\|\mathbf{r}'\|_2 \leq \frac{1}{K} \cdot \sum_{k=1}^{K} \left(\frac{x^{*0}_{j_k}}{\xi_{j_k}}\right) \cdot \|\boldsymbol{\alpha}_{j_k}\|_2 \leq \frac{1}{K} \cdot \sum_{k=1}^{K} \left(\frac{x^{*0}_{j_k}}{\xi_{j_k}}\right) \cdot H = s \cdot H,$$

so it follows that $\|\mathbf{r}''\|_2 = (1/s)\|\mathbf{r}'\|_2 \leq H$. We therefore have that $\|\mathbf{r}'' - \mathbf{r}'\|_2$ satisfies

$$\|\mathbf{r}'' - \mathbf{r}'\|_2 \leq \frac{CH}{\sqrt{K}}\sqrt{\frac{1}{2}\log\frac{4}{\delta}},$$

with probability at least $1 - \delta/2$.

Completing the proof: We now put these two bounds together to complete the bound in (C.19). Combining inequalities (C.1.5) and (C.20) together using the union bound, we have that with probability at least $1 - \delta$,

$$\begin{aligned} F_J - F_{\text{distr}} &\leq L\|\mathbf{r}'' - \mathbf{r}'\|_2 + L\|\mathbf{r}' - \mathbf{r}^{*0}\|_2 \\ &\leq L \cdot \frac{CH}{\sqrt{K}}\sqrt{\frac{1}{2}\log\frac{4}{\delta}} + L \cdot \frac{CH}{\sqrt{K}}\left(1 + \sqrt{2\log\frac{2}{\delta}}\right) \\ &\leq \frac{CHL}{\sqrt{K}}\left(1 + 3\sqrt{\log\frac{4}{\delta}}\right). \end{aligned}$$

By moving F_{distr} to the right hand side, and subtracting F from both sides, we obtain the desired inequality. □

C.1.6 Proof of Theorem 16

Before we can prove Theorem 16, we need to establish two auxiliary results. The first result is the analog of Lemma 12 for a collection of possibly dependent random variables, formulated in terms of forest complexity.

Lemma 15 *Let $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$ be K random vectors with same distribution. Let G be the dependency graph of $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$. In addition, assume $\|\mathbf{w}_k\|_2 \leq C$ for $k = 1, \ldots, K$. Let $\bar{\mathbf{w}} = (1/K) \cdot \sum_{k=1}^{K} \mathbf{w}_k$. Then for any $\delta \in (0, 1)$, we have, with probability at least $1 - \delta$,*

$$\|\bar{\mathbf{w}} - \mathbb{E}\bar{\mathbf{w}}\|_2 \leq C \cdot \left(\sqrt{\frac{K + 2 \cdot |E(G)|}{K^2}} + \sqrt{\frac{2 \cdot \Lambda(G)}{K^2} \cdot \log \frac{1}{\delta}} \right).$$

Proof of Lemma 15: Define a space $\mathcal{W} \equiv \{\mathbf{z} \mid \|\mathbf{z}\|_2 \leq C\}$. Consider a scalar function $f : \mathcal{W}^K \to \mathbb{R}$ defined as

$$f(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K) = \left\| \frac{1}{K} (\mathbf{z}_1 + \mathbf{z}_2 + \ldots + \mathbf{z}_K) - \mathbb{E}\bar{\mathbf{w}} \right\|_2$$

For any $k \in [K]$ and any $\mathbf{z}_1, \ldots, \mathbf{z}_k, \ldots, \mathbf{z}_K, \mathbf{z}'_k \in \mathcal{W}$, we have

$$|f(\mathbf{z}_1, \ldots, \mathbf{z}_k, \ldots, \mathbf{z}_K) - f(\mathbf{z}_1, \ldots, \mathbf{z}'_k, \ldots, \mathbf{z}_K)| \leq \frac{\|\mathbf{z}_k - \mathbf{z}'_k\|}{K} \leq \frac{2C}{K}.$$

Therefore, f has the bounded differences property (note that in [79], this is referred to as the **c**-Lipschitz property; see Definition 2.1 of that paper). By Theorem 3.6 of [79], for any $\epsilon > 0$, we have

$$\mathbf{Pr}\left[f(\mathbf{w}_1, \ldots, \mathbf{w}_K) - \mathbb{E}f(\mathbf{w}_1, \ldots, \mathbf{w}_K) \geq \epsilon\right] \leq \exp\left(-\frac{K^2\epsilon^2}{2C^2 \cdot \Lambda(G)}\right)$$

On the other hand, define $\mathbf{u}_i = \mathbf{w}_i - \mathbb{E}\mathbf{w}_i$. Then

$$\mathbb{E}\left[\mathbf{u}_i^T \mathbf{u}_j\right] = \begin{cases} \mathbb{E}\left[\mathbf{w}_i^T \mathbf{w}_j\right] - \|\mathbb{E}\mathbf{w}_i\|_2^2 \leq \mathbb{E}\left[\|\mathbf{w}_i\|_2 \|\mathbf{w}_j\|_2\right] \leq C^2, & \text{if } i = j \text{ or } \langle i, j\rangle \in E(G), \\ 0, & \text{otherwise.} \end{cases}$$

Therefore,

$$\begin{aligned} \mathbb{E}\left[f(\mathbf{w}_1, \ldots, \mathbf{w}_K)^2\right] &= \left\| \frac{1}{K} (\mathbf{w}_1 + \ldots + \mathbf{w}_K) - \mathbb{E}\bar{\mathbf{w}} \right\|_2^2 \\ &= \frac{1}{K^2} \left(\sum_{i,j \in [K]} \mathbb{E}\left[\mathbf{u}_i^T \mathbf{u}_j\right] \right) \\ &= \frac{1}{K^2} \left(\sum_{i \in [K]} \mathbb{E}\left[\mathbf{u}_i^T \mathbf{u}_i\right] + \sum_{\langle i,j\rangle \in E(G)} \mathbb{E}\left[\mathbf{u}_i^T \mathbf{u}_j\right] \right) \\ &\leq C^2 \cdot \frac{K + 2|E(G)|}{K^2}. \end{aligned}$$

As a result,

$$\mathbb{E}f(\mathbf{w}_1, \ldots, \mathbf{w}_K) \leq \sqrt{\mathbb{E}f(\mathbf{w}_1, \ldots, \mathbf{w}_K)^2} \leq C \cdot \sqrt{\frac{K + 2|E(G)|}{K^2}},$$

where the first inequality comes from the concavity of square root function. With all the results above, we have

$$\begin{aligned}\mathbf{P}\left[f(\mathbf{w}_1, \ldots, \mathbf{w}_K) - C \cdot \sqrt{\frac{K + 2|E(G)|}{K^2}} \geq \epsilon\right] &\leq \mathbf{P}\left[f(\mathbf{w}_1, \ldots, \mathbf{w}_K) - \mathbb{E}f(\mathbf{w}_1, \ldots, \mathbf{w}_K) \geq \epsilon\right] \\ &\leq \exp\left(-\frac{K^2\epsilon^2}{2C^2 \cdot \Lambda(G)}\right)\end{aligned}$$

Let $\epsilon = \sqrt{2C^2\Lambda(G)\log(1/\delta)/K^2}$. Then with probability at least $1 - \delta$, we have

$$f(\mathbf{w}_1, \ldots, \mathbf{w}_K) \leq C \cdot \sqrt{\frac{K + 2|E(G)|}{K^2}} + C\sqrt{\frac{2 \cdot \Lambda(G)}{K^2}\log\left(\frac{1}{\delta}\right)}.$$

We thus prove the statement. □

From Lemma 15, we can also straightforwardly prove the following result, which is the analog of Lemma 13 for possibly dependent random variables.

Corollary 1 *Let $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$ be K random vectors of size m and with same distribution. Let G be the dependency graph of $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$. In addition, assume $\|\mathbf{w}_k\|_\infty \leq C$ for $k = 1, \ldots, K$. Let $\bar{\mathbf{w}} = (1/K) \cdot \sum_{k=1}^K \mathbf{w}_k$. Then for any $\delta \in (0, 1)$, we have, with probability at least $1 - \delta$,*

$$\|\bar{\mathbf{w}} - E\bar{\mathbf{w}}\|_1 \leq \sqrt{m} \cdot C \cdot \left(\sqrt{\frac{K + 2 \cdot |E(G)|}{K^2}} + \sqrt{\frac{2 \cdot \Lambda(G)}{K^2} \cdot \log\frac{1}{\delta}}\right).$$

With these two results, we can now proceed with proving Theorem 16. We define $\mathbf{x}^{*0}$ and construct random vectors $\mathbf{w}_{j_1}, \ldots, \mathbf{w}_{j_K}$, $\mathbf{b}_{j_1}, \ldots, \mathbf{b}_{j_K}$ as in the proof of Proposition 17; we note that this construction is valid even if there exists dependency between the indices j_1,

..., and j_K. We further define $\mathbf{x}'$ as the sample mean of $\mathbf{w}_{j_1}, \ldots, \mathbf{w}_{j_K}$ and $\mathbf{b}'$ as the sample mean of $\mathbf{b}_{j_1}, \ldots, \mathbf{b}_{j_K}$. By Proposition 17 and Expression (C.9), we have

$$\Delta v(P_J) \leq \Delta v(P_{\text{distr}}) + \|\mathbf{x}' - \mathbf{x}^{*0}\|_2 + \|\mathbf{p}_J^*\|_\infty \cdot \|\mathbf{b}' - \mathbf{b}\|_1. \tag{C.23}$$

By invoking Lemma 15, with probability at least $1 - \delta$,

$$\|\mathbf{x}' - \mathbf{x}^{*0}\|_2 \leq C \cdot \left(\sqrt{\frac{K + 2 \cdot |E(G)|}{K^2}} + \sqrt{\frac{2 \cdot \Lambda(G)}{K^2} \cdot \log \frac{1}{\delta}} \right). \tag{C.24}$$

Similarly, by Corollary 1, with probability at least $1 - \delta$,

$$\|\mathbf{b}' - \mathbf{b}\|_1 \leq \sqrt{m} \cdot C \cdot \|\mathbf{A}\|_{\max} \cdot \left(\sqrt{\frac{K + 2 \cdot |E(G)|}{K^2}} + \sqrt{\frac{2 \cdot \Lambda(G)}{K^2} \cdot \log \frac{1}{\delta}} \right). \tag{C.25}$$

Combining inequalities (C.23), (C.24), and (C.25) and applying the union bound, we conclude that, with probability at least $1-\delta$, the following holds: if P_J is feasible and $\text{rank}(\mathbf{A}_J) = m$,

$$\Delta v(P_J) \leq \Delta v(P_{\text{distr}}) + C \cdot (1 + m\gamma \|\mathbf{A}\|_{\max}) \cdot \left(\sqrt{\frac{K + 2|E(G)|}{K^2}} + \sqrt{\frac{2\Lambda(G)\log(2/\delta)}{K^2}} \right). \tag{C.26}$$

Similarly, by Proposition 17 and inequality (C.11), we have

$$\Delta v(P_J) \leq \Delta v(P_{\text{distr}}) + \chi \cdot \|\mathbf{x}' - \mathbf{x}^{*0}\|_2. \tag{C.27}$$

Combining with inequality (C.24), we conclude that, with probability $1 - \delta$, the following holds: if P_J is feasible and $\text{rank}(\mathbf{A}_J) = m$,

$$\Delta v(P_J) \leq \Delta v(P_{\text{distr}}) + C \cdot \chi \cdot \left(\sqrt{\frac{K + 2|E(G)|}{K^2}} + \sqrt{\frac{2\Lambda(G)\log(1/\delta)}{K^2}} \right), \tag{C.28}$$

which completes the proof. □

REFERENCES

[1] S. Agrawal, Z. Wang, and Y. Ye. A dynamic near-optimal algorithm for online linear programming. *Operations Research*, 62(4):876–890, 2014.

[2] L. Alfandari, A. Hassanzadeh, and I. Ljubić. An exact method for assortment optimization under the nested logit model. *European Journal of Operational Research*, 291(3):830–845, 2021.

[3] A. Alptekinoğlu and J. H. Semple. The exponomial choice model: A new alternative for assortment and price optimization. *Operations Research*, 64(1):79–93, 2016.

[4] A. Aouad, V. Farias, R. Levi, and D. Segev. The approximability of assortment optimization under ranking preferences. *Operations Research*, 66(6):1661–1669, 2018.

[5] A. Aouad, V. F. Farias, and R. Levi. Assortment optimization under consider-then-choose choice models. *Available at SSRN 2618823*, 2015.

[6] A. Aouad, J. Feldman, and D. Segev. The exponomial choice model: Algorithmic frameworks for assortment optimization and data-driven estimation case studies. *Available at SSRN 3192068*, 2018.

[7] D. Ariely. *Predictably Irrational*. HarperCollins New York, 2008.

[8] A. Belloni, R. Freund, M. Selove, and D. Simester. Optimizing product line designs: Efficient methods and comparisons. *Management Science*, 54(9):1544–1552, 2008.

[9] M. Ben-Akiva and S. Lerman. *Discrete Choice Analysis: Theory and Application to Travel Demand*. MIT press, 1985.

[10] G. Berbeglia. The generalized stochastic preference choice model. *arXiv preprint arXiv:1803.04244*, 2018.

[11] D. P. Bertsekas. *Network optimization: continuous and discrete models*. 1998.

[12] D. Bertsimas, A. Chang, V. V. Mišić, and N. Mundru. The airlift planning problem. *Transportation Science*, 53(3):773–795, 2019.

[13] D. Bertsimas and V. V. Mišić. Exact first-choice product line optimization. *Operations Research*, 67(3):651–670, 2019.

[14] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific, Belmont, MA, USA., 1997.

[15] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *Journal of the ACM (JACM)*, 51(4):540–556, 2004.

[16] D. Bertsimas and R. Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas, 2005.

[17] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[18] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

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

[20] H. D. Block and J. Marschak. Random orderings and stochastic theories of response. Technical report, Cowles Foundation for Research in Economics, Yale University, 1959.

[21] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[22] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[23] B. J. Bronnenberg, M. W. Kruger, and C. F. Mela. Database paper—the IRI marketing data set. *Marketing science*, 27(4):745–748, 2008.

[24] B. J. Bronnenberg and C. F. Mela. Market roll-out and retailer adoption for new brands. *Marketing Science*, 23(4):500–518, 2004.

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

[26] G. Calafiore and M. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2005.

[27] G. C. Calafiore and M. C. Campi. The scenario approach to robust control design. *IEEE Transactions on automatic control*, 51(5):742–753, 2006.

[28] M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008.

[29] M. C. Campi and S. Garatti. Wait-and-judge scenario optimization. *Mathematical Programming*, 167(1):155–189, 2018.

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

[31] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[32] Y.-C. Chen and V. V. Mišić. Column-randomized linear programs: Performance guarantees and applications. *arXiv preprint arXiv:2007.10461*, 2020.

[33] Y.-C. Chen and V. V. Mišić. Assortment optimization under the decision forest model. *arXiv preprint arXiv:2103.14067*, 2021.

[34] Y.-C. Chen and V. V. Mišić. Decision forest: A nonparametric approach to modeling irrational choice. *Management Science*, 2022.

[35] I. Contreras, J.-F. Cordeau, and G. Laporte. Benders decomposition for large-scale uncapacitated hub location. *Operations Research*, 59(6):1477–1490, 2011.

[36] J.-F. Cordeau, F. Furini, and I. Ljubić. Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research*, 275(3):882–896, 2019.

[37] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

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

[39] D. P. De Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of operations research*, 29(3):462–478, 2004.

[40] A. Désir, V. Goyal, D. Segev, and C. Ye. Capacity constrained assortment optimization under the markov chain-based choice model. *Operations Research*, 2015. Forthcoming.

[41] A. Désir, V. Goyal, D. Segev, and C. Ye. Constrained assortment optimization under the markov chain–based choice model. *Management Science*, 66(2):698–721, 2020.

[42] A. Désir, V. Goyal, H. Topalogu, and J. Zhang. Robust assortment optimization under the markov chain model. Technical report, Working Paper, 2015.

[43] J. Desrosiers and M. E. Lübbecke. A primer in column generation. pages 1–32, 2005.

[44] S. Dogan and K. Yildiz. Choice through a unified lens: The prudential model. *Available at SSRN 3085542*, 2018.

[45] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

[46] Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European journal of operational research*, 54(1):7–22, 1991.

[47] I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[48] F. Echenique and K. Saito. General Luce model. *Economic Theory*, pages 1–16, 2015.

[49] F. Echenique, K. Saito, and G. Tserenjigmid. The perception-adjusted Luce model. *Mathematical Social Sciences*, 93:67–76, 2018.

[50] R. Eghbali, J. Saunderson, and M. Fazel. Competitive online algorithms for resource allocation over the positive semidefinite cone. *Mathematical Programming*, 170(1):267–292, 2018.

[51] A. N. Elmachtoub and P. Grigas. Smart "predict, then optimize". *arXiv preprint arXiv:1710.08005*, 2017.

[52] V. F. Farias, S. Jagabathula, and D. Shah. A nonparametric approach to modeling choice with limited data. *Management science*, 59(2):305–322, 2013.

[53] D. Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4or*, 8(4):407–424, 2010.

[54] J. Feldman, A. Paul, and H. Topaloglu. Assortment optimization with small consideration sets. *Operations Research*, 2018. Forthcoming.

[55] J. Feldman, A. Paul, and H. Topaloglu. Assortment optimization with small consideration sets. *Operations Research*, 67(5):1283–1299, 2019.

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

[57] M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning benders decomposition for large-scale facility location. *Management Science*, 63(7):2146–2162, 2017.

[58] A. Flores, G. Berbeglia, and P. Van Hentenryck. Assortment and price optimization under the two-stage luce model. *arXiv preprint arXiv:1706.08599*, 2017.

[59] L. R. Ford Jr and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101, 1958.

[60] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156. Morgan Kauffman, San Francisco, CA, 1996.

[61] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[62] G. Gallego, R. Ratliff, and S. Shebalov. A general attraction model and sales-based linear program for network revenue management under customer choice. *Operations Research*, 63(1):212–232, 2014.

[63] G. Gallego and H. Topaloglu. *Assortment Optimization*, pages 129–160. Springer New York, New York, NY, 2019.

[64] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. Freeman San Francisco, 1979.

[65] M. R. Garey and D. S. Johnson. *Computers and intractability*. W. H. Freeman New York, 1979.

[66] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[67] P. E. Green and A. M. Krieger. Conjoint analysis with product-positioning applications. *Handbooks in operations research and management science*, 5:467–515, 1993.

[68] J. R. Hauser. Consideration-set heuristics. *Journal of Business Research*, 67(8):1688–1699, 2014.

[69] S. Horan. Choice by tournament. Technical report, Working paper, Boston University., 2011.

[70] J. Huber, J. W. Payne, and C. Puto. Adding asymmetrically dominated alternatives: Violations of regularity and the similarity hypothesis. *Journal of consumer research*, 9(1):90–98, 1982.

[71] J. Huchette and J. P. Vielma. A combinatorial approach for small and strong formulations of disjunctive constraints. *Mathematics of Operations Research*, 44(3):793–820, 2019.

[72] S. Jagabathula and P. Rusmevichientong. A nonparametric joint assortment and price choice model. *Management Science*, 63(9):3128–3145, 2016.

[73] S. Jagabathula and P. Rusmevichientong. The limit of rationality in choice modeling: Formulation, computation, and implications. *Management Science*, 65(5):2196–2215, 2019.

[74] D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–292, 1979.

[75] T. Kitahara and S. Mizuno. A bound for the number of different basic solutions generated by the simplex method. *Mathematical Programming*, 137(1-2):579–586, 2013.

[76] A. Klose and A. Drexl. Lower bounds for the capacitated facility location problem based on column generation. *Management Science*, 51(11):1689–1705, 2005.

[77] J. Li and R. Tang. Every random choice rule is backwards-induction rationalizable. *Games and Economic Behavior*, 104:563–567, 2017.

[78] X. Li and Y. Ye. Online linear programming: Dual convergence, new algorithms, and regret bounds. *arXiv preprint arXiv:1909.05499*, 2019.

[79] X. Liu, Y. Wang, and L. Wang. McDiarmid-Type Inequalities for Graph-Dependent Variables and Stability Bounds. In *Advances in Neural Information Processing Systems*, pages 10889–10899, 2019.

[80] M. Lubin and I. Dunning. Computing in operations research using Julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

[81] A. S. Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

[82] R. Y. Maragheh, A. Chronopoulou, and J. M. Davis. A customer choice model with HALO effect. *arXiv preprint arXiv:1805.01603*, 2018.

[83] S. Martello and P. Toth. Algorithms for knapsack problems. In *North-Holland Mathematics Studies*, volume 132, pages 213–257. Elsevier, 1987.

[84] R. D. McBride and F. S. Zufryden. An integer programming approach to the optimal product line selection problem. *Marketing Science*, 7(2):126–140, 1988.

[85] D. McFadden and K. Train. Mixed MNL models for discrete response. *Journal of Applied Econometrics*, 15(5):447–470, 2000.

[86] I. Méndez-Díaz, J. J. Miranda-Bront, G. Vulcano, and P. Zabala. A branch-and-cut algorithm for the latent-class logit assortment problem. *Discrete Applied Mathematics*, 164:246–263, 2014.

[87] V. V. Mišić. *Data, models and decisions for large-scale stochastic optimization problems*. PhD thesis, Massachusetts Institute of Technology, 2016.

[88] V. V. Mišić. *Data, models and decisions for large-scale stochastic optimization problems*. PhD thesis, Massachusetts Institute of Technology, 2016.

[89] V. V. Mišić. Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624, 2020.

[90] P. Mohajerin Esfahani, T. Sutter, and J. Lygeros. Performance bounds for the scenario approach and an extension to a class of non-convex programs. *IEEE Transactions on Automatic Control*, 60(1):46–58, 2014.

[91] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *Advances in neural information processing systems*, pages 985–992, 2007.

[92] V. R. Nijs, S. Srinivasan, and K. Pauwels. Retail-price drivers and retailer profits. *Marketing Science*, 26(4):473–487, 2007.

[93] G. Optimization. Gurobi optimizer reference manual, 2020.

[94] S. H. Owen and M. S. Daskin. Strategic facility location: A review. *European journal of operational research*, 111(3):423–447, 1998.

[95] M. Pilanci and M. J. Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Transactions on Information Theory*, 61(9):5096–5115, 2015.

[96] W. Poundstone. *Priceless: The myth of fair value (and how to take advantage of it)*. Hill and Wang, 2010.

[97] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[98] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Francisco, CA, 1993.

[99] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[100] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.

[101] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[102] J. Rieskamp, J. R. Busemeyer, and B. A. Mellers. Extending the bounds of rationality: Evidence and theories of preferential choice. *Journal of Economic Literature*, 44(3):631–661, 2006.

[103] L. Rokach. Decision forest: Twenty years of research. *Information Fusion*, 27:111–125, 2016.

[104] R. P. Rooderkerk, H. J. Van Heerde, and T. H. A. Bijmolt. Incorporating context effects into a choice model. *Journal of Marketing Research*, 48(4):767–780, 2011.

[105] C. Schön. On the optimal product line selection problem with price discrimination. *Management Science*, 56(5):896–902, 2010.

[106] A. Şen, A. Atamtürk, and P. Kaminsky. A conic integer optimization approach to the constrained assortment problem under the mixed multinomial logit model. *Operations Research*, 66(4):994–1003, 2018.

[107] S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

[108] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[109] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2014.

[110] I. Simonson. Choice based on reasons: The case of attraction and compromise effects. *Journal of consumer research*, 16(2):158–174, 1989.

[111] A. S. Şimşek and H. Topaloglu. An expectation-maximization algorithm to estimate the parameters of the markov chain choice model. *Operations Research*, 66(3):748–760, 2018.

[112] M. Sumida, G. Gallego, P. Rusmevichientong, H. Topaloglu, and J. Davis. Revenue-utility tradeoff in assortment optimization under the multinomial logit model with totally unimodular constraints. *Management Science*, 2020.

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

[114] K. T. Talluri and G. J. van Ryzin. *The theory and practice of revenue management*, volume 68. Springer Science & Business Media, 2006.

[115] K. E. Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.

[116] A. Tversky. Intransitivity of preferences. *Psychological review*, 76(1):31, 1969.

[117] A. Tversky. Elimination by aspects: A theory of choice. *Psychological review*, 79(4):281, 1972.

[118] A. Tversky and I. Simonson. Context-dependent preferences. *Management science*, 39(10):1179–1189, 1993.

[119] G. van Ryzin and G. Vulcano. A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Science*, 61(2):281–300, 2014.

[120] G. van Ryzin and G. Vulcano. A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Science*, 61(2):281–300, 2015.

[121] G. van Ryzin and G. Vulcano. An expectation-maximization method to estimate a rank-based choice model of demand. *Operations Research*, 65(2):396–407, 2017.

[122] J. P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, 58(2):303–315, 2010.

[123] J. P. Vielma and G. L. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128(1-2):49–72, 2011.

[124] K. Vu, P.-L. Poirion, and L. Liberti. Random projections for linear programming. *Mathematics of Operations Research*, 43(4):1051–1071, 2018.

[125] S. Wager and S. Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

[126] Y. Xu and L. Zhou. Rationalizability of choice functions by game trees. *Journal of Economic theory*, 134(1):548–556, 2007.

[127] Y. Ye. A new complexity result on solving the markov decision problem. *Mathematics of Operations Research*, 30(3):733–749, 2005.

[128] Y. Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.