Title
Optimization Based Machine Learning Methods for Business Analytics

Author
Akcakus, Emine Irem

Publication Date
2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Optimization Based Machine Learning Methods for Business Analytics

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

by

Emine Irem Akcakus

2023

ABSTRACT OF THE DISSERTATION

Optimization Based Machine Learning Methods
for Business Analytics

by

Emine Irem Akcakus
Doctor of Philosophy in Management
University of California, Los Angeles, 2023
Professor Velibor Mišić, Chair

The growing availability of data and recent developments in optimization methods and machine learning have led to a revolution in modern business analytics. In this Ph.D. dissertation, we propose two frameworks based on mixed-integer optimization that advance business analytics in the context of two important problems: product design with market share maximization and learning optimal decision trees.

In the first problem, we aim to find a product, as defined by its attributes, that maximizes market share, which is a weighted sum of logistic probabilities when we assume each customer segment follows a logit choice model to make a purchase. At first glance, this problem appears hopeless: one must optimize an objective function that is neither convex nor concave over an exponentially-sized discrete set of attribute combinations. Surprisingly, we show that this problem can be reformulated as a mixed-integer convex program by exploiting an economic model. We further propose an exact methodology for solving this problem based on modern integer, convex, and conic optimization techniques. Using synthetic problem instances and instances derived from real conjoint data sets, we show that our methodology can solve large

problem instances to provable optimality or near-optimality within operationally feasible time frames.

In the second problem, we propose a mixed-integer program that learns optimal decision trees from data. While decision trees are among the most widely-used machine learning methods, their learning algorithms are usually based on top-down heuristics and cannot incorporate side constraints arising from real-world business operations. We show that our proposed mixed-integer formulation is theoretically stronger than other formulations in the literature by exploring its relaxation properties. We also develop a large-scale solution method based on constraint generation. Based on computational studies on real-world data sets, we show that our proposed model is significantly more tractable than alternative mixed-integer optimization models and our large-scale method based on constraint generation can further improve the solution time in several data sets.

Overall, we contribute to business analytics by proposing exact solution methods based on optimization to two significant but computationally challenging problems and developing efficient algorithms that make them more practical to use.

The dissertation of Emine Irem Akcakus is approved.

Elisa Long

Charles Corbett

Francisco Castro

Felipe Caro

Velibor Mišić, Committee Chair

University of California, Los Angeles

2023

To my parents Sevim and Mustafa Akcakus

TABLE OF CONTENTS

1 Introduction **1**

1.1 Exact Logit-Based Product Design 2

1.2 An Integer Programming Approach to Binary Decision Trees 3

2 Exact Logit-Based Product Design **6**

2.1 Literature Review 8

2.2 Model 13

2.2.1 Problem definition 13

2.2.2 Mixed-Integer Convex Programming Formulation 15

2.3 Solution Approaches 21

2.3.1 Solution Approach #1: Representation as Mixed-Integer Conic Program 21

2.3.2 Solution Approach #2: Gradient-Based Constraint Generation 25

2.4 Extensions 29

2.4.1 Extension to Expected Profit Maximization 29

2.4.2 Extension to Geometric Mean Maximization 30

2.5 Numerical Experiments 35

2.5.1 Experiments with synthetic instances 36

2.5.2 Experiments with instances based on real conjoint datasets 38

2.6 Conclusions 43

3 An Integer Programming Approach to Binary Decision Trees **46**

3.1 Literature Review 50

3.1.1 Top-down Approach and Other Heuristics 50
3.1.2 Mixed Integer Formulations 51
3.2 Model 53
3.2.1 Problem Definition 54
3.2.2 Mixed-Integer Optimization Formulation 56
3.2.3 Structural Properties 60
3.3 Comparison to Bertsimas and Dunn [16] Formulation 62
3.3.1 Bertsimas and Dunn [16] Formulation 63
3.3.2 Theoretical Comparison of Formulations 67
3.4 Constraint Generation-Based Solution Method 70
3.5 Computational Experiments 73
3.6 Conclusion 76

4 Conclusion 82

A Appendix to Chapter 2 84
A.1 Omitted proofs 84
A.1.1 Proof of Theorem 1 84
A.1.2 Proof of Theorem 2 88
A.1.3 Proof of Proposition 1 93
A.1.4 Proof of Theorem 3 95
A.1.5 Proof of Proposition 2 96
A.1.6 Proof of Theorem 4 96
A.2 Additional details for numerical experiments 99

A.2.1 Attributes for real data instances in Section 2.5.2 99
A.2.2 Hierarchical Bayesian model specification 103
A.2.3 Additional constraints for `immigrant` dataset 103
A.2.4 Competitive offerings for Section 2.5.2 104

B Appendix to Chapter 3 108
B.1 Omitted Proofs 108
B.1.1 Proof of Proposition 3 108
B.1.2 Proof of Theorem 5 114
B.1.3 Proof of Proposition 5 120
B.1.4 Proof of Proposition 6 123
B.1.5 Proof of Proposition 7 124
B.1.6 Proof of Proposition 8 126
B.1.7 Proof of Proposition 9 126

References 128

LIST OF FIGURES

2.1 Plot of the approximation factor Γ as a function of the ratio U/L, for different values of K, with uniformly-distributed $\boldsymbol{\lambda}$. 33

3.1 A decision tree example which classifies heart attack patients as under high risk (R) or not under high risk (N). 47

3.2 Example of a tree topology $\mathcal{T}$ (left-hand subfigure) and a complete decision tree (right-hand subfigure). 56

LIST OF TABLES

2.1 Results for numerical experiment with synthetic data. 37

2.2 Summary of real conjoint datasets used in Section 2.5.2. 39

2.3 Results for numerical experiment with real data. 42

2.4 Attributes of outside options, optimal solution and heuristic solution for `candidate` LC-MNL model with $K = 20$ segments. 44

3.1 Summary of data sets used in the numerical experiments. 75

3.2 Solution time results for UCI data set (part 1 of 2). 77

3.3 Solution time results for UCI data set (part 2 of 2). 78

3.4 Optimality gap results for UCI data set (part 1 of 2). 79

3.5 Optimality gap results for UCI data set (part 2 of 2). 80

A.1 Attributes for `bank` dataset. 99

A.2 Attributes for `candidate` dataset. 100

A.3 Attributes for `immigrant` dataset. 101

A.4 Attributes for `timbuk2` dataset. 102

A.5 Outside options for `bank` dataset problem instances. 104

A.6 Outside options for `candidate` dataset problem instances. 105

A.7 Outside options for `immigrant` dataset problem instances. 106

A.8 Outside options for `timbuk2` dataset problem instances. (For ease of comparison, only one level of each binary attribute is shown.) 107

ACKNOWLEDGMENTS

I would like to express my special appreciation and gratitude to my advisor Professor Velibor Misic. Completing this dissertation and my Ph.D. degree would not be possible without his guidance and support. Velibor always set a great example to me on how to be a researcher. I also appreciate his understanding and sincerity in giving advice, not only in research, but in other career and life decisions as well. I never had any doubt that he had my best interest in mind when I was consulting with him. Velibor, I was fortunate to be your student and learned a lot from you in the past five years.

I am also grateful to Professors Felipe Caro, Francisco Castro, Charles Corbett, and Elisa Long for being in my thesis committee. Their feedback during my advancement to candidacy and final defense provided valuable insights and enabled me to think of my research from different perspectives. I would also like to thank Professor Fernanda Bravo, who was the Ph.D. liaison in my first two years at Anderson, for always checking in with us about how we were adjusting to the Ph.D. program and offering support and advice.

I feel very fortunate to have met so many great people during my time at Anderson. Unfortunately, the pandemic limited our opportunities to spend time together in person, however I am grateful for their companionship during my Ph.D. In particular, I am thankful to the fellow students in DOTM: Jian Gao, Saeed Ghodsi, Martin Gonzalez Cabello, Xinyi Guan, Jingyuan Hu, Nur Kaynar, Bobby Nyotta, Zach Siegel, Abolfazl Taghavi, Mirel Yavuz, and Jingwei Zhang.

Saeed, thank you so much for your support during my job search and sharing your experiences with me. Xinyi, I appreciate the Zoom chats we did during the lockdowns, which helped me feel less isolated. Bobby, thank you for mentoring me during my internship. Nur and Mirel, I knew I could always rely on your friendship during tough times and I never felt lonely in Los Angeles thanks to you. Zach, thank you for hosting home parties, organizing

online game nights, restaurant recommendations, and being a bouldering buddy! All of you made my past five years at Anderson more memorable.

I cannot express enough how grateful I am to my family. Throughout my Ph.D. journey, my parents, Sevim and Mustafa, have been my constant support system, providing me with encouragement and guidance. Our frequent video calls helped me stay motivated and gave me the strength to overcome any obstacle. I owe everything to their teachings and am forever grateful. I also want to express my appreciation to my sister, Didem, and brother, Selahattin, for their support in distracting me during stressful times and lifting up my spirits whenever we talked.

Finally, I would like to express my heartfelt gratitude to Yi-Chun for his unwavering love and support throughout my Ph.D. journey. His presence have been the biggest source of strength for me. I cannot thank him enough for bringing so much joy and happiness into my life, and for making Los Angeles feel like home to me. I feel incredibly blessed to have him by my side, and I look forward to starting on the next chapter of our lives together.

VITA

2016	B.S. (Industrial Engineering), Bilkent University
2018	M.S. (Industrial Engineering), Bilkent University
2018	Anderson Fellowship, UCLA Anderson School of Management
2022	Data Science Ph.D. Intern, Gopuff

PUBLICATIONS

Akçakuş, İ. and Mišić, V.V. (2022). Exact Logit-based Product Design. Under Revision in *Management Science*, arXiv preprint arXiv:2106.15084

Akçakuş, İ. and Mišić, V.V. (2023). A Novel Integer Programming Approach to Binary Decision Trees. *In Preparation*

CHAPTER 1

Introduction

Over the past decade, businesses went through a digital transformation which moved many of their operations online. The emerge of digital channels and processes made a large volume of data available. This led to the replacement of traditional business models, which relied on deriving insights from simplified presentations of real-life practices, by modern data-driven analysis. For example, traditionally, product development leaned on the firms' experience and customer insight from market research. However, the abundance of data created the need for methods that can analyze and use the data to make decisions efficiently. This resulted in the development of business analytics, which focuses on how to use data and mathematical models to make better-informed business decisions.

Machine learning and optimization has been at the heart of data-driven approaches to address the question of how to transform the data into effective decisions. As a result, the research in operations management in recent years focused heavily on employing machine learning methods to address problems in business analytics. Examples include data-driven inventory optimization through empirical risk minimization and kernel regression [11], prediction of consumer choice based on a decision-tree representation of customer segments [7], recommendation of personalized drug treatment regimens to diabetic patients by the K-nearest neighbors algorithm [18], and product personalization through matrix completion algorithms that learn from customer-product interactions [38].

While machine learning methods have become very popular data-driven approaches, they have several shortcomings. First, they mostly use suboptimal training algorithms, which

implies that such algorithms do not guarantee the best performance. Moreover, most of these training algorithms cannot incorporate additional constraints that arise from real-world scenarios. This necessitates the development of data-driven methods that are (i) provably-optimal and (ii) capable of handling side constraints.

In this dissertation, we address problems in business analytics and propose data-driven machine learning methods based on optimization, which produces provably-optimal solutions and can handle side constraints easily. In Part I, we study a product design problem, and develop solution approaches based on modern integer, convex and conic optimization. We show that our methods can scale well to larger instances of data using real-life conjoint data sets. In Part II, we develop a new mixed-integer optimization model for the design of binary decision trees, which are commonly used machine learning models in a wide range of application areas. We further provide theoretical results on the structure of the model and propose large-scale solution approaches. Finally, we show that our model outperforms the existing exact models in the literature in terms of computation time.

In the remainder of this section, we provide a high level discussion of our contributions in each chapter.

1.1 Exact Logit-Based Product Design

In this chapter[1], we consider a firm which has to design a product that maximizes the market share, i.e., number of customers that will buy the product. We define a product as a combination of its attributes, such as color and price, and we refer product design as choosing each product attribute from a finite set of options. The firm will offer its product to a collection of customers, and each customer decides to purchase the product or not according to her/his utility. This problem of finding a product that maximizes market share is referred to as the share-of-choice product design (SOCPD) problem, and has received a

[1]This chapter is based on my doctoral research work "Exact Logit-Based Product Design" [3].

significant amount of attention in the marketing science research literature.

The share-of-choice product design (SOCPD) problem is challenging for several reasons. First, the number of candidate products scales exponentially with the number of attributes, since we need to consider all the attribute combinations. Therefore, even for problems with a small number attributes, the number of candidate products can be very large. Second, if we follow the common approach in the literature and assume that each customer follows a logit choice model to make decisions, i.e., the choice probability of the product is a logit function of the utility, then we obtain an expression for the market share that is neither convex, nor concave, which is computationally hard to optimize. Third, the firm may have additional constraints, such as a budget constraint, that need to be incorporated when designing the product. Therefore, the method we will adopt should be able to handle these kinds of constraints.

In this chapter, we develop an exact methodology for solving this problem based on modern integer, convex and conic optimization. Our key result is showing that the logit-based SOCPD problem can be exactly reformulated as a mixed-integer convex program. This allows us to use cutting-edge commercial solvers to solve this problem. Using both synthetic problem instances and instances derived from real conjoint data sets, we show that our methodology can solve large instances to provable or near optimality in operationally feasible time frames. To the best of our knowledge, this is the first exact solution methodology for the logit-based SOCPD problem.

1.2 An Integer Programming Approach to Binary Decision Trees

Decision trees are graphs with a tree-like structure which map the input data to a leaf node that predicts the label of the output data. In a decision tree, for each observation, starting from the root, we proceed to the right or left node depending on the binary test applied in the current node. The label of the leaf node we reach following this process then becomes

the prediction for the corresponding observation.

Decision trees are widely used in predictive and prescriptive modeling in statistics and machine learning. The main advantage of decision trees is their interpretable nature, which allows the user to observe and evaluate the process leading to a decision. The main application areas of decision trees are classification and policy learning. For example, in the classification setting, one can use decision trees to identify high risk patient after a heart attack based on the initial 24-hour data [27]. In the policy learning setting, which describes learning the rule of decision making that matches an individual to a treatment based on the characteristics of the individual, decision trees can select offers, prices, advertisements, or emails to send to consumers, as well as in the problem of determining which medication to prescribe to a patient [89].

Despite the simplicity and popularity of decision trees, learning an optimal decision tree that maximizes the number of correctly predicted observations or other objectives from data is a notoriously difficult problem and has been proven to be NP-hard. The predominant approach for building such trees is greedy top-down induction, where one starts with a single leaf node, and iteratively adds splits to the tree. While such approaches are simple to implement, they are not able to guarantee that the resulting tree is optimal. Following recent advances in computing technology and solution software, there has been increasing interest from the operations research community in applying mixed-integer optimization techniques to decision tree learning problem.

In this chapter, we develop a new mixed-integer optimization model for the design of binary decision trees. We provide theoretical results on the structure of the model and show that the formulation is stronger than alternative formulations that have been previously proposed. To evaluate the performance of our model, we conduct computational experiments on real-life data sets from the UCI Machine Learning Repository. We present a comparison of our model and other methods in the literature in terms of out-of-sample prediction accuracy and solution time, and show that (1) our model is signicantly more tractable than alternate

mixed-integer optimization approaches and (2) our model produces trees that improve on trees obtained by top-down induction in out-of-sample predictive performance.

CHAPTER 2

Exact Logit-Based Product Design

Consider the following canonical marketing problem. A firm has to design a product, which has a collection of attributes, and each attribute can be set to one of a finite set of levels. The product will be offered to a collection of customers, which differ in their preferences and specifically, in the utility that they obtain from different levels of different attributes. What product should the firm offer – that is, to what level should each attribute be set – so as to maximize the share of customers who choose to purchase the product? This problem is referred to as the share-of-choice product design (SOCPD) problem, and has received a significant amount of attention in the marketing science research literature.

The SOCPD problem is a challenging problem for several reasons. First, since a product corresponds to a combination of attribute levels, the number of candidate products scales exponentially with the number of attributes, and can be enormous for even a modest number of attributes. This, in turn, renders solution approaches based on brute force enumeration computationally cumbersome. Second, it is common to represent customers using discrete choice models that are built on the multinomial logit model. Under this assumption of customer behavior, the problem becomes more complex, because the purchase probability under a logit choice model is a nonlinear function of the product design's utility that is neither convex nor concave. Finally, product design problems in real life settings may also often involve constraints, arising from engineering or other considerations, which can further constrain the set of candidate products.

In this chapter, we consider the logit-based SOCPD problem. In this problem, the firm

must design a product that maximizes the expected number of customers who choose to purchase a product, where customers are assumed to follow logit models of choice, and the probability of a customer purchasing a product is given by a logistic response function (i.e., the function $f(u) = e^u/(1 + e^u)$). We propose an exact solution methodology for this problem that is based on modern integer, convex and conic optimization. To the best of our knowledge, this is the first exact solution methodology for the logit-based SOCPD problem.

We make the following specific contributions:

1. We formally define the logit-based SOCPD problem and characterize the complexity of it. Specifically, we show that we can reduce the maximum independent set problem into the logit-based SOCPD problem, which implies that the latter problem is NP-Hard to approximate. We further show that the problem is NP-Hard even when there are only two customer segments.

2. Surprisingly, we prove that this problem can be exactly reformulated as a mixed-integer convex programming (MICONVP) problem. Our reformulation here relies on a useful characterization of logit probabilities as being the optimal solutions to a representative agent problem, in which an agent chooses the probability of selecting two alternatives so as to maximize a regularized expected utility.

3. We propose two specialized solution methods for this problem. The first approach involves further reformulating our MICONVP using conic constraints, leading to a mixed-integer conic programming problem that can be solved using cutting-edge solvers for such problems (such as Mosek). The second approach is a constraint generation procedure that adds constraints corresponding to gradient-based linear approximations of the principal nonlinear convex constraint in our MICONVP problem, thereby allowing the problem to be solved as a mixed-integer linear program using well-established commercial solvers such as Gurobi and CPLEX.

4. We consider two extensions of our base problem. In the first extension, we show how our optimization model can be readily modified for the optimization of expected profit when the profit of a product is a linear function of its attributes. In the second extension, we consider the problem of optimizing the geometric mean of the purchase probability across the customer types. For this latter extension, we prove that the relative performance gap between the resulting solution and the optimal solution of our original problem can be bounded in terms of the probability distribution of the customer types and the ratio between the lowest and highest purchase probabilities attainable by any product design over all of the customer types. We additionally show how the geometric mean maximization problem can also be formulated as a MICONVP problem, via a logarithmic transformation.

5. We demonstrate the practical tractability of our approach using synthetic problem instances, as well as a set of problem instances derived from real conjoint data sets.

The rest of this chapter is organized as follows. Section 2.1 provides a review of the related literature. Section 2.2 provides a formal definition of the logit-based SOCPD problem, and our exact reformulation of the problem as a MICONVP problem. Section 2.3 presents our two solution approaches based on mixed-integer conic programming and gradient-based constraint generation. Section 2.5 presents the results of our numerical experiments. Lastly, in Section 2.6, we conclude. All proofs are relegated to Appendix A.1.

2.1 Literature Review

We divide our literature review according to four subsets: the single product design literature; the product line design literature; the representative agent model literature; and lastly, the broader optimization literature.

Single product design: Product design has received significant attention in the marketing science community; we refer readers to [76] and [43] for overviews of this topic. The majority of papers on the SOCPD problem assume that customers follow a deterministic, first-choice model, i.e., they purchase the product if the utility exceeds a "hurdle" utility, and do not purchase it otherwise. Many papers have proposed heuristic approaches for this problem; examples include [55, 56] and [10]. Other papers have also considered exact approaches based on branch-and-bound [28] and nested partitions [78].

The main difference between our work and the majority of the prior work on the product design problem is the use of a logit-based share-of-choice objective function. As stated earlier, when the share-of-choice is defined as the sum of logit probabilities, the SOCPD problem becomes a discrete nonlinear optimization problem, and becomes significantly more difficult than the SOCPD problem when customers follow first-choice/max-utility models. To the best of our knowledge, our approach is the first approach for obtaining provably optimal solutions to the SOCPD problem when customers follow a logit model.

Product line design/assortment optimization: Besides the product design problem, a more general problem is the product line design (PLD) problem, where one must select several products so as to either maximize the share-of-choice, the expected profit or some other criterion. A number of papers have considered the PLD problem under a first-choice model of customer behavior, where customers deterministically select the product with the highest utility; examples of such papers include [63], [57], [88], [13] and [21]. Besides the first-choice model, several papers have also considered the PLD problem under the (single-class) multinomial logit model [29, 77]. Other work has also considered objective functions corresponding to a worst-case expectation over a family of choice models [20].

Outside of the marketing literature, the PLD problem is closely related to the problem of assortment optimization which appears in the operations management literature. In this problem, one must select a set of products from a larger universe of products so as to max-

imize expected revenue. The difference between PLD and assortment optimization arises from where the choice model comes: in PLD, the choice model usually comes from conjoint survey data and the task is to select a set of new products, whereas in assortment optimization, typically the set of candidate products consists of products that have been offered in the past, and the choice model is estimated from historical transactions. There is an extensive literature on solving this problem under a variety of choice models, such as the single class MNL model [79], the nested logit model [34] and the Markov chain choice model [39]; we refer readers to [42] for an recent overview of the literature.

Our paper differs from the product line and assortment optimization literatures in that we focus on the selection of a single product, and the decision variables of our optimization problem are the attributes of the product. In contrast, virtually all mathematical programming-based approaches to PLD/assortment optimization require one to input a set of candidate products, and the main decision variable is a set of products from the overall set of candidate products. The attributes of the products are only relevant in specifying problem data (e.g., in an MNL assortment problem, one would determine the utilities of the candidate products from their attributes), but do not directly appear as decision variables.

Representative agent model: Our MICONVP formulation is based on a characterization of logit probabilities as solutions of a concave maximization problem where the decision variables correspond to the choice probabilities and the objective function is the entropy-regularized expected utility. This concave maximization problem is an example of a representative agent model, and has been studied in a number of papers in the economics and operations management literatures [6, 47, 40]. The goal of our paper is not to contribute directly to this literature, but rather to leverage one such result so as to obtain an exact and computationally tractable reformulation of the logit-based SOCPD problem. To the best of our knowledge, the representative agent-based characterization of logit probabilities has not been previously used in optimization models arising in marketing or operations; we believe

that this characterization could potentially be useful in other contexts outside of product design.

Optimization literature: Lastly, we comment on the relation of our paper to the general optimization literature. Our paper contributes to the growing literature on mixed-integer convex and mixed-integer conic programming. In the optimization community, there has been an increasing interest in developing general solution methods for this class of problems (see, for example, [61, 33]) as well as understanding what types of optimization problems can and cannot be modeled as mixed-integer convex programs [60]. At the same time, mixed-integer convex and mixed-integer conic programming have been used in a variety of applications, such as power flow optimization [59], robotics [58], portfolio optimization [15], joint inventory-location problems [9] and designing battery swap networks for electric vehicles [62]. One of our solution approaches (see Section 2.3.1) specifically relies on the exponential cone; our paper contributes to a growing set of applications of exponential cone programming, which include scheduling charging of electric vehicles [30], robust optimization with uncertainty sets motivated by estimation objectives [90] and manpower planning [51].

Outside of this literature, we note that a couple of prior papers have considered the problem of designing a product to maximize the share-of-choice under a mixture of logit models. The first is the paper of [83] that considers the sum of sigmoids optimization problem, which is an optimization problem where the objective function is a sum of sigmoid (S-shaped) functions; the logistic response function $f(u) = e^u/(1 + e^u)$ is a specific type of sigmoid function. The paper of [83] develops a general purpose branch-and-bound algorithm for solving this problem when the decision variables are continuous. Our paper differs from that of [83] in that our paper is focused specifically on an objective that corresponds to a sum of logistic response functions, and the main decision variables of our formulation are binary variables, indicating the presence or absence of certain attributes. In addition, our formulation is an exact reformulation of the problem into a mixed-integer convex problem,

which can then be solved directly using a commercial mixed-integer conic solver (such as Mosek) or can be solved with a cutting plane approach implemented with an ordinary mixed-integer linear programming solver (such as Gurobi). In contrast, the approach of [83] requires one to solve the problem using a custom branch-and-bound algorithm.

The second is the paper of [48], which develops a mixed-integer linear programming formulation for general nonconvex piecewise linear functions. As an example of the application of the framework, the paper applies the framework to the problem of deciding on continuous product attributes to maximize a logit-based share-of-choice objective, which involves approximating the logistic response function $f(u) = e^u/(1 + e^u)$ using a piecewise linear function. As with our discussion of [83], our formulation differs in that it is exact, and that the attributes are discrete rather than continuous.

Besides [83] and [48], our paper is also related to the recent paper of [54], which considers the problem of estimating lexicographic rules from choice data. The paper formulates this problem as a nonlinear optimization problem, where one models the utility of each attribute as a random variable and chooses deterministic components of these random utilities so as to maximize the expected number of nonreversals (comparisons that are consistent with the choice data). While this is a nonconvex optimization problem, the paper shows that one can obtain a tractable approximate formulation by maximizing the geometric mean of the probability of a nonreversal, and develops a bound on the performance of this solution with respect to the original expected value objective. Our consideration of the geometric mean-based product design problem in Section 2.4.2 draws inspiration from [54]. However, aside from this high-level similarity, the formulations we present are different from those in [54]. In addition, the performance guarantee that we establish (Theorem 4) differs from the guarantee presented in [54]; the guarantee in [54] is a post-hoc bound, which requires one to know the optimal solution of the geometric mean problem in order to compute the approximation factor. In contrast, our guarantee in Theorem 4 is an a priori guarantee that can be calculated before one solves the geometric mean product design problem, and requires

a different proof technique.

Lastly, we note that our paper also contributes to a growing literature on optimization models where the objective function to be optimized is obtained from a predictive model or a machine learning model. Some examples include work on optimizing objective functions obtained from tree ensemble models (such as random forests; see [41], [64]) and neural networks [5].

2.2 Model

We begin by formally defining our model in Section 2.2.1, and then presenting our transformation of the problem into a mixed-integer convex program in Section 2.2.2.

2.2.1 Problem definition

We assume that there are n binary attributes. We assume the product design is described by a binary vector $\mathbf{a} = (a_1, \ldots, a_n)$, where a_i denotes the presence of attribute i. While we formulate the problem in terms of binary attributes, we note that this is without loss of generality, as we can formulate an attribute with M levels into $M-1$ binary attributes. We let $\mathcal{A} \subseteq \{0,1\}^n$ denote the set of feasible attribute vectors.

We assume that there are K different segments or *customer types*. Each customer type is associated with a nonnegative weight λ_k, which is the fraction of customers who belong to that type/segment, or alternatively the probability that a customer belongs to that type/segment; note that we always have that $\sum_{k=1}^{K} \lambda_k = 1$. Each customer type is also associated with a partworth vector $\boldsymbol{\beta_k} = (\beta_{k,1}, \ldots, \beta_{k,n}) \in \mathbb{R}^n$, where $\beta_{k,i}$ is the partworth of attribute i. In addition, we let $\beta_{k,0} \in \mathbb{R}$ denote the constant part of the customer's utility. Given a candidate design $\mathbf{a} \in \mathcal{A}$, the customer's utility for the product is given by

$$u_k(\mathbf{a}) = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i.$$

We assume that each customer type is choosing between our product design corresponding to the vector $\mathbf{a}$, and an outside/no-purchase option. Without loss of generality, we fix the utility of the outside option to zero. This assumption is not restrictive, as choice probabilities under the logit model are unaffected when all of the utilities are adjusted by a constant. In particular, an equivalent representation (one that would lead to the same choice probabilities) is to specify the utility of the product as $\sum_{i=1}^{n} \beta_{k,i} a_i$ and the utility of the no-purchase option as $-\beta_{k,0}$. As a result, the constant term $\beta_{k,0}$ effectively captures the utility of the no-purchase option.

We assume that each customer type chooses to buy or not buy the product according to a multinomial logit model. Thus, given $\mathbf{a} \in \mathcal{A}$, the customer chooses to purchase the product with probability $\exp(u_k(\mathbf{a}))/(1 + \exp(u_k(\mathbf{a})))$ and chooses the outside option with probability $1/(1 + \exp(u_k(\mathbf{a})))$.

With these definitions, the logit-based share-of-choice product design problem can then be defined as

$$\underset{\mathbf{a} \in \mathcal{A}}{\text{maximize}} \sum_{k=1}^{K} \lambda_k \cdot \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))}. \tag{2.1}$$

The objective function of this problem can be thought of as the share or fraction of all customers who choose to purchase the product, or the (unconditional) probability that a random customer chooses to purchase the product.

Our first major theoretical result is that problem (2.1) is theoretically intractable.

Theorem 1 *The problem of approximating Problem* (2.1) *within a factor* $O(n^{1-\epsilon})$ *for any fixed* $\epsilon > 0$ *is NP-Hard.*

This result (see Section A.1.1 for the proof) follows by reducing the maximum independent set problem, a well-known NP-Complete problem, to problem (2.1). This theorem implies that Problem (2.1) is NP-Hard and even the problem of approximating Problem (2.1) is extremely challenging. Our next result further illustrates the difficulty of Problem (2.1)

by proving that Problem (2.1) is NP-Hard even when the number of customer segments is two.

Theorem 2 *Problem* (2.1) *is NP-Hard even if* $K = 2$.

We prove this result (see Appendix A.1.2 for the proof) by showing that we can reduce the partition problem, a well-known NP-Complete problem, to problem (2.1). Notwithstanding Theorem 1 and 2, it is unreasonable to expect problem (2.1) to be easy: it is an optimization problem over a discrete feasible region, with a nonlinear objective function that is neither convex nor concave. In the next section, we turn our attention to solving this problem using mathematical programming. Finally, we remark that Theorem 1 and 2 can be linked to the two complexity results in the assortment optimization literature. Specifically, it has been shown that the assortment optimization problem under the ranking-based model is inapproximable by reducing the maximum independent set problem [8] and the assortment optimization under the latent-class MNL is NP-hard even when there are only two customer types [74]. Our results here differ from the literature by focusing on the product design problem.

2.2.2 Mixed-Integer Convex Programming Formulation

As discussed in Section 2.2.1, problem (2.1) is a challenging optimization problem. Surprisingly, it turns out that this problem can be reformulated as a mixed-integer convex programming (MICP) problem; that is, a problem of the form

$$\underset{\mathbf{x}}{\text{minimize}} \quad \mathbf{c}^T\mathbf{x} \tag{2.2a}$$

$$\text{subject to} \quad \mathbf{x} \in \mathcal{X}, \tag{2.2b}$$

$$x_i \in \mathbb{Z},\ l_i \leq x_i \leq u_i, \quad \forall i \in I, \tag{2.2c}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the decision variable, $\mathcal{X} \subseteq \mathbb{R}^n$ is a closed convex set, $I \subseteq \{1, \ldots, n\}$ is the subset of the decision variables restricted to take integer values, and l_i and u_i are lower

and upper bounds on each integer variable. In what follows, we show how to obtain this formulation.

We begin by first presenting some related background on the representative agent model. In the representative agent model, an agent is faced with M alternatives. Each alternative $m \in \{1, \ldots, M\}$ is associated with a utility π_m. The agent must choose the probability x_m with which each alternative will be selected; we let $\mathbf{x} = (x_1, \ldots, x_M)$ be the probability distribution over the M alternatives. The agent seeks to maximize his adjusted expected utility, where the adjustment is achieved through a convex regularization function $V(\mathbf{x})$. The representative agent model is then the following optimization problem:

$$\underset{\mathbf{x}}{\text{maximize}} \quad \sum_{i=1}^{M} \pi_m x_m - V(\mathbf{x}) \tag{2.3a}$$

$$\text{subject to} \quad \sum_{m=1}^{M} x_m = 1, \tag{2.3b}$$

$$x_m \geq 0, \quad \forall\, m \in \{1, \ldots, M\}. \tag{2.3c}$$

Since the function $V(\cdot)$ is a convex function, the above problem is a concave maximization problem. By carefully choosing the function V, the optimal solution of this problem – the probability distribution $\mathbf{x}$ – can be made to coincide with choice probabilities under different choice models. In particular, it is known that the function $V(\mathbf{x}) = \sum_{i=1}^{M} x_i \log(x_i)$ gives choice probabilities corresponding to a multinomial logit model [6]. We refer the reader to the excellent paper of [40] for a complete characterization of which discrete choice models can be captured by the representative agent model.

For our problem, the specific instantiation of the representative agent model that we are interested in is one corresponding to the choice of the kth customer type between our product and the no-purchase option. We let $x_{k,1}$ denote the probability of choosing our product with attribute vector $\mathbf{a}$, and $x_{k,0}$ denote the probability of choosing the no-purchase option. Recall that the utility of our product is $u_k(\mathbf{a})$, and the utility of the no-purchase option is 0. The representative agent model for this customer type can thus be formulated

as

$$\underset{x_{k,0},x_{k,1}}{\text{maximize}} \quad u_k(\mathbf{a}) \cdot x_{k,1} + 0 \cdot x_{k,0} - x_{k,1}\log(x_{k,1}) - x_{k,0}\log(x_{k,0}) \tag{2.4a}$$

$$\text{subject to} \quad x_{k,1} + x_{k,0} = 1, \tag{2.4b}$$

$$x_{k,1}, x_{k,0} \geq 0. \tag{2.4c}$$

We now show two key properties of this problem. First, we show that the unique optimal solution $(x^*_{k,1}, x^*_{k,0})$ is exactly the logit choice probabilities for the two alternatives. Second, we show that the optimal objective value can be found in closed form.

Proposition 1 *The unique optimal solution* $(x^*_{k,1}, x^*_{k,0})$ *of problem* (2.4) *is given by*

$$x^*_{k,1} = \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))},$$
$$x^*_{k,0} = \frac{1}{1 + \exp(u_k(\mathbf{a}))}.$$

In addition, the optimal objective value is $\log(1 + \exp(u_k(\mathbf{a})))$.

The proof of this proposition is provided in Appendix A.1.3. Using this result, we can now proceed with the formulation of our SOCPD problem. We first make the following assumption on the structure of the set $\mathcal{A}$.

Assumption 1 *The set* $\mathcal{A}$ *can be written as* $\mathcal{A} = \{\mathbf{a} \in \{0,1\}^n \mid \mathbf{C}\mathbf{a} \leq \boldsymbol{d}\}$ *for some choice of* $\mathbf{C} \in \mathbb{R}^{m \times n}$ *and* $\boldsymbol{d} \in \mathbb{R}^m$*, where* $m \in \mathbb{Z}_+$.

Assumption 1 just requires that the set of candidate products $\mathcal{A}$ can be represented as the set of binary vectors satisfying a finite collection of linear inequality constraints. This assumption is necessary in order to ensure that our problem can be formulated as a mixed-integer convex program of finite size. We note that this assumption is not too restrictive, as many natural constraints can be expressed in this way. We provide some examples at the end of this section.

With a slight abuse of notation, let u_k be a decision variable that denotes the utility of the candidate product $\mathbf{a}$ for customer type k. As before, let $x_{k,1}$ and $x_{k,0}$ denote the probability of customer type k purchasing and not purchasing the product, respectively. Then, the optimization problem can be formulated as

$$\underset{\mathbf{a},\mathbf{u},\mathbf{x}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k \cdot x_{k,1} \tag{2.5a}$$

$$\text{subject to} \quad x_{k,1} + x_{k,0} = 1, \quad \forall k \in \{1, \ldots, K\}, \tag{2.5b}$$

$$u_k x_{k,1} - x_{k,1} \log(x_{k,1}) - x_{k,0} \log(x_{k,0}) \geq \log(1 + \exp(u_k)), \quad \forall k \in \{1, \ldots, K\}, \tag{2.5c}$$

$$u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall k \in \{1, \ldots, K\}, \tag{2.5d}$$

$$\mathbf{Ca} \leq \mathbf{d}, \tag{2.5e}$$

$$a_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, \tag{2.5f}$$

$$x_{k,1}, x_{k,0} \geq 0, \quad \forall k \in \{1, \ldots, K\}. \tag{2.5g}$$

We now prove the validity of this formulation.

Theorem 3 *Problem* (2.5) *is equivalent to problem* (2.1).

Theorem 3 (see Appendix A.1.4 for the proof) establishes that problem (2.5) is equivalent to the original SOCPD problem (2.1). The key feature of this formulation is that it no longer explicitly involves the logit choice probabilities, which are a nonconvex function of u_k. We note that this problem is *almost* a mixed-integer convex program. In the main nonlinear constraint (2.5c), the functions $-x_{k,1} \log(x_{k,1})$ and $-x_{k,0} \log(x_{k,0})$ appearing on the left hand side are instances of the *entropy function* $-x \log(x)$ [25] and are concave in $x_{k,1}$ and $x_{k,0}$. Similarly, the function $\log(1 + \exp(u_k))$ appearing on the right hand side, which is known as the *softplus function* [65], is a convex function of u_k. Thus, this constraint can almost be written in the form $F(u_k, \mathbf{x}_k) \leq 0$, where F is a convex function. The main obstacle that

prevents us from doing this is the bilinear term $u_k x_{k,1}$, which is the product of two decision variables and is not jointly concave in u_k and $x_{k,1}$.

Fortunately, we can use the fact that $u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i$ to re-write this bilinear term as

$$\begin{aligned} u_k x_{k,1} &= (\beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i) x_{k,1} \\ &= \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} \cdot a_i x_{k,1}. \end{aligned}$$

Using the fact that each $a_i \in \{0, 1\}$ and that $0 \leq x_{k,1} \leq 1$, we can now linearize the bilinear terms $a_i x_{k,1}$ using a standard modeling technique from integer programming. In particular, we introduce a continuous decision variable $y_{k,i}$ for each k and i which corresponds to the product $a_i x_{k,1}$, and a continuous decision variable w_k which corresponds to the product $u_k x_{k,1}$. This leads to the following equivalent formulation:

$$\underset{\mathbf{a},\mathbf{u},\mathbf{w},\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k \cdot x_{k,1} \tag{2.6a}$$

$$\text{subject to} \quad x_{k,1} + x_{k,0} = 1, \quad \forall k \in \{1, \ldots, K\}, \tag{2.6b}$$

$$w_k - x_{k,1} \log(x_{k,1}) - x_{k,0} \log(x_{k,0}) \geq \log(1 + \exp(u_k)), \quad \forall k \in \{1, \ldots, K\}, \tag{2.6c}$$

$$u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall k \in \{1, \ldots, K\}, \tag{2.6d}$$

$$w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \quad \forall k \in \{1, \ldots, K\}, \tag{2.6e}$$

$$y_{k,i} \leq x_{k,1}, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.6f}$$

$$y_{k,i} \leq a_i, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.6g}$$

$$y_{k,i} \geq a_i - 1 + x_{k,1}, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.6h}$$

$$y_{k,i} \geq 0, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.6i}$$

$$\mathbf{C}\mathbf{a} \leq \mathbf{d}, \tag{2.6j}$$

$$a_i \in \{0,1\}, \quad \forall i \in \{1,\ldots,n\}, \tag{2.6k}$$

$$x_{k,1}, x_{k,0} \geq 0, \quad \forall k \in \{1,\ldots,K\}. \tag{2.6l}$$

Note that in the formulation above, when $\mathbf{a} \in \mathcal{A}$, $y_{k,i}$ is forced to take the value of $a_i \cdot x_{k,1}$, and w_k takes the value of $u_k \cdot x_{k,1}$. Problem (2.6) is a mixed-integer convex program, and can be tackled through a number of solution approaches, which we discuss next (Section 2.3).

Before continuing, we briefly discuss the modeling capability of the constraint $\mathbf{Ca} \leq \mathbf{d}$ arising from Assumption 1. The constraint $\mathbf{Ca} \leq \mathbf{d}$ can be used to encode a variety of requirements on the attribute vectors $\mathbf{a}$ as linear constraints. For example, if the product has two attributes, where the first attribute has three levels and the second attribute has four levels, then one can model the product through the vector $\mathbf{a} = (a_1, a_2, a_3, a_4, a_5)$, where a_1 and a_2 are dummy variables to represent two out of the three levels of the first attribute and a_3, a_4, a_5 are dummy variables to represent three out of the four levels of the second attribute. One would then need to enforce the constraints

$$a_1 + a_2 \leq 1, \tag{2.7}$$

$$a_3 + a_4 + a_5 \leq 1 \tag{2.8}$$

to ensure that at most one out of the variables a_1, a_2 is set to 1 and at most one variable out of a_3, a_4, a_5 is set to 1. This can be achieved by specifying $\mathbf{C}$ and $\mathbf{d}$ as

$$\mathbf{C} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \ \mathbf{d} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Beside the ability to represent multi-level attributes, one can use the constraint $\mathbf{Ca} \leq \mathbf{d}$ to represent design requirements such as weight and cost; for example, one may be interested in imposing the constraint

$$b_0 + \sum_{i=1}^{n} b_i a_i \leq B,$$

where b_0 is the base weight of the product, b_i is the incremental weight added to the product from attribute i and B is a limit on the overall weight of the product. This constraint can

be modeled by specifying $\mathbf{C}$ and $\mathbf{d}$ as

$$\mathbf{C} = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}, \ \mathbf{d} = [B - b_0].$$

2.3 Solution Approaches

In this section, we present two different approaches for solving problem (2.6). In Section 2.3.1, we present a solution approach based on transforming the mixed-integer convex program (2.6) into a mixed-integer conic program, thereby allowing for the problem to be solved by mixed-integer conic solvers such as Mosek. In Section 2.3.2, we present a solution approach based on generating constraints corresponding to the linear underapproximations of the convex functions appearing in constraint (2.6c), effectively allowing the problem to be solved by mixed-integer linear solvers such as Gurobi and CPLEX.

2.3.1 Solution Approach #1: Representation as Mixed-Integer Conic Program

The first approach that we describe for solving problem (2.6) involves further reformulating the problem into a mixed-integer conic programming (MICP) problem. We begin by providing a brief overview of mixed-integer conic optimization problems, and then present our formulation.

A mixed-integer conic programming problem has the following general form:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & \mathbf{c}^T\mathbf{x} && \text{(2.9a)} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}, && \text{(2.9b)} \\ & x_i \in \mathbb{Z}, \quad \forall\ i \in \{1, \ldots, I\}, && \text{(2.9c)} \end{aligned}$$

where n is the dimension of the decision variable, $\mathbf{c}$ is an n-dimensional vector, $\mathbf{b}$ is an m-dimensional vector, $\mathbf{A}$ is an m-by-n matrix, $I \leq n$ is the number of integer variables in the problem and finally, $\mathcal{K}$ is a closed convex cone. A closed convex cone $\mathcal{K}$ is a closed subset

of $\mathbb{R}^n$ that contains all nonnegative combinations of its elements, i.e., a set $\mathcal{K}$ satisfying the following property:

$$\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{K} \;\Rightarrow\; \alpha_1 \mathbf{y}_1 + \alpha_2 \mathbf{y}_2 \in \mathcal{K} \text{ for any } \alpha_1, \alpha_2 \geq 0. \tag{2.10}$$

While the cone $\mathcal{K}$ can in theory be specified as any set that satisfies (2.10), in practice, it is common to model $\mathcal{K}$ as a Cartesian product of a collection of cones drawn from the set of standard cones. An example of a standard cone is the the cone $\mathcal{K}_{\geq} = \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y} \geq \mathbf{0}\}$, where $\mathbf{0}$ is an m-dimensional vector of zeros. This cone is known as the nonnegative cone, as it corresponds to the nonnegative orthant of $\mathbb{R}^n$. The constraint $\mathbf{A}\mathbf{x} - \mathbf{b} \in \mathcal{K}_{\geq}$ is equivalent to the constraint $\mathbf{A}\mathbf{x} \geq \mathbf{b}$, which is just a linear constraint. Other standard cones include the zero cone, the second order cone, the exponential cone and the positive semidefinite cone; we refer readers to [65] for an overview of other standard cones.

Having described mixed-integer conic programming in generality, we now elaborate on why this representation is valuable. Many mixed-integer convex programs involve complicated nonlinear functions. Until recently, the method of choice for tackling such problems has been to use mixed-integer nonlinear programming solvers, which treat these nonlinear functions in a "black-box" fashion and rely on evaluating these functions and their derivatives to solve the problem. Often, it turns out that constraints involving these nonlinear functions can be re-written through additional variables and additional conic constraints involving standard cones. [1] In so doing, one obtains a mixed-integer conic program, which is then amenable to solution methods for such problems. This is important because conic programs – problems of the same form as (2.9), without the integrality constraint (2.9c) – are considered to be among the easiest continuous nonlinear programs to solve: the theory of numerical algorithms for solving these problems is quite developed, there are numerous software packages for solving these problems at practical scale, and there continues to be active

[1] A notable recent example of this is the paper of [61], which found that all 194 mixed-integer convex programming problems in the MINLPLIB2 (`http://www.gamsworld.org/minlp/minlplib2/html/`) benchmark library could be represented as mixed-integer conic programs using standard cones.

development both in the theory and in software implementations. Solution algorithms for mixed-integer conic programs are built on top of algorithms for (continuous) conic programs and can exploit the conic structure. By re-writing our mixed-integer convex program (2.6) as a mixed-integer conic program, one is able to use state-of-the-art commercial solvers such as Mosek [66], as well as new open-source solvers such as Pajarito [33] to solve the problem.

Thus motivated, we proceed with our reformulation of (2.6) as a MICP problem. To do so, we will make use of a standard cone known as the *exponential cone*, which is defined as

$$\mathcal{K}_{\text{exp}} = \{(r, 0, t) \in \mathbb{R}^3 \mid r \geq 0, t \leq 0\} \cup \{(r, s, t) \in \mathbb{R}^3 \mid s > 0, r \geq s \exp(t/s)\}. \tag{2.11}$$

The exponential cone is useful precisely because it can be used to represent the entropy function $x \log(x)$ and the softplus function $\log(1+e^x)$ which appear in our formulation (2.6). For the former, the constraint

$$t \leq -x \log(x) \tag{2.12}$$

can be written as the conic constraint

$$(1, x, t) \in \mathcal{K}_{\text{exp}}.$$

For the latter, the constraint

$$t \geq \log(1 + \exp(x)) \tag{2.13}$$

can be written by introducing auxiliary variables v_1, v_2 and then using the following set of conic constraints:

$$\begin{aligned} &v_1 + v_2 \leq 1, \\ &(v_1, 1, x - t) \in \mathcal{K}_{\text{exp}}, \\ &(v_2, 1, -t) \in \mathcal{K}_{\text{exp}}. \end{aligned}$$

Armed with these two properties, we recall constraint (2.6c) for a fixed k:

$$w_k - x_{k,1} \log(x_{k,1}) - x_{k,0} \log(x_{k,0}) \geq \log(1 + \exp(u_k))$$

We now introduce the auxiliary variables $\theta_{k,1}, \theta_{k,0}$ and ϕ_k, and reformulate this as the following equivalent set of constraints:

$$\begin{aligned}
&w_k + \theta_{k,1} + \theta_{k,0} \geq \phi_k, \\
&\theta_{k,1} \leq -x_{k,1} \log(x_{k,1}), \\
&\theta_{k,0} \leq -x_{k,0} \log(x_{k,0}), \\
&\phi_k \geq \log(1 + \exp(u_k)).
\end{aligned}$$

We next introduce the auxiliary variables $v_{k,0}$ and $v_{k,1}$ to represent the softplus function, and re-write the above four constraints as the following family of conic constraints:

$$\begin{aligned}
&w_k + \theta_{k,1} + \theta_{k,0} \geq \phi_k, \\
&(1, x_{k,1}, \theta_{k,1}) \in \mathcal{K}_{\text{exp}}, \\
&(1, x_{k,0}, \theta_{k,0}) \in \mathcal{K}_{\text{exp}}, \\
&v_{k,0} + v_{k,1} \leq 1, \\
&(v_{k,1}, 1, u_k - \phi_k) \in \mathcal{K}_{\text{exp}}, \\
&(v_{k,0}, 1, -\phi_k) \in \mathcal{K}_{\text{exp}}.
\end{aligned}$$

This leads to the following mixed-integer conic programming formulation of our original problem:

$$\begin{aligned}
\underset{\mathbf{a},\mathbf{u},\mathbf{v},\mathbf{w},\mathbf{x},\mathbf{y},\boldsymbol{\theta},\boldsymbol{\phi}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda_k \cdot x_{k,1} && \text{(2.14a)} \\
\text{subject to} \quad & x_{k,1} + x_{k,0} = 1, \quad \forall k \in \{1, \ldots, K\}, && \text{(2.14b)} \\
& w_k + \theta_{k,1} + \theta_{k,0} \geq \phi_k, \quad \forall k \in \{1, \ldots, K\}, && \text{(2.14c)} \\
& (1, x_{k,1}, \theta_{k,1}) \in \mathcal{K}_{\text{exp}}, \quad \forall k \in \{1, \ldots, K\}, && \text{(2.14d)} \\
& (1, x_{k,0}, \theta_{k,0}) \in \mathcal{K}_{\text{exp}}, \quad \forall k \in \{1, \ldots, K\}, && \text{(2.14e)} \\
& v_{k,0} + v_{k,1} \leq 1, \quad \forall k \in \{1, \ldots, K\}, && \text{(2.14f)} \\
& (v_{k,1}, 1, u_k - \phi_k) \in \mathcal{K}_{\text{exp}}, \quad \forall k \in \{1, \ldots, K\}, && \text{(2.14g)}
\end{aligned}$$

$$(v_{k,0}, 1, -\phi_k) \in \mathcal{K}_{\text{exp}}, \quad \forall k \in \{1, \ldots, K\}, \tag{2.14h}$$

$$u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall k \in \{1, \ldots, K\}, \tag{2.14i}$$

$$w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \quad \forall k \in \{1, \ldots, K\}, \tag{2.14j}$$

$$y_{k,i} \leq x_{k,1}, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.14k}$$

$$y_{k,i} \leq a_i, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.14l}$$

$$y_{k,i} \geq a_i - 1 + x_{k,1}, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.14m}$$

$$y_{k,i} \geq 0, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.14n}$$

$$\mathbf{Ca} \leq \mathbf{d}, \tag{2.14o}$$

$$a_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, \tag{2.14p}$$

$$x_{k,1}, x_{k,0} \geq 0, \quad \forall k \in \{1, \ldots, K\}. \tag{2.14q}$$

Formulation (2.14) is attractive because, as mentioned earlier, it is in a form where it can be solved by mixed-integer conic programming solvers. In our numerical experiments in Section 2.5, we use one such solver, Mosek, which as of 2019 supports the exponential cone and mixed-integer problems involving the exponential cone.

2.3.2 Solution Approach #2: Gradient-Based Constraint Generation

Our second approach for solving this problem is gradient-based constraint generation. The idea of this approach is to sequentially approximate the nonlinear convex functions in the formulation (2.6) using their linear approximations.

To motivate this approach, we recall a standard property of convex functions. For any differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$ and any point $\bar{\mathbf{x}} \in \mathbb{R}^n$, the function f is lower-bounded by the first-order approximation of f at $\bar{\mathbf{x}}$:

$$f(\mathbf{x}) \geq f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}). \tag{2.15}$$

A consequence of this property is that the function f can be written as the pointwise maximum of a family of linear functions, where each linear function in the family corresponds to the above first-order approximation:

$$f(\mathbf{x}) = \max_{\bar{\mathbf{x}} \in \mathbb{R}^n} \left\{ f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}) \right\} \tag{2.16}$$

Thus, a convex constraint of the form

$$f(\mathbf{x}) \leq 0 \tag{2.17}$$

can be written as the constraint

$$\max_{\bar{\mathbf{x}} \in \mathbb{R}^n} \left\{ f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}) \right\} \leq 0, \tag{2.18}$$

or equivalently, as

$$f(\bar{\mathbf{x}}) + \nabla f(\bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}) \leq 0, \quad \forall\ \bar{\mathbf{x}} \in \mathbb{R}^n. \tag{2.19}$$

In formulation (2.6), the principal constraint which involves a convex, differentiable function is constraint (2.6c). We can re-write this constraint as

$$F(w_k, x_{k,1}, x_{k,0}, u_k) \leq 0 \tag{2.20}$$

where

$$F(w_k, x_{k,1}, x_{k,0}, u_k) = -w_k + x_{k,1} \log x_{k,1} + x_{k,0} \log x_{k,0} + \log(1 + \exp(u_k)), \tag{2.21}$$

and the domain of F is

$$\mathcal{V}_k = \{(w_k, x_{k,1}, x_{k,0}, u_k) \in \mathbb{R}^4 \mid x_{k,1} > 0, x_{k,0} > 0\}. \tag{2.22}$$

The gradient of F is

$$\nabla F = \begin{bmatrix} -1 \\ \log x_{k,1} + 1 \\ \log x_{k,0} + 1 \\ \frac{\exp(u_k)}{1+\exp(u_k)} \end{bmatrix}. \tag{2.23}$$

The constraint $F(w_k, x_{k,1}, x_{k,0}, u_k) \leq 0$ can therefore be equivalently re-written as the following family of linear constraints:

$$F(\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k) + (-1)(w_k - \bar{w}_k) + (\log \bar{x}_{k,1} + 1)(x_{k,1} - \bar{x}_{k,1}) \\ +(\log \bar{x}_{k,0} + 1)(x_{k,0} - \bar{x}_{k,0}) + \frac{\exp(\bar{u}_k)}{1 + \exp(\bar{u}_k)} \cdot (u_k - \bar{u}_k) \leq 0, \\ \forall\ (\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k) \in \mathcal{V}_k. \tag{2.24}$$

We therefore obtain the following formulation which is equivalent to formulation (2.6):

$$\underset{\mathbf{a},\mathbf{u},\mathbf{w},\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k \cdot x_{k,1} \tag{2.25a}$$

$$\text{subject to} \quad x_{k,1} + x_{k,0} = 1, \quad \forall k \in \{1, \ldots, K\}, \tag{2.25b}$$

$$F(\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k) + (-1)(w_k - \bar{w}_k) + (\log \bar{x}_{k,1} + 1)(x_{k,1} - \bar{x}_{k,1}) \\ + (\log \bar{x}_{k,0} + 1)(x_{k,0} - \bar{x}_{k,0}) + \frac{\exp(\bar{u}_k)}{1 + \exp(\bar{u}_k)} \cdot (u_k - \bar{u}_k) \leq 0, \\ \forall\ (\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k) \in \mathcal{V}_k,\ k \in \{1, \ldots, K\}, \tag{2.25c}$$

$$u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall k \in \{1, \ldots, K\}, \tag{2.25d}$$

$$w_k = \beta_{k,0} x_{k,1} + \sum_{i=1}^{n} \beta_{k,i} y_{k,i}, \quad \forall k \in \{1, \ldots, K\}, \tag{2.25e}$$

$$y_{k,i} \leq x_{k,1}, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.25f}$$

$$y_{k,i} \leq a_i, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.25g}$$

$$y_{k,i} \geq a_i - 1 + x_{k,1}, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.25h}$$

$$y_{k,i} \geq 0, \quad \forall k \in \{1, \ldots, K\},\ i \in \{1, \ldots, n\}, \tag{2.25i}$$

$$\mathbf{Ca} \leq \mathbf{d}, \tag{2.25j}$$

$$a_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, \tag{2.25k}$$

$$x_{k,1}, x_{k,0} \geq 0, \quad \forall k \in \{1, \ldots, K\}. \tag{2.25l}$$

The difference between the new formulation (2.25) and the original formulation (2.6) is that the convex constraint (2.6c) has been replaced by an infinite family of linear constraints.

Although the new formulation remains difficult to solve, the key advantage of this formulation is that it is in a form that is suited to constraint generation. The idea of constraint generation is to solve problem (2.25) with constraint (2.25c) enforced only at a finite set of points $\bar{\mathcal{V}}_k \subset \mathcal{V}_k$, to obtain a candidate solution $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$. We then test whether this candidate solution satisfies constraint (2.25c) for each k; by the equivalence (2.16), it is sufficient to simply evaluate $F(w_k, x_{k,1}, x_{k,0}, u_k)$ and check whether it is less than or equal to zero. If it is, we conclude that $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies constraint (2.25c) for customer type k. If not, then we add the $(\bar{w}_k, \bar{x}_{k,1}, \bar{x}_{k,0}, \bar{u}_k)$ to the set $\bar{\mathcal{V}}_k$ and solve the problem again. In this process, if a solution $(\bar{\mathbf{a}}, \bar{\mathbf{u}}, \bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies constraint (2.25c) for all customer types k, we conclude that it is the optimal one.

We comment on two important aspects of this approach. First, this approach is attractive because it represents the convex constraint $F(w_k, x_{k,1}, x_{k,0}, u_k) \leq 0$ for each k through a family of linear constraints, quantified over the finite set $\bar{\mathcal{V}}_k$. Thus, the problem is a mixed-integer linear program. Such a problem can be solved using commercial solvers for such problems like Gurobi or CPLEX. Second, in implementing a constraint generation procedure for solving problem (2.25), one can use lazy constraint generation. In traditional constraint generation, one re-solves problem (2.25) from scratch using branch-and-bound whenever one discovers a violated instance of constraint (2.25c). In lazy constraint generation, one solves problem (2.25) using a single branch-and-bound tree, and checks whether constraint (2.25c) is satisfied for each integer solution that is encountered; if a violated instance of constraint (2.25c) is found, it can then be added to the current node in the branch-and-bound tree. This type of approach is potentially more efficient as one does not re-solve the problem from scratch each time that a violated constraint is added.

2.4 Extensions

In this section, we discuss how to extend our base model, which considers the share-of-choice objective, to two different objective functions: the expected per-customer profit (Section 2.4.1) and the geometric mean of the customer purchase probability (Section 2.4.2).

2.4.1 Extension to Expected Profit Maximization

While our final mixed-integer convex program (2.6) corresponds to the share-of-choice objective, it turns out that it is straightforward to generalize the model so as to optimize a profit-based objective. In particular, suppose that the marginal profit of a design $\mathbf{a}$ is given by a function $R(\mathbf{a})$ defined as

$$R(\mathbf{a}) = r_0 + \sum_{i=1}^{n} r_i a_i.$$

In other words, the profit $R(\mathbf{a})$ is a linear function of the binary attributes. One can model various types of profit structures with this assumption. For example, if all of the attributes correspond to non-price features that affect the cost of the product, then one can set r_0 to be the price of the product (a positive quantity), and each r_i to be the marginal incremental cost of attribute i (a negative quantity).

With this assumption, the logit-based expected profit product design problem can be written as

$$\underset{\mathbf{a}\in\mathcal{A}}{\text{maximize}} \quad R(\mathbf{a}) \cdot \left[\sum_{k=1}^{K} \lambda_k \cdot \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \right]. \tag{2.26}$$

Using similar steps as for the SOCPD problem, one can derive the following bilinear formulation with a bilinear objective function:

$$\underset{\mathbf{a},\mathbf{u},\mathbf{x}}{\text{maximize}} \quad R(\mathbf{a}) \cdot \left[\sum_{k=1}^{K} \lambda_k \cdot x_{k,1} \right] \tag{2.27a}$$

$$\text{subject to} \quad \text{constraints (2.5b) - (2.5g).} \tag{2.27b}$$

For the objective function, observe that we can re-write it as

$$R(\mathbf{a}) \cdot \left[\sum_{k=1}^{K} \lambda_k \cdot x_{k,1}\right] = \left(r_0 + \sum_{i=1}^{n} r_i a_i\right) \cdot \left[\sum_{k=1}^{K} \lambda_k \cdot x_{k,1}\right]$$
$$= \sum_{k=1}^{K} \lambda_k \cdot \left[r_0 x_{k,1} + \sum_{i=1}^{n} r_i \cdot a_i x_{k,1}\right].$$

Notice that this last expression includes terms of the form $a_i x_{k,1}$, which we can already represent through the variables $y_{k,i}$ introduced for problem (2.6). Problem (2.26) can therefore be exactly reformulated as the following mixed-integer convex program:

$$\underset{\mathbf{a},\mathbf{u},\mathbf{w},\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k \cdot \left[r_0 x_{k,1} + \sum_{i=1}^{n} r_i \cdot y_{k,i}\right] \tag{2.28a}$$

$$\text{subject to} \quad \text{constraints (2.6b) - (2.6l).} \tag{2.28b}$$

Thus, the expected profit product design problem can be handled through the same formulation as problem (2.6), only with a modified objective function.

2.4.2 Extension to Geometric Mean Maximization

To motivate this approach, recall that the logit-based SOCPD problem (2.1) involves maximizing the fraction of customers who purchase a product. This objective function is formulated as the weighted sum of the logit probabilities of each customer purchasing the product. An alternate way of understanding this objective is that it is the weighted arithmetic mean of the logit probabilities of the customer types.

Instead of formulating the objective of our product design problem as an arithmetic mean, we will instead consider formulating the problem using the geometric mean. This leads to the following optimization problem:

$$\underset{\mathbf{a}\in\mathcal{A}}{\text{maximize}} \prod_{k=1}^{K} \left[\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))}\right]^{\lambda_k}. \tag{2.29}$$

In other words, rather than trying to optimize the weighted arithmetic mean of the purchase probabilities, this problem seeks to optimize the weighted geometric mean of the purchase

probabilities, where the weights indicate the relative proportion of each customer type in the population.

This formulation is interesting to consider because it provides a lower bound on the optimal value of problem (2.1).

Proposition 2 *Let Z^*_{AM} and Z^*_{GM} be the optimal objective values of problems* (2.1) *and* (2.29)*, respectively. Then $Z^*_{AM} \geq Z^*_{GM}$.*

Proposition 2 (see Appendix A.1.5 for the proof) implies that, by solving problem (2.29), we obtain a lower bound on problem (2.1); by evaluating the objective value of the optimal solution of (2.29) within problem (2.1), we obtain an even stronger lower bound. The solution of the geometric mean problem (2.29) can be used as an approximate solution of the arithmetic mean problem (2.1).

We can further analyze the approximation quality of the solution of problem (2.29) with regard to the original problem. With a slight abuse of notation, let us use $\mathbf{x} = (x_1, \ldots, x_K)$ to denote the vector of purchase probabilities for the K different customer types, and let us use $\mathbf{x}(\mathbf{a})$ to denote the vector of purchase probabilities for a given product $\mathbf{a} \in \mathcal{A}$:

$$\mathbf{x}(\mathbf{a}) = (x_1(\mathbf{a}), \ldots, x_K(\mathbf{a})) = \left(\frac{\exp(u_1(\mathbf{a}))}{1 + \exp(u_1(\mathbf{a}))}, \ldots, \frac{\exp(u_K(\mathbf{a}))}{1 + \exp(u_K(\mathbf{a}))} \right).$$

Let us also use $\mathcal{X}$ be the set of achievable customer choice probabilities, given by

$$\mathcal{X} = \left\{ \mathbf{x} \in [0,1]^K \mid x_k = \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))} \text{ for some } \mathbf{a} \in \mathcal{A} \right\}. \tag{2.30}$$

Given a vector of choice probabilities $\mathbf{x}$, we use the function $f : \mathcal{X} \to \mathbb{R}$ to denote the weighted arithmetic mean of $\mathbf{x}$, with the weights $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$:

$$f(\mathbf{x}) = \sum_{k=1}^{K} \lambda_k x_k. \tag{2.31}$$

Similarly, we use $g : \mathcal{X} \to \mathbb{R}$ to denote the weighted geometric mean of $\mathbf{x}$:

$$g(\mathbf{x}) = \prod_{k=1}^{K} x_k^{\lambda_k}. \tag{2.32}$$

Thus, in terms of these two functions, the original logit-based SOCPD problem can be written as $\max_{\mathbf{a}\in\mathcal{A}} f(\mathbf{x}(\mathbf{a}))$, while the geometric mean problem (2.29) can be written as $\max_{\mathbf{a}\in\mathcal{A}} g(\mathbf{x}(\mathbf{a}))$. We then have the following guarantee on the performance of any solution of the geometric mean problem (2.29) with respect to the objective of the original logit-based SOCPD problem (2.1).

Theorem 4 *Let L and U be nonnegative numbers satisfying $L \leq x_k(\mathbf{a}) \leq U$ for all $k \in \{1,\ldots,K\}$ and $\mathbf{a} \in \mathcal{A}$. Let $\mathbf{a}^* \in \arg\max_{\mathbf{a}\in\mathcal{A}} f(\mathbf{x}(\mathbf{a}))$ be a solution of the arithmetic mean problem, and $\hat{\mathbf{a}} \in \arg\max_{\mathbf{a}\in\mathcal{A}} g(\mathbf{x}(\mathbf{a}))$ be a solution of the geometric mean problem. Then the geometric mean solution $\hat{\mathbf{a}}$ satisfies*

$$f(\mathbf{x}(\hat{\mathbf{a}})) \geq \frac{1}{\sum_{k=1}^{K} \lambda_k \left(\frac{U}{L}\right)^{1-\lambda_k}} \cdot f(\mathbf{x}(\mathbf{a}^*)).$$

The proof of Theorem 4 (see Section A.1.6 of the ecompanion) follows by finding constants $\underline{\alpha}$ and $\overline{\alpha}$ such that $\underline{\alpha} f(\mathbf{x}) \leq g(\mathbf{x}) \leq \overline{\alpha} g(\mathbf{x})$ for any vector of probabilities $\mathbf{x}$, and then showing that a solution $\hat{\mathbf{a}}$ that maximizes $g(\mathbf{x}(\cdot))$ must be within a factor $\underline{\alpha}/\overline{\alpha}$ of the optimal objective of the arithmetic mean problem. Theorem 4 is valuable because it provides some intuition for when a solution $\hat{\mathbf{a}}$ obtained by solving the geometric mean problem (2.29) will be close in performance to the optimal solution of the original (arithmetic mean) problem (2.1). In particular, the factor Γ defined as

$$\Gamma = \underline{\alpha} \,/\, \overline{\alpha} = \frac{1}{\sum_{k=1}^{K} \lambda_k \left(\frac{U}{L}\right)^{1-\lambda_k}}$$

is decreasing in the ratio U/L. Recall that U is an upper bound on the highest purchase probability that can be achieved for any customer type, while L is similarly a lower bound on the lowest purchase probability that can be achieved for any customer type. When the ratio U/L is large, it implies that there is a large range of choice probabilities spanned by the set of product designs $\mathcal{A}$. On the other hand, when U/L is small, then the range of choice

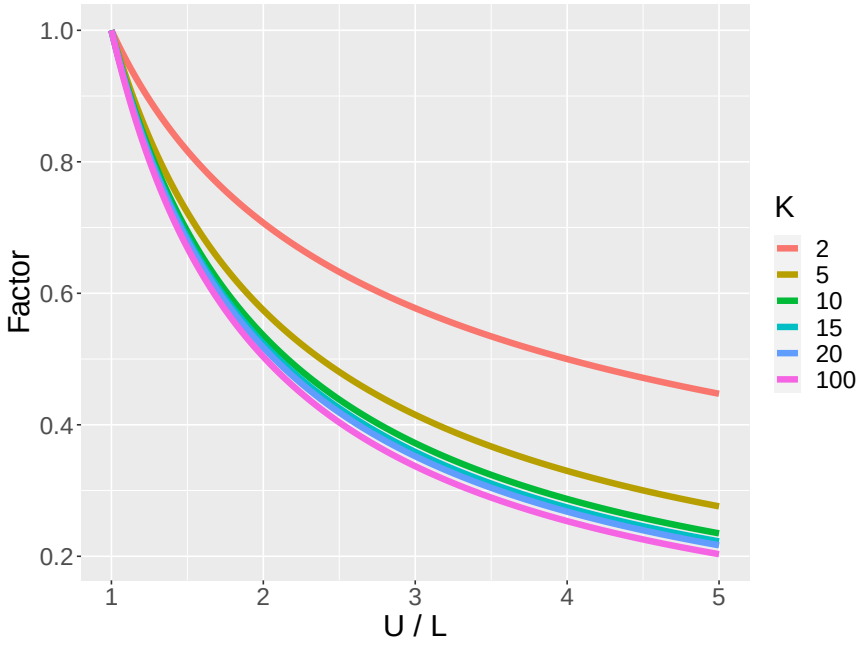


Figure 2.1: Plot of the approximation factor Γ as a function of the ratio U/L, for different values of K, with uniformly-distributed $\boldsymbol{\lambda}$.

probabilities is smaller. Thus, the smaller the range of choice probabilities spanned by the set $\mathcal{A}$ is small, the closer we should expect the geometric mean solution to be in performance to the optimal solution of the arithmetic mean problem. Figure 2.1 visualizes the dependence of the factor Γ on U/L when $\boldsymbol{\lambda}$ is assumed to be the discrete uniform distribution and K is varied.

In addition to the ratio U/L, the factor Γ is also affected by $\boldsymbol{\lambda}$. It can be verified that the factor Γ is minimized when the customer type distribution is uniform, i.e., $\boldsymbol{\lambda} = (1/K, \ldots, 1/K)$. In addition, it can also be verified that when $\boldsymbol{\lambda}$ is such that $\lambda_k = 1$ for a single customer type (and $\lambda_{k'} = 0$ for all others), the factor Γ becomes 1. Thus, the more "unbalanced" the customer type distribution $\boldsymbol{\lambda}$ is, the closer the geometric mean solution should be in performance to the optimal solution of the arithmetic mean problem.

We will see in our numerical experiments that the solution of problem (2.29) is often significantly better than the lower bound of Theorem 4.

Lastly, with regard to the bounds U and L, we note that these can be found easily. In particular, for each customer type k, one can compute $u_{k,\max} = \max_{\mathbf{a}\in\mathcal{A}} u_k(\mathbf{a})$ and $u_{k,\min} = \min_{\mathbf{a}\in\mathcal{A}} u_k(\mathbf{a})$, which are the highest and lowest utilities that one can attain for customer type k; for many common choices of $\mathcal{A}$ this should be an easy problem. (For example, if $\mathcal{A}$ is simply $\{0,1\}^n$, we can find $u_{k,\max}$ by setting to 1 those attributes for which $\beta_{k,i} > 0$ and setting to 0 all other attributes; $u_{k,\min}$ can be found in a similar manner). One can then compute L and U as

$$U = \max_{k=1,\ldots,K} \frac{\exp(u_{k,\max})}{1+\exp(u_{k,\max})},$$
$$L = \min_{k=1,\ldots,K} \frac{\exp(u_{k,\min})}{1+\exp(u_{k,\min})}.$$

We now turn our attention to how one can solve problem (2.29). While problem (2.29) is still a challenging nonconvex problem, it is possible to transform it into a mixed-integer convex problem. To do so, we consider taking the logarithm of the objective function of (2.29):

$$\begin{aligned}\log \prod_{k=1}^{K} \left[\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))}\right]^{\lambda_k} &= \sum_{k=1}^{K} \lambda_k \log\left(\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))}\right) \\ &= \sum_{k=1}^{K} \lambda_k \cdot \left(u_k(\mathbf{a}) - \log(1+\exp(u_k(\mathbf{a})))\right).\end{aligned}$$

This transformation is useful because the logarithm function is monotonic, so any solution that maximizes the logarithm of the objective function maximizes the objective function itself. This leads to the following mixed-integer convex program:

$$\underset{\mathbf{a},\mathbf{u}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k \cdot (u_k - \log(1+\exp(u_k))) \tag{2.33a}$$

$$\text{subject to} \quad u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall\ k \in \{1,\ldots,K\}, \tag{2.33b}$$

$$\mathbf{Ca} \leq \mathbf{d}, \tag{2.33c}$$

$$a_i \in \{0,1\}, \quad \forall i \in \{1,\ldots,n\}. \tag{2.33d}$$

This problem can be solved using a similar gradient-based constraint generation approach as the one described in Section 2.3.2. Alternatively, as in Section 2.3.1, one can also translate this problem into a mixed-integer conic program using the exponential cone, leading to the following formulation:

$$\underset{\mathbf{a},\mathbf{u},\mathbf{v},\mathbf{t},\boldsymbol{\phi}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda_k \cdot t_k \tag{2.34a}$$

$$\text{subject to} \quad u_k = \beta_{k,0} + \sum_{i=1}^{n} \beta_{k,i} a_i, \quad \forall\, k \in \{1,\ldots,K\}, \tag{2.34b}$$

$$t_k + \phi_k \leq u_k, \quad \forall\, k \in \{1,\ldots,K\}, \tag{2.34c}$$

$$v_{k,1} + v_{k,0} \leq 1, \quad \forall\, k \in \{1,\ldots,K\}, \tag{2.34d}$$

$$(v_{k,1}, 1, u_k - \phi_k) \in \mathcal{K}_{\text{exp}}, \quad \forall\, k \in \{1,\ldots,K\}, \tag{2.34e}$$

$$(v_{k,0}, 1, -\phi_k) \in \mathcal{K}_{\text{exp}}, \quad \forall\, k \in \{1,\ldots,K\}, \tag{2.34f}$$

$$\mathbf{C}\mathbf{a} \leq \mathbf{d}, \tag{2.34g}$$

$$a_i \in \{0,1\}, \quad \forall i \in \{1,\ldots,n\}. \tag{2.34h}$$

While problem (2.33) and problem (2.34) bear some resemblance to their counterparts (2.6) and (2.14), they differ in that the choice probabilities $x_{k,1}$ and $x_{k,0}$ do not explicitly appear. More importantly, they do not involve any linearization of bilinear terms of the form $x_{k,1} \cdot a_i$. These differences are important because they make problems (2.33) and (2.34) much easier to solve than (2.6) and (2.14).

2.5 Numerical Experiments

In this section, we present the results of our numerical experiments. Section 2.5.1 presents the results of our experiments with synthetically generated problem instances, while Section 2.5.2 presents the results of our experiments with instances derived from real conjoint datasets. All of our numerical experiments are implemented in the Julia technical computing language, version 1.5 [23] using the JuMP package (Julia for Mathematical Programming; see [36]),

and executed on a 2017 Apple MacBook Pro with a 3.1GHz Intel i7 quad core CPU and 16GB of memory.

2.5.1 Experiments with synthetic instances

In our first set of numerical experiments, we test our approaches on synthetically generated problem instances. We generate these instances as follows. For a fixed number of binary attributes n and number of customer types K, we randomly generate the partworth $\beta_{k,i}$ by drawing an independent uniformly distributed random number in the interval $[-10, +10]$ for each customer type k and attribute i. For the intercept $\beta_{k,0}$ of the utility function, we assume that there are three competitive offerings $\mathbf{a}', \mathbf{a}'', \mathbf{a}'''$ which are drawn independently and uniformly at random from $\{0,1\}^n$, and thus the value of $\beta_{k,0}$ is computed as

$$\beta_{k,0} = -\log\left(\exp(u_k(\mathbf{a}')) + \exp(u_k(\mathbf{a}'')) + \exp(u_k(\mathbf{a}'''))\right). \tag{2.35}$$

We assume that the probability λ_k of each customer type k is set to $1/K$. We vary $n \in \{10, 20, 30\}$ and $K \in \{10, 20, 30, 40\}$. For each combination of n and K, we generate 5 replications. We do not impose any additional constraints on the set of feasible product vectors, i.e., we omit the constraint $\mathbf{Ca} \leq \mathbf{d}$ from the formulations. We compare the mixed-integer conic program (2.14) solved via Mosek (indicated by MICP), the gradient-based constraint generation method (formulation (2.25)) implemented in Gurobi (indicated by GCG) and the geometric mean problem (2.34) solved via Mosek (indicated by GM). For MICP and GCG, we impose a 1 hour computation time limit.

Table 2.1 shows the results. The first three columns report the objective value of the three methods, with respect to the logit-based share-of-choice objective. The next two columns report the optimality gap, which is the relative difference (UB − LB)/UB, where UB is the best upper bound found by Mosek/Gurobi, while LB is the objective value of the best product vector found by Mosek/Gurobi) for the MICP and GCG methods. The subsequent column reports the achieved approximation gap of the GM solution, which is the ratio of the

		Objective Value			Gap (%)			Computation Time (s)		
n	K	GCG	MICP	GM	GCG	MICP	GM	GCG	MICP	GM
10	10	0.591	0.591	0.422	0.0	0.0	31.9	1.1	1.0	3.0
10	20	0.478	0.478	0.282	0.0	0.0	39.8	0.6	2.4	0.1
10	30	0.456	0.456	0.314	0.0	0.0	32.0	0.9	4.3	0.2
10	40	0.391	0.391	0.233	0.0	0.0	40.3	1.4	6.3	0.2
20	10	0.752	0.752	0.501	0.0	0.0	34.0	5.3	12.8	0.1
20	20	0.682	0.682	0.509	0.0	0.0	27.1	32.4	70.1	0.2
20	30	0.529	0.529	0.341	0.0	0.0	36.0	149.3	360.1	0.3
20	40	0.477	0.477	0.295	0.0	0.0	38.0	504.8	636.5	0.3
30	10	0.916	0.916	0.827	0.0	0.0	10.0	30.3	104.0	0.1
30	20	0.761	0.761	0.586	0.0	6.6	24.5	1524.4	2992.3	0.3
30	30	0.668	0.681	0.445	15.8	17.2	36.2	3600.0	3600.7	0.3
30	40	0.538	0.536	0.325	30.3	31.9	40.5	3600.0	3600.6	0.4

Table 2.1: Results for numerical experiment with synthetic data.

difference between the objective of the GM solution and the best objective of any of the three solution methods, and the best objective over the three methods. The last three columns report the computation time required of each of the three methods. Each row corresponds to a combination of n and K, and all performance metrics are averaged over the 5 replications.

From Table 2.1, we can see that for $n \leq 20$, most of the instances can be solved to complete optimality by MICP or GCG within the 1 hour time limit. For $n = 30$, the instances become more challenging, and their difficulty increases as K increases; for $K = 10$ and $K = 20$, GCG and MICP are in general able to solve the problems to optimality within the one hour time limit, while for $K = 30$ and $K = 40$, both methods terminate with a

suboptimal solution, with an average optimality gap of roughly 15-17% (for $K = 30$) and an average optimality gap of roughly 30-32% (for $K = 40$). For the GM method, we can see that it obtains solutions of modest quality with an approximation gap ranging between 10 and 40%. For these instances, the factor of Γ (cf. Theorem 4) is in general extremely small (less than 10^{-11}, which would correspond to an approximation gap of close to 100%) because for each customer type it is possible to achieve utilities that are significantly smaller and greater than zero; our results thus show that the GM method is able to achieve significantly better solutions. In addition, the computation time required for GM is a fraction of the time required by the other methods.

2.5.2 Experiments with instances based on real conjoint datasets

In our second set of numerical experiments, we test our approaches using instances built with logit models estimated from real conjoint datasets. We use four different data sets: `timbuk2`, a dataset on preferences for laptop bags produced by Timbuk2 from [81] (see also [13], [20, 21], which also use this data set for profit-based product line design); `bank`, a dataset on preferences for credit cards from [4] (accessed through the `bayesm` package for R; see [73]); `candidate`, a dataset on preferences for a hypothetical presidential candidate from [45]; `immigrant`, a dataset on preferences for a hypothetical immigrant from [45]. The high-level characteristics of each dataset are summarized in Table 2.2 below.

We note that for some of these datasets, the product design problem is of a more hypothetical nature. For example, for `candidate`, the problem is to "design" a political candidate maximizing the share of voters who would vote for that candidate. Similarly, for `immigrant`, the problem is to "design" an ideal immigrant that would maximize the fraction of people who would support granting admission to such an immigrant. Clearly, it is not possible to "create" a political candidate or immigrant with certain characteristics. Despite this, we believe that identifying what an optimal "product" would be for these data sets, and what share-of-choice such a product would achieve, would still be insightful. Notwithstanding

Dataset	Respondents	Attributes	Attribute Levels	n
`bank`	946	7	$4 \times 4 \times 3 \times 3 \times 3 \times 2 \times 2$	14
`candidate`	311	8	$6 \times 2 \times 6 \times 6 \times 6 \times 6 \times 6 \times 2$	32
`immigrant`	1396	9	$7 \times 2 \times 10 \times 3 \times 11 \times 4 \times 4 \times 5 \times 4$	41
`timbuk2`	330	10	$7 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$	15

Table 2.2: Summary of real conjoint datasets used in Section 2.5.2.

these concerns, these datasets are still valuable from the perspective of verifying that our optimization methodology can solve problem instances derived from real data.

For each data set, we develop two different types of logit models, which we summarize below.

1. *Latent-class logit*: For each dataset, we estimate a latent-class (LC) multinomial logit with a finite number of classes K. We estimate each model using a custom implementation of the expectation-maximization (EM) algorithm [82]. For each dataset, we run the EM algorithm from five randomly chosen starting points, and retain the model with the lowest log likelihood. To ensure numerical stability, we impose the constraint $-10 \leq \beta_{k,i} \leq 10$ for each i in the M step of the algorithm. We vary the number of classes K in the set $\{5, 10, 15, 20, 30, 40, 50\}$. Thus, in the associated logit-based SOCPD instance, each customer class corresponds to one of the customer types and the customer type probability λ_k is the class k probability estimated via EM.

2. *Hierarchical Bayes*: For each dataset, we estimate a mixture multinomial logit (MMNL) model with a multivariate normal mixture distribution using the hierarchical Bayesian (HB) approach; we use a standard specification with normal-inverse Wishart second stage priors (see Section A.2.2 of the ecompanion for more details). We estimate this model using Markov chain Monte Carlo (MCMC) via the `bayesm` package in R [73]. We simulate 50,000 draws from the posterior distribution of $(\beta_{r,1}, \ldots, \beta_{r,n})$ for each re-

spondent r, and thin the draws to retain every 100th draw. Of those draws, we retain the last $J = 100$ draws, which we denote as $(\beta^j_{r,1}, \ldots, \beta^j_{r,n})$, where $j \in \{1, \ldots, J\}$, and we compute the average partworth vector $(\beta_{k,1}, \ldots, \beta_{k,n})$ as

$$(\beta_{r,1}, \ldots, \beta_{r,n}) = (\frac{1}{J}\sum_{j=1}^{J} \beta^j_{r,1}, \ldots, \frac{1}{J}\sum_{j=1}^{J} \beta^j_{r,n}). \tag{2.36}$$

This approach leads to an estimate of the partworths for each of the respondents. In the corresponding logit-based SOCPD instance, the number of customer types K corresponds to the number of respondents, and the probability of each customer type k is $1/K$.

Before continuing, we note that there may be other approaches for defining a mixture of logits model. (For example, given an estimate of the mean and covariance matrix of a normal mixture distribution defining a mixture logit model, one could sample a set of K partworth vectors and use those as the set of customer types, with each $\lambda_k = 1/K$.) We emphasize that our goal is not to advocate for one approach over another. The estimation approaches described here are simply for the purpose of obtaining problem instances that are of a realistic scale and correspond to real data. We note that our optimization approach is agnostic to how the customer choice model is constructed and is compatible with any estimation approach, so long as it results in a finite set of customer types that each follow a logit model of choice.

For each dataset, we define the set $\mathcal{A}$ to be the set of all binary vectors of size n that respect the attribute structure of the dataset; in particular, for attributes that are not binary, we introduce constraints of the form $\sum_{i \in S} a_i \leq 1$ as appropriate (cf. constraints (2.7) and (2.8) in Section 2.2.2). For `immigrant`, we also follow [45] in not allowing certain combinations of attributes (for example, it is not possible for a hypothetical immigrant to be a doctor and have only a high school education). We briefly describe the constraints for `immigrant` in Section A.2.3 of the ecompanion.

With regard to the no-purchase option, recall from Section 2.2.1 that the constant part of each customer's utility function, $\beta_{k,0}$, can be thought of as the negative of the utility of the no-purchase option. We assume that in each problem instance, each customer can choose from three different competitive offerings which are defined using the same attributes as the product that is to be designed. This is a standard assumption in the product design and product line design literature [13, 20, 21]. For each dataset, we provide the details of the competitive offerings in Section A.2.4 of the ecompanion. The parameter $\beta_{k,0}$ is then calculated in the same way as for the synthetic instances (see equation (2.35)).

Table 2.3 shows the computation time and the objective value of all of the different models for all four datasets; the columns containing the results follow the same structure as Table 2.1. From this table, we can see that all of the LC (latent class logit) instances can be solved to complete optimality within roughly 3 minutes. For the HB instances, MICP and GCG are able to solve the instances for `bank`, `candidate` and `timbuk2` within approximately 25 minutes. With regard to the geometric mean approach, we find that Mosek is able to solve all of the instances very quickly (within 20 seconds in all cases). In contrast to the synthetic instances, the geometric mean solution also tends to perform well, achieving a share-of-choice that is on average only about 14% below the best solution across all of the instances.

Of the instances in Table 2.3, the `immigrant` HB instance appears to be the most challenging. While Mosek is able to solve the MICP formulation to optimality within 2 hours, GCG exhausts the 2 hour time limit and returns a suboptimal solution with an optimality gap of about 12.6%. This performance is likely because the GCG method needs to add a very large number of constraints due to the large number of customer types. Moreover, the structure of the partworths appears to pose some numerical difficulty, as it appears that solutions that are near-optimal are such that the purchase probabilities are extremely close to 1 or 0 for a large number of customer types. This can be problematic because the constraints that GCG adds include terms of the form $\log(x_{k,1}) + 1$ and $\log(x_{k,0}) + 1$ (derivatives

			Objective Value			Computation Time (s)		
Dataset	Model	K	GCG	MICP	GM	GCG	MICP	GM
`bank`	LC	5	0.742	0.742	0.675	4.3	0.8	14.4
	LC	10	0.749	0.749	0.685	0.3	1.0	0.0
	LC	15	0.752	0.752	0.682	0.5	1.0	0.1
	LC	20	0.719	0.719	0.687	0.5	1.6	0.1
	LC	30	0.764	0.764	0.637	1.0	2.8	0.1
	LC	40	0.757	0.757	0.665	1.3	3.6	0.1
	LC	50	0.749	0.749	0.642	2.0	5.8	0.2
	HB	946	0.817	0.817	0.812	380.6	373.1	2.2
`candidate`	LC	5	0.626	0.626	0.509	1.4	4.8	0.0
	LC	10	0.694	0.694	0.637	2.9	9.6	0.1
	LC	15	0.670	0.670	0.651	4.5	13.2	0.1
	LC	20	0.705	0.705	0.574	8.2	16.3	0.1
	LC	30	0.627	0.627	0.534	29.0	53.2	0.1
	LC	40	0.671	0.671	0.537	31.4	89.6	0.3
	LC	50	0.710	0.710	0.680	192.0	102.5	0.4
	HB	311	0.852	0.852	0.851	1581.5	452.6	0.9
`immigrant`	LC	5	0.689	0.689	0.687	10.1	12.3	0.0
	LC	10	0.738	0.738	0.688	5.6	14.0	0.1
	LC	15	0.726	0.726	0.393	9.4	19.5	0.1
	LC	20	0.756	0.756	0.552	6.4	32.7	0.2
	LC	30	0.675	0.675	0.467	14.9	52.5	0.2
	LC	40	0.724	0.724	0.344	14.6	75.7	0.2
	LC	50	0.731	0.731	0.628	31.8	147.5	0.3
	HB	1396	0.836	0.865	0.846	7201.3	7053.0	4.7
`timbuk2`	LC	5	0.519	0.519	0.510	0.2	0.9	0.0
	LC	10	0.543	0.543	0.536	0.4	1.7	0.1
	LC	15	0.567	0.567	0.430	0.6	2.1	0.1
	LC	20	0.557	0.557	0.556	1.0	2.7	0.1
	LC	30	0.620	0.620	0.436	1.0	3.6	0.1
	LC	40	0.579	0.579	0.560	2.1	5.9	0.1
	LC	50	0.628	0.628	0.446	2.0	6.2	0.2
	HB	330	0.644	0.644	0.644	40.6	85.2	0.8

Table 2.3: Results for numerical experiment with real data.

of $x_{k,1}\log(x_{k,1})$ and $x_{k,0}\log(x_{k,0})$), which respectively blow up as $x_{k,1}$ and $x_{k,0}$ approach 0.

In addition to the performance of the different methods, it is also interesting to examine the optimal solutions. Table 2.4 visualizes the optimal solution for the `candidate` dataset for the LC model with $K = 20$ segments. The table also shows the three outside options/competitive offerings that were defined for this dataset. In addition, the table also shows the structure of a heuristic solution, which is obtained by finding the vector $\mathbf{a}$ in $\mathcal{A}$ that maximizes the average utility, i.e., that maximizes $\sum_{k=1}^{K}\lambda_k u_k(\mathbf{a})$.

From this table, we can see that the optimal solution matches some of the outside options on certain attributes (such as income and profession), but differs on some (for example, age). In addition, while the optimal solution does match the heuristic on many attributes, it differs on a couple of key attributes, namely race/ethnicity (the optimal candidate is Black, while the heuristic candidate is Asian American) and gender (the optimal candidate is male, while the heuristic candidate is female). While this may appear to be a minor difference, it results in a substantial difference in market share: the heuristic candidate attracts a share of 0.563, while the optimal candidate attracts a share of 0.705, which is an improvement of 25%. This illustrates that intuitive solutions to the logit-based product design problem can be suboptimal, and demonstrates the value of a principled optimization-based approach to this problem.

2.6 Conclusions

In this paper, we have studied the logit-based share-of-choice product design problem. While this problem is theoretically intractable, we show how it is possible to transform this problem into a mixed-integer convex optimization problem. Our transformation leverages an alternate characterization of logit choice probabilities arising from the representative agent model, which may be of utility in other optimization models involving logit models. We propose two practically viable approaches for solving this problem: one based on further reformulating

Attribute	Outside Option 1	Outside Option 2	Outside Option 3	Optimal Solution	Heuristic Solution
Age: 36	■				
Age: 45				■	■
Age: 52		■			
Age: 60					
Age: 68			■		
Age: 75					
Military Service: Did not serve	■		■		
Military Service: Served		■		■	■
Religion: None	■	■		■	■
Religion: Jewish					
Religion: Catholic			■		
Religion: Mainline protestant					
Religion: Evangelical protestant					
Religion Mormon					
College: No BA					
College: Baptist college					
College: Community college	■				
College: State university					
College: Small college		■			
College: Ivy League university			■	■	■
Income: 32K	■				
Income: 54K					
Income: 65K		■			
Income: 92K					
Income: 210K			■	■	■
Income 5.1M					
Profession: Business owner	■				
Profession: Lawyer			■	■	■
Profession: Doctor					
Profession: High school teacher		■			
Profession: Farmer					
Profession: Car dealer					
Race/Ethnicity: White	■				
Race/Ethnicity: Native American					
Race/Ethnicity: Black				■	
Race/Ethnicity: Hispanic		■			
Race/Ethnicity: Caucasian					
Race/Ethnicity: Asian American			■		■
Gender: Male	■	■		■	
Gender: Female			■		■

Table 2.4: Attributes of outside options, optimal solution and heuristic solution for `candidate` LC-MNL model with $K = 20$ segments.

the problem as a mixed-integer conic program involving the exponential cone, which can be directly solved by cutting edge solvers; and the other based on gradient-based constraint generation, which allows the problem to be solved as a mixed-integer linear program. We also show how our methodology can be extended to handle expected profit rather than share-of-choice, and how our methodology can also be used to optimize the geometric mean of the purchase probabilities, leading to an approximation algorithm for the original share-of-choice problem. Lastly, our numerics show how our approach can obtain high quality solutions to large instances, whether generated synthetically or from real conjoint data, within reasonable time limits. To the best of our knowledge, this is the first methodology for solving the logit-based share-of-choice product design problem to provable optimality.

CHAPTER 3

An Integer Programming Approach to Binary Decision Trees

Decision trees are graphs with a tree-like structure which map the input data to a leaf node that predicts the label of the output data. In a decision tree, for each observation, starting from the root, we proceed to the right or left node depending on the binary test applied in the current node. The label of the leaf node we reach following this process then becomes the prediction for the corresponding observation.

The main application area of decision trees is classification problems. By asking a set of questions, observations are partitioned and the leaves in the decision tree represent each one of these partitions. Decision trees are used in predicting the classes of observations by assigning labels to leaves. For example, Figure 3.1 illustrates a decision tree produced to identify high risk patients (R denotes high risk, N denotes not high risk) after a heart attack based on the initial 24-hour data [27]. Decision trees are also commonly used in policy learning which describes learning the rule of decision making that matches an individual to a treatment based on the characteristics of the individual. The treatment in policy learning can refer to a wide range of actions such as selecting offers, prices, advertisements, or emails to send to consumers, as well as the problem of determining which medication to prescribe to a patient [53, 89].

Decision trees have been widely used as predictive models, because of their simplicity and ability to model nonlinear relationships between the input data and the output data. They divide a complicated decision making problem into a collection of smaller problems

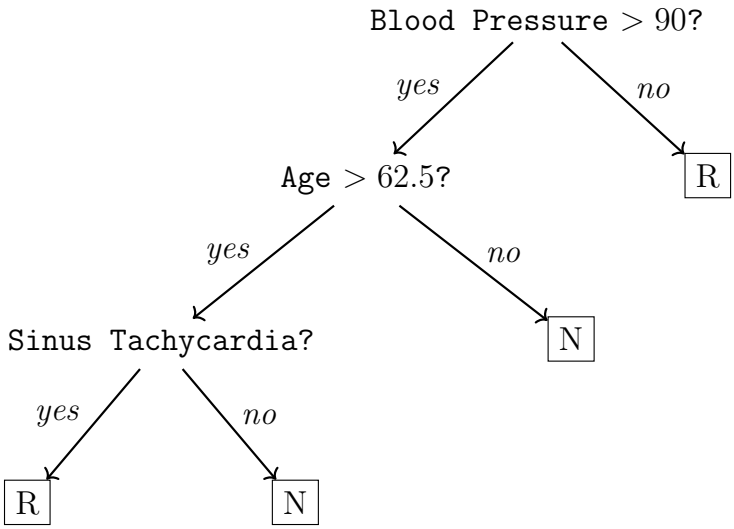


Figure 3.1: A decision tree example which classifies heart attack patients as under high risk (R) or not under high risk (N).

[75]; thus, they are especially appealing in healthcare, where interpretable decision making mechanisms are preferred. For example, decision trees can be used to design a risk calculator for emergency surgery [17]. Other application areas of decision trees include optimal stopping problems (e.g. when to start a price promotion in marketing, whether a patient should accept an available organ or wait for the next one in healthcare, etc.) [32], choice modeling [31, 7], rationalizing optimal policies to stochastic dynamic programs [26] and model extraction for interpreting complex, blackbox machine learning models [12]. Moreover, decision trees can deal with both categorical and real-valued variables; thus, they have a high modeling capability.

While decision trees are attractive due to their simplicity, interpretability and ability to model complex relationships between the dependent and independent variables, constructing decision trees from data is challenging. Although the idea behind constructing a decision tree is simple, the problem of finding the optimal binary tree is shown to be NP-hard [50]. As a result, given a training set, constructing a decision tree that represents it the best has been studied for years. The predominant approaches to learning decision trees are heuristic

in nature and most of them uses top-down induction, which builds the decision tree starting from the root node in a recursive manner. Although numerous heuristics have been developed to construct decision trees throughout the years, they have several disadvantages. First, these approaches lack optimality guarantees and usually, it is not even possible to tell how close the heuristic solution is to the optimal. Moreover, in many application areas, the decision maker may want to impose side constraints while learning a classification model or a policy, which need to be incorporated in the method used to construct the decision tree. Examples of such constraints can be preventing unfair discrimination based on protected characteristics [1] or limiting the number of people assigned to a policy out of budget concerns [89]. Adding these restrictions increases the computational complexity and the heuristics are usually unable to handle such side constraints.

Recently, spurred by advances in computing and solution capabilities of commercial mixed-integer optimization solvers [24, 19], the operations research community has proposed approaches to decision tree learning based on mixed-integer optimization (MIO). Such approaches are attractive, because they directly address the weaknesses of heuristic methods: MIO formulations not only produce solutions that are provably optimal, but allow the modeler to easily incorporate side constraints using additional constraints and variables in the formulations. Although a number of MIO formulations for the problem of decision tree learning formulations exist, these formulations are still not scalable to very large data sets. These formulations model splits in a continuous manner leading to formulations involving big-M constraints. Such constraints are known to result in formulations that are loose and may be challenging to solve provable optimality at a large scale.

In this paper, we propose a new MIO formulation of the decision tree learning problem. Our modeling framework is general in that it allows one to learn a decision tree that maps input data to one of a finite set of decisions; thus, our model can be used for binary and multi-class classification, as well as more complicated applications such as static policy learning. A critical aspect of our formulation is that it does not explicitly represent the split points of

the tree, since only the ordering of observations in terms of the variables is necessary while constructing the decision tree. This allows us to avoid big-M constraints for split conditions and obtain a stronger formulation compared to many previous models in the literature.

We make the following specific contributions:

1. We develop a new MIO formulation of the decision tree learning problem. Our formulation naturally extends to generalizations of decision trees beyond binary classification, such as multiclass classification and policy learning. Unlike the existing models in the literature, our formulation does not include big-M type of constraints and is stronger

2. We show theoretically that our formulation is stronger than the precedent MIO formulation that has been proposed in [16]. We also show that, in the special case of a single data point, our formulation is integral (i.e., all extreme points are integer), whereas the formulation in [16], in general, is not.

3. We propose a solution approach for scaling our formulation based on iteratively generating the constraints that link the splits of the tree to the assignment of observations to leaves; we show that it is possible to efficiently separate those constraints.

4. We demonstrate through numeric experiments that our formulation outperforms the precedent formulation in [16] in terms of computation time. Moreover, we show that our proposed large scale solution method improves the performance of the model in terms of solution time in several data sets.

The rest of the paper is structured as follows. In Section 3.1, we provide a review of the related literature. In Section 3.2, we propose a mixed-integer optimization model for decision tree learning, provide theoretical results and analyze the strength of our formulation. We propose a solution method to scale our formulation to larger data sets in Section 3.4. In Section 3.5, we present numerical experiments, which compares our model to the precedent

formulation in [16] and tests the performance of our proposed large scale method. Finally, in Section 3.6, we conclude.

3.1 Literature Review

Decision tree models are widely used in several disciplines such as machine learning and statistics for predictive and prescriptive modeling [27]. As the problem of finding the optimal binary tree is shown to be NP-hard [50], given a training set, constructing a decision tree that represents it the best has been a challenging problem for years and numerous heuristics have been developed to build a decision tree. For a review of the heuristics, we refer the reader to [72]. Moreover, with the increase in the computational power of mixed integer optimization (MIO) models and commercial MIO solvers in the recent years, several integer programming formulations of the decision tree problems have been developed. In Section 3.1.1, we go over the heuristics in the machine learning literature to construct decision trees and in Section 3.1.2, we present a review of integer programming approach to decision trees.

3.1.1 Top-down Approach and Other Heuristics

The most common heuristic approach to decision tree learning is top-down induction. The top-down induction approach involves starting with an initial tree, which is typically The seminal work of [27] which proposed classification and regression tree (CART) algorithm is one of the main examples of top-down approach. Other notable heuristics that use top-down approach include C4.5 [70] and ID3 [69]. Top-down approach has a greedy nature and usually includes two steps: growing the tree and pruning to control the size of the tree and avoid overfitting [72]. The main disadvantage of the top-down approach is that when growing the tree, it considers each split in isolation and makes the best decision for that split only. However, it is possible to obtain a better split by choosing a worse classifier in a previous split. Top-down approach fails to capture a tradeoff like this and does not result in globally

optimal trees [14]. Moreover, in top-down approach, usually an impurity measure, which takes several factors into consideration, is used to grow the tree, whereas misclassification rate is used as the criterion for pruning [27]. Therefore, misclassification rate, which is more consistent with the objective of the problem, is not used when the split decisions are made.

A weakness of the top-down induction approach is that each split is chosen without consideration of future splits that may be chosen for the leaves that result from that split. As a result, there has been research that has considered lookahead techniques, which attempt to improve the top-down approach by predicting the effect of a split decision on the descendant nodes. Several lookahead heuristics that have been proposed in the machine learning literature are IDX [68], LSID3 and ID3-k [37]. Lookahead heuristics are reported to have mixed performance results, where they outperform greedy heuristics in some data sets [68] and produce larger and less accurate trees in others [67].

3.1.2 Mixed Integer Formulations

In contrast to heuristic approaches, there has been a significant and growing interest in the operations research community in the last five years in the application of mixed-integer programming models to the problem of learning a binary decision tree. In this body of literature, the landmark paper of [16] was the first to study the problem of learning a classification tree using mixed-integer optimization. This paper formulates the problem of learning a decision tree as an integer program and aim to achieve global optimality and eliminate the need for pruning by solving the model in one step. Our work relates to this paper in attempting to construct an optimal binary decision tree in a single step using an integer program. However, our model is differentiated from the model of Bertsimas and Dunn [16] in two significant dimensions. First, that model incorporates decision variables that determine whether to branch in a node or not; thus, the size of the tree and the tree topology is determined by their model. In addition, the objective function of the Bertsimas and Dunn [16] model is motivated by the objective used in CART: in particular, to capture

the tradeoff between the complexity of the tree and accuracy, the model add a penalty to the objective function using the complexity parameter. Our model differs from the Bertsimas and Dunn [16] model by assuming the tree topology is given and using the misclassification rate as the objective function. Second, Bertsimas and Dunn [16] formulation uses a continuous decision variable to represent each split's value, which leads to big-M constraints; thus, the formulation can potentially have a weak relaxation and be difficult to scale [85]. Since the actual values of the independent variables never appear in our formulation, we avoid big-M constraints and propose a stronger formulation than Bertsimas and Dunn [16]. As we will see later in Section 3.5, our formulation can be solved to a lower optimality gap than the formulation of Bertsimas and Dunn [16] within a fixed computational time limit. Since the paper of [16], [53] and [89] developed similar MIO formulations for policy learning problems and [1] developed a similar MIO formulation for designing a decision tree with concerns over fairness in decision making.

Since the paper of Bertsimas and Dunn [16], a number of other papers have considered alternate formulations for the problem of learning a binary decision tree. [84] develop a MIO formulation which aims to reduce the number of binary variables compared to the formulation of Bertsimas and Dunn [16]. They use binary encoding to model the split conditions and propose a formulation where the number of decision variables depends on the maximum amount of unique values among all features, instead of the number of observations in the data. Although their formulation has a smaller number of binary variables, the constraints that establish the split conditions still rely on big-M constants.

[44] develop a MIO formulation specifically tailored for binary classification when all the independent variables are categorical variables. They represent the categorical data as a binary vector and each split checks whether the assigned features in the binary vector belonging to an observation is 0 or 1. The decision mechanism in each split in [44] is different than our formulation, where each split classifies an observation based on whether the assigned feature's value is greater or less than the split point. Since the binary test

in each split in [44] can be reversed without affecting the optimal solution, this decision mechanism allows them to exploit the symmetry in decision tree and preassign the labels to leaves before solving the optimization model in binary classification case. Our modeling framework is similar to [44] in assuming the tree topology is given. However, a key difference is that our formulation accommodates both real-valued numeric variables and categorical variables (through one-hot encoding). Also, while their formulation is restricted to binary classification, we develop a more general model which can handle multi-class classification as well as policy learning problems with more than two treatments.

In a concurrent and independent work, [2] develop a MIO formulation to construct optimal decision trees for binary valued data sets. They use a flow-based approach by transforming a decision tree into an acyclic directed graph with a single source and sink node. If we use a one-hot encoding approach to transform non-binary valued data into binary valued data, i.e., for each non-binary value in a column, if we create a new column which has the value of 1 for the corresponding value and 0 for all others, their formulation is equivalent to ours. The paper additionally proposes a Benders decomposition algorithm which exploits the max-flow structure in the subproblem and provides polyhedral results on the convex hull of the feasible region of decision trees. Our work differ from theirs in developing a constraint generation based large scale method and proposes additional structural results.

3.2 Model

In this section, we develop a new MIO to construct decision trees and prove several properties of it. The remainder of this section is organized as follows. In Section 3.2.1, we define the parameters used in MIO. Then, we provide the formulation in Section 3.2 and show some structural properties of the model in Section 3.2.3.

3.2.1 Problem Definition

We begin by first defining the data, and then defining the decision tree. To represent the data, let $\mathbf{X} = (X_1, \ldots, X_d)$ be the vector of independent variables that comprise an observation. We assume that each independent variable $X_1, \ldots, X_d$ is continuous/numeric, i.e., $X_1, \ldots, X_d \in \mathbb{R}$. Let n be the number of observations in the data, and $\mathbf{X}^1, \ldots, \mathbf{X}^n$ denote the independent variable vectors for the n observations. For a particular independent variable i, $X_i^1, \ldots, X_i^n$ are the values of that independent variable in the data.

We now define the components of the decision tree. We denote the set of leaf nodes and the set of split nodes in the tree by **leaves** and **splits**, respectively. For a given split $s \in \mathbf{splits}$, we define the functions $\mathbf{leftchild} : \mathbf{splits} \to \mathbf{splits} \cup \mathbf{leaves}$ and $\mathbf{rightchild} : \mathbf{splits} \to \mathbf{splits} \cup \mathbf{leaves}$ as the mappings of each split node to its left child node and right child node, respectively; given a split node s, $\mathbf{leftchild}(s)$ is the left child node of s, and $\mathbf{rightchild}(s)$ is the right child node of s. We let the tuple $\mathcal{T} = (\mathbf{leaves}, \mathbf{splits}, \mathbf{leftchild}, \mathbf{rightchild})$ denote the topology of the tree. We assume that the topology $\mathcal{T}$ is given and fixed.

We use $v(s)$ to denote the split variable index of split $s \in \mathbf{splits}$, and $\mathbf{v} = (v(s))_{s \in \mathbf{splits}}$ denote the vector of split point indices. We also $\theta(s)$ to denote the split point in split $s \in \mathbf{splits}$, and $\boldsymbol{\theta} = (\theta(s))_{s \in \mathbf{splits}}$ to denote the vector of split points. Given an observation that arrives at split s, a query in the form "Is $X_{v(s)} \leq \theta(s)$?" is applied to the observation. If the query is true, the observation is mapped to $\mathbf{leftchild}(s)$; otherwise, the observation is mapped to $\mathbf{rightchild}(s)$.

Given a tree defined by $\mathcal{T}$, $\mathbf{v}$ and $\boldsymbol{\theta}$, the decision tree maps the observation to a leaf by the following process. Starting at the root node, we check if the node is a leaf. If it is a leaf node, we terminate and we output that node. Otherwise, the node is a split node s, and we check the split's query $X_{v(s)} \leq \theta(s)$, and based on the outcome of the query, we proceed to the left or the right node. This process then repeats at the new node, until a

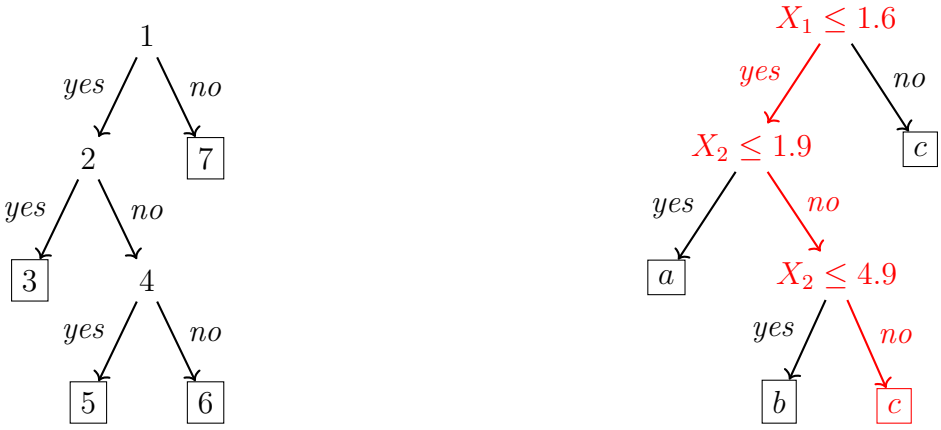
leaf is reached. Given a topology $\mathcal{T}$, split variable indices $\mathbf{v}$ and split point vector $\boldsymbol{\theta}$, we use $\ell(\mathbf{X}; \mathcal{T}, \mathbf{v}, \boldsymbol{\theta})$ to denote the leaf to which an observation with feature vector $\mathbf{X}$ is mapped to. We let $a(\ell)$ denote the action that corresponds to leaf ℓ, which is chosen from a finite action space $\mathcal{A}$; once an observation reaches a leaf ℓ, the decision tree outputs the action $a(\ell)$. We let $\mathbf{a} = (a(\ell))_{\ell \in \mathbf{leaves}}$ denote the vector of leaf-to-action assignments.

Example 1 *Figure 3.2 provides an example of a decision tree. In this example, each observation consists of two features, i.e.,* $\mathbf{X} = (X_1, X_2)$*, and the action set consists of three actions,* $\mathcal{A} = \{\alpha, \beta, \gamma\}$*. The left-hand side of the figure displays the topology* $\mathcal{T}$ *of a tree, with the nodes indexed from 1 to 6; here,* $\mathbf{leaves} = \{3, 5, 6, 7\}$ *and* $\mathbf{splits} = \{1, 2, 4\}$*, and we have* $\mathbf{leftchild}(1) = 2, \mathbf{leftchild}(2) = 3, \mathbf{leftchild}(4) = 5, \mathbf{rightchild}(1) = 7, \mathbf{rightchild}(2) = 4, \mathbf{rightchild}(4) = 6$*. The right-hand side figure displays the decision tree, which displays the query for each split, and the action of each leaf; in this example, the split variable indices are* $v(1) = 1, v(2) = 2, v(4) = 2$*, the split points are* $\theta(1) = 1.6, \theta(2) = 1.9, \theta(4) = 4.9$*, and the leaf actions are* $a(3) = \alpha, a(5) = \beta, a(6) = \gamma, a(7) = \gamma$*.*

To illustrate how the tree maps an observation to a leaf and returns an action, suppose that $\mathbf{X} = (X_1, X_2) = (1.6, 5.1)$*. The red path shown in the right-hand side figure of Figure 3.2 illustrates how this observation is mapped to leaf 6. The action that is returned by the tree is then the action* $a(6)$*, which is* γ*.*

Having defined the essential components of a decision tree, we now define the decision tree learning problem. We assume that each assignment of an observation to an action carries a reward. We let $c_{m,a}$ denote the reward of taking action a for observation m, which we assume is fixed and known to us. For a fixed topology $\mathcal{T}$, we define the decision tree learning problem as the problem of finding the split variables $\mathbf{v}$, the split points $\boldsymbol{\theta}$ and the leaf actions $\mathbf{a}$ so as to maximize the sum of the rewards garnered from the observations; we formally define this problem as

$$\underset{\mathbf{v}, \boldsymbol{\theta}, \mathbf{a}}{\text{maximize}} \quad \sum_{m=1}^{n} c_{m, a(\ell(\mathbf{X}^m; \mathcal{T}, \mathbf{v}, \boldsymbol{\theta}))} \tag{3.1}$$



(a) Example of tree topology $\mathcal{T}$.

(b) Example of a complete binary decision tree.

Figure 3.2: Example of a tree topology $\mathcal{T}$ (left-hand subfigure) and a complete decision tree (right-hand subfigure).

3.2.2 Mixed-Integer Optimization Formulation

Problem (3.1) is a complicated combinatorial optimization problem. To solve it, we will model it as a mixed-integer optimization (MIO) problem. We begin by defining a set of parameters that are needed to construct our formulation. We define the parameter ω_{ij} as the jth lowest unique value of variable i in the training set $\mathbf{X}^1, \ldots, \mathbf{X}^n$. Let J_i be the number of unique values of the ith independent variable, so that, j ranges from 1 to J_i. We thus have:

$$\omega_{i,1} < \omega_{i,2} < \cdots < \omega_{i,J_i-1} < \omega_{i,J_i}.$$

For convenience, we also define $\omega_{i,0} = -\infty$.

Let $\tau_{i,m}$ be the value of $j \in \{1, \ldots, J_i\}$ such that $X_i^m = \omega_{i,\tau_{i,m}}$; i.e., for observation m, it tells us where that observation's value of feature i is in the overall ranking of all unique values of that feature.

We now define the decision variables of the problem. For each split s, feature i and index j, we let $\lambda_{s,i,j}$ be a binary decision variable that is 1 if split s uses feature i and the split

point $\theta_s = \omega_{i,j}$ and 0 otherwise. To track the leaf an observation is mapped into, for each observation m and leaf ℓ we let $y_{m,\ell}$ be a binary decision variable that is 1 if the tree maps observation m to leaf ℓ, and 0 otherwise. We let $\pi_{\ell,a}$ be a binary decision variable that is 1 if leaf ℓ prescribes action a, and 0 otherwise. Finally, we define the binary decision variable $w_{m,\ell,a}$ as 1 if observation m is mapped to leaf ℓ and action a is assigned to leaf ℓ, and 0 otherwise.

With these definitions, we define our formulation below.

$$\underset{\mathbf{y},\boldsymbol{\lambda},\mathbf{w},\boldsymbol{\pi}}{\text{maximize}} \quad \sum_{m=1}^{n} \sum_{\ell \in \mathbf{leaves}} \sum_{a \in \mathcal{A}} c_{m,a} w_{m,\ell,a} \tag{3.2a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}} y_{m,\ell} = 1, \quad \forall m \in [n], \tag{3.2b}$$

$$\sum_{i=1}^{d} \sum_{j=1}^{J_i} \lambda_{s,i,j} = 1, \quad \forall s \in \mathbf{splits}, \tag{3.2c}$$

$$\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} \leq 1 - \sum_{\ell \in \mathbf{left}(s)} y_{m,\ell} \quad \forall s \in \mathbf{splits}, \quad m \in [n], \tag{3.2d}$$

$$\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} \leq 1 - \sum_{\ell \in \mathbf{right}(s)} y_{m,\ell} \quad \forall s \in \mathbf{splits}, \quad m \in [n], \tag{3.2e}$$

$$\sum_{a \in \mathcal{A}} \pi_{\ell,a} = 1, \quad \forall \ell \in \mathbf{leaves}, \tag{3.2f}$$

$$y_{m,\ell} = \sum_{a \in \mathcal{A}} w_{m,\ell,a}, \quad \forall m \in [n], \quad \ell \in \mathbf{leaves}, \tag{3.2g}$$

$$w_{m,\ell,a} \leq \pi_{\ell,a}, \quad \forall m \in [n], \quad \ell \in \mathbf{leaves}, \quad a \in \mathcal{A}, \tag{3.2h}$$

$$y_{m,\ell} \in \{0,1\}, \quad \forall m \in [n], \quad \ell \in \mathbf{leaves}, \tag{3.2i}$$

$$\pi_{\ell,a} \geq 0, \quad \forall \ell \in \mathbf{leaves}, \quad a \in \mathcal{A}, \tag{3.2j}$$

$$w_{m,\ell,a} \geq 0, \quad \forall m \in [n], \quad \ell \in \mathbf{leaves}, \quad a \in \mathcal{A} \tag{3.2k}$$

$$\lambda_{s,i,j} \geq 0, \quad \forall s \in \mathbf{splits}, \quad i \in [d], \quad j \in \{0, \ldots, J_i\}. \tag{3.2l}$$

The constraints of this formulation have the following meaning. Constraint (3.2c) ensures that exactly one variable and exactly one split point is assigned to each split s. Con-

straint (3.2b) ensures each observation is mapped to exactly one leaf. To understand constraint (3.2d), observe that the left-hand side is 1 if and only if we choose a split variable i for split s and a split point θ_s that is greater than $\omega_{i,\tau_{i,m}} = X_i^m$, while the right hand side is 1 if and only if the observation m is not mapped to a leaf to the left of split s. Thus, the constraint states that if we choose split variable i for split s and we choose a split point $\theta_s < \omega_{i,\tau_{i,m}}$, then observation m cannot be mapped to any leaf to the left of split s. Constraint (3.2e) can be interpreted in a similar way, requiring that if we choose a split variable i for split s and we choose a split point $\theta_s \leq \omega_{i,\tau_{i,m}}$, then observation m cannot be mapped to any leaf to the right of split s. Constraint (3.2f) ensures that exactly one action is assigned to each leaf. Constraints (3.2g) and (3.2h) ensure that the $w_{m,\ell,a}$ variables are correctly defined, that is, $w_{m,\ell,a} = 1$ if and only if $y_{m,\ell} = 1$ and $\pi_{\ell,a} = 1$. Finally, constraints (3.2i)-(3.2l) define decision variables to be binary and continuous. The objective function of the model is the sum of the rewards obtained from the actions that are assigned to the observations.

We pause here to comment on a few important characteristics of our model. First, we note that our formulation as we have defined it above restricts the split point of each split to be one of the unique values of one of the d features, i.e., $\omega_{i,j}$ for some choice of feature i and some $j \in [J_i]$. We note that this is without loss of generality, because there are finitely many observations, and for any feature, there is an interval of split points that are equivalent in how they map each observation to the left or to the right. In particular, observe that a split of the form $X_i \leq \theta$ has exactly the same behavior for any choice of $\theta \in [\omega_{i,j}, \omega_{i,j+1})$. Thus, we do not lose anything by restricting the choice of split point for a split on feature i to the values $\omega_{i,0}, \ldots, \omega_{i,J_i}$.

Building on the previous point, this brings us to the second important characteristic of our model. In our formulation, we do not need to explicitly model the values of the features of the observations and the value of the split point. Instead, we use the fact that only the relative ordering of the observations in terms of the features is important in determining the

behavior of a decision tree. Our formulation takes advantage of this and does not require the raw data to enter the constraints. This modeling approach provides two advantages over the existing formulations in the literature. First, many prior formulations (in particular, [16] and [84]), rely on big-M constraints to model how an observation gets mapped to the left or to the right of a split; as a result, these formulations can be difficult to solve at large scales. Unlike these prior formulations, our model that can handle real-valued variables without big-M constraints.

The third important characteristic of our model is that several binary decision variables can actually be relaxed to be continuous; in particular, note that variables $\boldsymbol{\lambda}$, $\boldsymbol{\pi}$ and $\mathbf{w}$ are defined as continuous variables. In the next section, we will show that defining $\mathbf{y}$ to be binary suffices to maintain the validity of the formulation and that the $\boldsymbol{\lambda}$, $\boldsymbol{\pi}$ and $\mathbf{w}$ variables will take their correct values when $\mathbf{y}$ is binary, allowing us to safely relax these variables.

Finally, we note that constraints (3.2d) and (3.2e) can be written in the independent branching form [87, 49]. Independent branching scheme is a modeling approach for disjunctive constraints based on a representation of these constraints as a series of choices between several alternatives. This way of modeling disjunctive constraints result in small and strong formulations. In particular, the formulation of a pairwise independent branching is known to be ideal [86, 87]. In problem (3.2), the split conditions for each split can be expressed using the independent branching scheme in the following way. Let $\tilde{y}^L_{s,m}$ and $\tilde{y}^R_{s,m}$ be decision variables which take the value 1 if observation m goes to left and right from split s, respectively, and 0 otherwise. We also define a decision variable $\tilde{y}^N_{s,m}$ which takes the value 1 if observation m does not arrive split s, and 0 otherwise. Observe that, this implies $\tilde{y}^L_{s,m} = \sum_{\ell \in \mathbf{left}(s)} y_{m,\ell}$ and $\tilde{y}^R_{s,m} = \sum_{\ell \in \mathbf{right}(s)} y_{m,\ell}$. We can now add constraints that define the new decision variables and rewrite constraints (3.2d) and (3.2e) for each split in the following

way:

$$\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} \leq 1 - \tilde{y}^L_{s,m}, \quad \forall m \in [n], \tag{3.3a}$$

$$\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} \leq 1 - \tilde{y}^R_{s,m}, \quad \forall m \in [n], \tag{3.3b}$$

$$\tilde{y}^L_{s,m} + \tilde{y}^R_{s,m} + \tilde{y}^N_{s,m} = 1, \quad \forall m \in [n], \tag{3.3c}$$

$$\tilde{y}^L_{s,m}, \tilde{y}^R_{s,m}, \tilde{y}^N_{s,m} \in \{0,1\}, \quad \forall m \in [n]. \tag{3.3d}$$

If the split conditions are established as we described above, constraint system (3.3) along with constraint (3.2c) are in the form of an independent branching scheme for each split. Specifically, the constraints construct a pairwise independent branching scheme for the root split and 3-way independent branching scheme for the following splits. The advantage of independent branching scheme is that it is able to fix a set of components of λ to zero independent of the previous branching decisions and may simplify branching rules in the branch-and-bound tree [49]. It therefore has the potential to improve the performance of MIP solvers. However, despite its computational advantages, a formulation with constraints (3.3) has more decision variables than formulation (3.2). We thus retain formulation (3.2) in the remaining of the paper.

3.2.3 Structural Properties

In this section, we discuss several structural properties of formulation (3.2). In particular, formulation (3.2) has a number of decision variables that are originally defined as binary variables, but actually can be safely relaxed. Our first result establishes that this relaxation is without any loss of validity to the formulation.

Proposition 3 *There exists an optimal solution* $(\mathbf{y}, \boldsymbol{\lambda}, \mathbf{w}, \boldsymbol{\pi})$ *of problem* (3.2) *such that* $\boldsymbol{\lambda}$, $\mathbf{w}$ *and* $\boldsymbol{\pi}$ *are all binary.*

The proof (see Section B.1.1 of the ecompanion) is divided in two parts. In the first part,

we show that there exists an optimal solution with $\mathbf{w}$ and $\boldsymbol{\pi}$ being binary, which uses the fact that when $\mathbf{y}$ is fixed, problem (3.2) can be divided into $|\mathbf{leaves}|$ independent optimization problems over $\mathbf{w}_\ell$ and $\boldsymbol{\pi}_\ell$ for which an optimal solution involves setting exactly one $\pi_{\ell,a}$ to 1 and the rest to zero, which forces $w_{m,\ell,a}$ to be either 0 or 1. In the second part of the proof, we show that when $\mathbf{y}$ is fixed, the set of feasible $\boldsymbol{\lambda}$ is a polyhedron defined by a totally unimodular constraint matrix. Thus, when $\mathbf{y}$ is binary, there exists a feasible $\boldsymbol{\lambda}$ that is an extreme point and therefore also binary.

We note here that our formulation enforces $\mathbf{y}$ to be binary and allows $\boldsymbol{\lambda}$ to be continuous. Alternatively, one can consider allowing $\mathbf{y}$ to be continuous and $\boldsymbol{\lambda}$ to be binary. Our next result, Proposition 4, establishes that whenever $\boldsymbol{\lambda}$ is binary, the constraints of the formulation force the $\mathbf{y}$ variables to their correct binary values.

Proposition 4 *Let* $\boldsymbol{\lambda} \in \{0,1\}^{\mathbf{splits}\times[d]\times\{0,\dots,J_i\}}$ *be a binary vector that satisfies constraint* (3.2c). *Consider the split variable index vector* $\mathbf{v}$ *and the split point vector* $\boldsymbol{\theta}$ *defined as*

$$v(s) = \sum_{i=1}^{d} i \cdot \left[\sum_{j=0}^{J_i} \lambda_{s,i,j} \right], \quad \forall s \in \mathbf{splits},$$

$$\theta(s) = \sum_{i=1}^{d} \sum_{j=0}^{J_i} \omega_{i,j} \lambda_{s,i,j}, \quad \forall s \in \mathbf{splits}.$$

Let $\ell_m^* = \ell(\mathbf{X}^m; \mathcal{T}, \mathbf{v}, \boldsymbol{\theta})$ *be the leaf that observation* m *is mapped to according to* $(\mathcal{T}, \mathbf{v}, \boldsymbol{\theta})$. *Let* $\mathbf{y}$ *be defined as*

$$y_{m,\ell} = \begin{cases} 1 & \text{if } \ell = \ell_m^*, \\ 0 & \text{otherwise.} \end{cases}$$

Then, $\mathbf{y}$ *is the only solution that satisfies constraints* (3.2b), (3.2d) *and* (3.2e).

The proof of Proposition 4 follows similarly to the proof of Proposition 4 of [64]. An immediate consequence of Proposition 4 is the following corollary.

Corollary 1 *Consider the following alternate form of problem* (3.2) *where* $\boldsymbol{\lambda}$ *is constrained to be binary, and* $\mathbf{y}$ *is relaxed:*

$$\underset{\mathbf{y},\boldsymbol{\lambda},\mathbf{w},\boldsymbol{\pi}}{\textit{maximize}} \quad \sum_{m=1}^{n} \sum_{\ell \in \textbf{leaves}} \sum_{a \in \mathcal{A}} c_{m,a} w_{m,\ell,a} \tag{3.4a}$$

$$\textit{subject to} \quad (3.2b) - (3.2h)$$

$$y_{m,\ell} \geq 0, \quad \forall m \in [n], \quad \ell \in \textbf{leaves}, \tag{3.4b}$$

$$\pi_{\ell,a} \geq 0, \quad \forall \ell \in \textbf{leaves}, \quad a \in \mathcal{A}, \tag{3.4c}$$

$$w_{m,\ell,a} \geq 0, \quad \forall m \in [n], \quad \ell \in \textbf{leaves}, \quad a \in \mathcal{A} \tag{3.4d}$$

$$\lambda_{s,i,j} \in \{0,1\}, \quad \forall s \in \textbf{splits}, \quad i \in [d], \quad j \in \{0,\ldots,J_i\}. \tag{3.4e}$$

There exists an optimal solution $(\mathbf{y}, \boldsymbol{\lambda}, \mathbf{w}, \boldsymbol{\pi})$ *of problem* (3.4) *such that* $\mathbf{y}$*,* $\boldsymbol{\lambda}$*,* $\mathbf{w}$ *and* $\boldsymbol{\pi}$ *are all binary.*

The value of this corollary is that it furnishes us with two different ways of relaxing our optimization problem. The first approach is to relax all of the variables except for $\mathbf{y}$, as done in problem (3.2). The second approach is to relax all of the variables except for $\boldsymbol{\lambda}$, as done in problem (3.4). These two approaches can be advantageous in different settings. For example, for data sets where there is a large number of observations but the number of unique split points across all of the features is small, it may be advantageous to use problem (3.4) as there will be fewer $\lambda_{s,i,j}$ variables than $y_{m,\ell}$ variables to branch on when solving the problem using branch-and-bound. Similarly, when there is a small number of observations but a large number of features, it is more advantageous to use problem (3.2) as there will be fewer $y_{m,\ell}$ variables than $\lambda_{s,i,j}$ variables.

3.3 Comparison to Bertsimas and Dunn [16] Formulation

In this section, we compare our formulation to the precedent formulation for decision tree in [16].

3.3.1 Bertsimas and Dunn [16] Formulation

We begin by defining the Bertsimas and Dunn [16] formulation. Let **leaves** and **splits** be defined in the same way as in our formulation and $\mathcal{A}$ be the finite set of actions. For a node $t \in \textbf{leaves} \cup \textbf{splits}$, let $p(t)$ denote the parent node of node t. Let $A_L(t)$ and $A_R(t)$ denote the set of ancestors of t whose left and right branches have been followed on the path from the root node to node t, respectively. We define N_{min} as the minimum number of observations that should be mapped to any leaf. To penalize the incorrect classifications, for all $m \in [n]$ and $a \in \mathcal{A}$, the parameter $\bar{Y}_{m,a}$ is defined as follows:

$$\bar{Y}_{m,a} = \begin{cases} +1 & \text{if } Y^m = a, \\ -1 & \text{otherwise.} \end{cases}$$

We have the following decision variables. Let d_s be a binary decision variable, which is 1 if node s applies a split and 0 otherwise. Let α_ℓ be a binary decision variable, which is 1 if leaf ℓ contains any points and 0 otherwise. Let $f_{s,i}$ be a binary decision variable, which is 1 if feature i is assigned to split s and 0 otherwise. Let b_s be the decision variable that denotes the value of the split variable assigned to split s. We define $N_{a,\ell}$ as the number of points with action a in leaf ℓ and $\bar{N}_\ell$ as the total number of points assigned to leaf ℓ. Let M_ℓ denote the number of points misclassified. The binary decision variables $y_{m,\ell}$ and $\pi_{\ell,a}$ are defined in the same way as our formulation.

Note that, the split rule in Bertsimas and Dunn [16] formulation is of the form $X_i < \theta_s$, where i is the variable assigned to split s, whereas in our formulation, the split rule is of the form $X_i \leq \theta_s$. Since the difference in the definitions does not affect the way models work, we modify the Bertsimas and Dunn [16] model to conform with the split rule in our formulation, in order to facilitate the comparison between the models. In addition, the paper of Bertsimas and Dunn [16] assumes that all of the features are normalized to lie between 0 and 1; we shall make the same assumption.

The MIO formulation is as follows:

$$\underset{\mathbf{y},\boldsymbol{\pi},\mathbf{M},\bar{\mathbf{N}},\mathbf{N},\mathbf{f},\mathbf{b}}{\text{minimize}} \quad \sum_{\ell \in \textbf{leaves}} M_\ell \tag{3.5a}$$

$$\text{subject to} \quad M_\ell \geq \bar{N}_\ell - N_{a,\ell} - n(1-\pi_{\ell,a}), \quad \forall a \in \mathcal{A}, \quad \ell \in \textbf{leaves}, \tag{3.5b}$$

$$M_\ell \leq \bar{N}_\ell - N_{a,\ell} + n\pi_{\ell,a}, \quad \forall a \in \mathcal{A}, \quad \ell \in \textbf{leaves}, \tag{3.5c}$$

$$M_\ell \geq 0, \quad \forall \ell \in \textbf{leaves}, \tag{3.5d}$$

$$N_{a,\ell} = \frac{1}{2}\sum_{m=1}^{n}(1+\bar{Y}_{m,a})y_{m,\ell}, \quad \forall a \in \mathcal{A}, \quad \ell \in \textbf{leaves}, \tag{3.5e}$$

$$\bar{N}_\ell = \sum_{m=1}^{n} y_{m,\ell}, \quad \forall \ell \in \textbf{leaves}, \tag{3.5f}$$

$$\sum_{i=1}^{d} f_{s,i}X_i^m \geq b_s - 1 + (1+\epsilon)y_{m,\ell}, \quad \forall m \in [n],\, \ell \in \textbf{leaves},\, s \in A_R(\ell), \tag{3.5g}$$

$$\sum_{i=1}^{d} f_{s,i}X_i^m \leq b_s + 1 - y_{m,\ell}, \quad \forall m \in [n], \ell \in \textbf{leaves}, s \in A_L(\ell), \tag{3.5h}$$

$$\sum_{\ell \in \textbf{leaves}} y_{m,\ell} = 1, \quad \forall m \in [n], \tag{3.5i}$$

$$\sum_{i=1}^{d} f_{s,i} = 1, \quad \forall s \in \textbf{splits} \tag{3.5j}$$

$$0 \leq b_s \leq 1, \quad \forall s \in \textbf{splits}, \tag{3.5k}$$

$$f_{s,i} \in \{0,1\}, \quad \forall s \in \textbf{splits}, \quad i \in [d], \tag{3.5l}$$

$$y_{m,\ell} \in \{0,1\}, \quad \forall m \in [n], \quad \ell \in \textbf{leaves}, \tag{3.5m}$$

$$\pi_{\ell,a} \in \{0,1\}, \quad \forall \ell \in \textbf{leaves}, \quad a \in \mathcal{A}. \tag{3.5n}$$

In this formulation, constraints (3.5b)-(3.5d) characterizes the number of observations misclassified in each leaf. Constraint (3.5e) finds the number of number of points with action a in leaf ℓ for all $a \in \mathcal{A}$ and $\ell \in \textbf{leaves}$. Constraint (3.5f) finds the total number of observations assigned to each leaf. Constraints (3.5g) and (3.5h) establish the split conditions. Constraint (3.5i) ensures each observation is mapped to exactly one leaf. Constraint (3.5j) ensures exactly one variable is assigned to a split. Constraint (3.5k) ensures that split

value for a node is 0 if that node does not apply a split. Constraints (3.5l)-(3.5n) defines the binary decision variables.

A few comments are in order. First, the key difference between formulation (3.5) and our formulation (3.2) is how the splits are modeled. In particular, in problem (3.5), the splits are modeled using continuous variables to represent the continuous split point and big-M constraints (see constraints (3.5g) and (3.5h)). This is strikingly different to our formulation (3.2), which models split points in a discrete fashion. This modeling choice can potentially lead to a weaker formulation; as we will show in Section 3.3.2, our formulation (3.2) is at least as strong as formulation (3.5). Having said this, formulation (3.5) has the advantage that it is a smaller formulation than ours: in our formulation, the number of $\lambda_{s,i,j}$ variables needed to model the choice of split point for split s effectively grows with the number of observations, whereas in formulation (3.5), there is only one continuous variable b_s needed to model the choice of split point. For some instances, it is possible that the reduced size of the formulation compensates for the loss of tightness by allowing the nodes in the branch-and-bound tree to be solved more quickly, and in such cases, it is possible that formulation (3.5) can be solved more quickly than formulation (3.2). In our numerical experiments with real classification data sets in Section 3.5, we will see that in general, our formulation (3.2) performs better in terms of solution time and optimality gap than formulation (3.5). Besides the advantage of a smaller formulation, formulation (3.5) also has the advantage that it can be modified relatively easily to allow for oblique/hyperplane splits of the form "Is $\mathbf{g}^T\mathbf{X} \leq \theta$?", where $\mathbf{g} \in \mathbb{R}^d$, which are more general than the univariate/rectilinear splits of the form "Is $X_i \leq \theta$?" used by our formulation. It does not appear to be straightforward to adapt our formulation (3.2) to allow for oblique splits.

Second, as noted above, the formulation (3.5) differs from the precise formulation given in [16] slightly. There are three points of difference:

1. **Optimization over the tree topology**. The first difference is that the formulation of Bertsimas and Dunn [16] incorporates an additional collection of decision variables

which model whether or not a split node is included in the tree topology or not. Formally, this is modeled by adding a binary decision variable d_s which is 1 if split s is used in the tree and 0 if not, and replacing constraints (3.5j)-(3.5k) with the following three constraints:

$$\sum_{i=1}^{p} f_{s,i} = d_s, \quad \forall s \in \textbf{splits} \tag{3.6}$$

$$0 \leq b_s \leq d_s, \quad \forall s \in \textbf{splits}, \tag{3.7}$$

$$d_s \leq d_{p(s)}, \quad \forall s \in \textbf{splits} \setminus \{1\}, \tag{3.8}$$

In addition, the original formulation of Bertsimas and Dunn [16] also includes a term in the objective function to penalize the number of splits used in the tree, in the spirit of complexity control in the original CART algorithm [27].

We omit these variables and the associated constraints to align the formulation with our decision tree problem (3.1) and our MIO formulation (3.2). It is straightforward to extend our formulation (3.2) to that setting by adding the binary decision variable d_s for each $s \in \textbf{splits}$, replacing constraint (3.2c) with

$$\sum_{i=1}^{d} \sum_{j=1}^{J_i} \lambda_{s,i,j} = d_s, \quad \forall\ s \in \textbf{splits},$$

and adding the constraint (3.8).

2. **Minimum number of observations required in a leaf**. The second difference is that the Bertsimas and Dunn [16] formulation includes an additional constraint that requires that the number of observations mapped to a leaf, if it is included in the topology, must be at least some number $N_{\min}$. Formally, this is modeled by adding a binary decision variable α_ℓ which is 1 if leaf ℓ has at least $N_{\min}$ observations mapped to it and 0 if there are no observations mapped to it, and then adding the following

family of constraints:

$$y_{m,\ell} \leq \alpha_\ell, \quad \forall\ \ell \in \textbf{leaves},\ m \in [n], \tag{3.9}$$

$$\sum_{m=1}^{n} y_{m,\ell} \geq N_{\min}\alpha_\ell, \quad \forall\ \ell \in \textbf{leaves} \tag{3.10}$$

As with the variables and constraints for modeling the choice of topology, we omit these variables and constraints to keep the formulation consistent with our problem.

Third, as noted above, the Bertsimas and Dunn [16] formulation, and our alternate version of this formulation (problem (3.5)) apply to binary and multiclass classification problems where all misclassifications are weighted the same. In this setting, the set of actions $\mathcal{A}$ corresponds to a set of labels or classes, and the weights $c_{m,a}$ in problem (3.1) are defined as

$$c_{m,a} = \begin{cases} 1 & \text{if } Y_a^m = a, \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

Formulation (3.5) and the original formulation of Bertsimas and Dunn [16] does not apply directly to more general choices of $\mathbf{c} = (c_{m,a})_{m\in[n],a\in\mathcal{A}}$. In our theoretical comparison of formulation (3.5), we thus focus on the case when $\mathbf{c}$ corresponds to a multiclass classification problem.

3.3.2 Theoretical Comparison of Formulations

We now theoretically compare our formulation with the formulation of Bertsimas and Dunn [16]. For our formulation, we let $\mathcal{F}_{AM}$ denote the feasible region of the LP relaxation of problem (3.2). Similarly, we let $\mathcal{F}_{BD}$ denote the feasible regions of LP relaxations of Bertsimas and Dunn [16].

A challenge that arises when trying to compare the polyhedra $\mathcal{F}_{AM}$ and $\mathcal{F}_{BD}$ is that these polyhedra are not defined in terms of a common set of decision variables; for example, while both our formulation and the Bertsimas and Dunn [16] formulation include the variables $\mathbf{y}$

(assignment of observations to leaves) and $\boldsymbol{\pi}$ (assignment of actions to leaves), the Bertsimas and Dunn [16] formulation includes the variable $\mathbf{b}$ (a real valued vector that represents the collection of chosen split points) which does not appear in our formulation, whereas our formulation includes the variable $\boldsymbol{\lambda}$ which does not appear in the Bertsimas and Dunn [16] formulation. To circumvent this difficulty, we will consider several auxiliary polyhedra that embed each solution in $\mathcal{F}_{AM}$ in a higher dimensional space that includes the variables of each of the other polyhedra that are absent in our formulation.

For the Bertsimas and Dunn [16] formulation, we define the polyhedron $\mathcal{F}'_{AM}$ as follows:

$$\mathcal{F}'_{AM} = \left\{ (\boldsymbol{\lambda}, \mathbf{y}, \mathbf{w}, \boldsymbol{\pi}, \mathbf{M}, \bar{\mathbf{N}}, \mathbf{N}, \mathbf{f}, \mathbf{b}) \;\middle|\; \begin{array}{l} (\boldsymbol{\lambda}, \mathbf{y}, \mathbf{w}, \boldsymbol{\pi}) \in \mathcal{F}_{AM}; \\ (\mathbf{M}, \bar{\mathbf{N}}) \in \mathcal{M}(\mathbf{y}, \boldsymbol{\pi}); \\ \mathbf{N} \in \mathcal{N}(\mathbf{y}); \\ \mathbf{f} \in \mathbf{F}(\boldsymbol{\lambda}); \\ \mathbf{b} \in \mathbf{B}(\boldsymbol{\lambda}) \end{array} \right\},$$

where

$$\mathcal{M}(\mathbf{y}, \boldsymbol{\pi}) = \left\{ (\mathbf{M}, \bar{\mathbf{N}}) \in \mathbb{R}_+^{|\mathbf{leaves}|} \times \mathbb{R}_+^{|\mathbf{leaves}|} \mid \right.$$
$$\left. \bar{N}_\ell = \sum_{m=1}^{n} y_{m,\ell}, M_\ell = \bar{N}_\ell - \sum_{m=1}^{n} \sum_{a \in \mathcal{A}} c_{m,a} \min\{y_{m,\ell}, \pi_{\ell,a}\}, \ell \in \mathbf{leaves} \right\},$$

$$\mathcal{N}(\mathbf{y}) = \left\{ \mathbf{N} \in \mathbb{R}_+^{|\mathcal{A}| \times |\mathbf{leaves}|} \mid N_{a,\ell} = \frac{1}{2} \sum_{m=1}^{n} (1 + \bar{Y}_{m,a}) y_{m,\ell}, a \in \mathcal{A}, \ell \in \mathbf{leaves} \right\},$$

$$\mathbf{F}(\boldsymbol{\lambda}) = \left\{ \mathbf{f} \in [0,1]^{|\mathbf{splits}| \times d} \mid f_{s,i} = \sum_{j \in J[i]} \lambda_{s,i,j}, i \in [d], s \in \mathbf{splits} \right\},$$

$$\mathbf{B}(\boldsymbol{\lambda}) = \left\{ \mathbf{b} \in \mathbb{R}_+^{|\mathbf{splits}|} \mid b_s = \min_{m \in [n]} \left\{ \sum_{i \in [d]} \sum_{j \in J[i]} \lambda_{s,i,j} X_i^m - \left(\sum_{i \in [d]} \sum_{j : j < \tau_{i,m}} \lambda_{s,i,j} \right) + 1 \right\}, s \in \mathbf{splits} \right\},$$

are the feasible sets of variables $\mathbf{M}, \bar{\mathbf{N}}, \mathbf{N}, \boldsymbol{\alpha}, \mathbf{f}$ and $\mathbf{b}$, respectively. Note that for the polyhedron $\mathcal{F}'_{AM}$, it is straightforward to see that projecting it on the original variables of our formulation recovers the feasible region of the relaxation of our formulation, that is, $\mathbf{proj}_{(\mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\pi}, \mathbf{w})}(\mathcal{F}'_{AM}) = \mathcal{F}_{AM}$.

Our first main result is that our formulation is at least as strong as formulation (3.5).

Theorem 5 $\mathbf{proj}_{(\boldsymbol{M},\bar{\boldsymbol{N}},\boldsymbol{N},\boldsymbol{f},\boldsymbol{b},\boldsymbol{\pi},\mathbf{y})}\, \mathcal{F}'_{AM} \subseteq \mathcal{F}_{BD}$.

In addition to Theorem 5 (see Appendix B.1.2 for the proof), we can further compare our formulation with formulation (3.5) in two special cases. The first special case that we will consider is the case when there is only one observation, that is, $n = 1$. The following proposition (see Section B.1.3 of the ecompanion for the proof) asserts that our formulation (3.2) is integral in this case.

Proposition 5 *When $n = 1$, $\mathcal{F}_{AM}$ is integral, that is, every extreme point $(\mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\pi}, \mathbf{w})$ of $\mathcal{F}_{AM}$ satisfies $\mathbf{y} \in \{0,1\}^{[n]\times\mathbf{leaves}}$.*

In contrast, this property does not hold for the Bertsimas and Dunn [16] formulations.

Proposition 6 *When $n = 1$, $\mathcal{F}_{BD}$ is not integral in general.*

We establish Proposition 6 by providing an example of a specific instance with a single observation where the Bertsimas and Dunn [16] formulation has a non-integral extreme point.

The second special case that we consider is when $|\mathbf{splits}| = 1$ (i.e., the decision tree consists of a single split) and the assignment of actions to leaves $\boldsymbol{\pi}$ is fixed a priori. For this special case, we define the following restricted versions of $\mathcal{F}_{AM}$ and $\mathcal{F}_{BD}$ where $\bar{\boldsymbol{\pi}}$ is a fixed assignment of actions to leaves (a binary vector satisfying $\sum_{a\in\mathcal{A}} \bar{\pi}_{\ell,a} = 1$):

$$\mathcal{F}_{AM}(\bar{\boldsymbol{\pi}}) = \{(\mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\pi}, \mathbf{w}) \in \mathcal{F}_{AM} \mid \boldsymbol{\pi} = \bar{\boldsymbol{\pi}}\}, \tag{3.12}$$

$$\mathcal{F}_{BD}(\bar{\boldsymbol{\pi}}) = \{(\mathbf{M}, \bar{\mathbf{N}}, \mathbf{N}, \mathbf{f}, \mathbf{b}, \boldsymbol{\pi}, \mathbf{y}) \in \mathcal{F}_{BD} \mid \boldsymbol{\pi} = \bar{\boldsymbol{\pi}}\}. \tag{3.13}$$

In this special case, we show that for any $\bar{\boldsymbol{\pi}}$, the restricted polyhedron $\mathcal{F}_{AM}(\bar{\boldsymbol{\pi}})$ is integral whenever there is a single split, for any number of observations.

Proposition 7 *Suppose that the tree is such that* $|\mathbf{splits}| = 1$ *and let* $\bar{\boldsymbol{\pi}}$ *be such that* $\bar{\pi}_{\ell,a} \in \{0,1\}$ *for all* ℓ *and* a*, and* $\sum_{a\in\mathcal{A}} \bar{\pi}_{\ell,a} = 1$*. Then* $\mathcal{F}_{AM}(\bar{\boldsymbol{\pi}})$ *is integral.*

The proof of Theorem 7 follows from the pairwise independent branching property of the root split (see Section B.1.5 of the ecompanion). In contrast to our formulation, this property does not apply to the Bertsimas and Dunn [16] formulation.

Proposition 8 *Suppose that the tree is such that* $|\mathbf{splits}| = 1$ *and let* $\bar{\boldsymbol{\pi}}$ *be such that* $\bar{\pi}_{\ell,a} \in \{0,1\}$ *for all* ℓ *and* a*, and* $\sum_{a\in\mathcal{A}} \bar{\pi}_{\ell,a} = 1$*. Then* $\mathcal{F}_{BD}(\bar{\boldsymbol{\pi}})$ *is not integral in general.*

3.4 Constraint Generation-Based Solution Method

In this section, we propose an exact solution method for formulation (3.2) based on constraint generation.

Recall from Section 3.2.3 that the role of continuous $\boldsymbol{\lambda}$ variables in formulation (3.2) is to enforce feasibility conditions on the $\mathbf{y}$ variables, and thus on the $\mathbf{w}$ variables, to ensure the split rules in the decision tree are satisfied. Moreover, the $\boldsymbol{\lambda}$ variables do not appear in the objective function and only become relevant when the $y_{m,\ell}$ variable for the leaf ℓ to which observation m is mapped to takes the value of 1. To demonstrate this, consider a given vector $\mathbf{y}$ such that $y_{m,\ell} = 1$ for some $m \in [n]$ and leaf $\ell \in \mathbf{leaves}$. For a split s that is not an ancestor of ℓ, constraints (3.2d) and (3.2e) will be vacuous, since they give $\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} \leq 1$ and $\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} \leq 1$, respectively, which are already implied by constraint (3.2c). Similarly, for the splits which are ancestors of leaf ℓ, if we define left ancestors as splits which direct observation m to the left and right ancestors as splits which direct observation m to the right, constraints (3.2d) are not needed for right ancestors and constraints (3.2e) are not needed for left ancestors. This suggests that as one solves problem (3.2) using branch-and-bound, one will be encumbered by many vacuous constraints.

The purpose of our constraint generation approach is to avoid adding constraints (3.2d)

and (3.2e) to the model unless they are needed. For this reason, instead of solving formulation (3.2) with constraints (3.2d) and (3.2e) for all splits $s \in \textbf{splits}$ and observations $m \in [n]$, we solve a relaxation of it by adding constraints (3.2d) and (3.2e) for a subset of splits and observations. This gives us an upper bound which may not be feasible to the original problem. We then introduce constraints (3.2d) and (3.2e) for splits and observations that violate them in the current solution and repeat this procedure until we find a solution which satisfies all the excluded constraints (3.2d) and (3.2e). By iteratively solving a relaxation of formulation (3.2) and only adding violated constraints, we circumvent including constraints which are not necessary to find the optimal solution.

The procedure we described above requires finding violated constraints for a given $\mathbf{y}$ in each step of the algorithm and adding them into the model. One way of doing this is to check whether a given solution satisfies constraints (3.2d) and (3.2e) for all $s \in \textbf{splits}$ and $m \in [n]$ and whenever a violated constraint is found, include it to the formulation and solve it again. While this is straightforward, this approach does not work efficiently in constraint generation for the following reason. Let $(\boldsymbol{\lambda}', \mathbf{y}')$ be a candidate solution to formulation (3.2) that satisfies constraints (3.2b), (3.2c) and constraints (3.2f)-(3.2l). According to the scheme we just described, assume that we check the feasibility of each split and observation and find constraint (3.2d) is violated for some $s' \in \textbf{splits}$ and $m' \in [n]$ without loss of generality. We then add constraint (3.2d) to the formulation for split s' and observation m' and solve it again. This may result in a candidate solution $(\boldsymbol{\lambda}'', \mathbf{y}'')$ such that $\mathbf{y}' = \mathbf{y}''$ and $\sum_{j \in J[i']} |\lambda'_{s,i',j} - \lambda''_{s,i',j}| = 2$ for some $i' \in [d]$. In words, this implies that new candidate solution differs from the previous one only in the split value assigned to split s in feature i' so that the assignment of observations to leaves found in the previous iteration, $\mathbf{y}'$, is still feasible. Alternatively, to avoid this issue, we can find all violated constraints for a candidate solution and solve the formulation after including all of them. Observe that, in either case, moving to another candidate solution $(\boldsymbol{\lambda}'', \mathbf{y}'')$ such that $\mathbf{y}'' \neq \mathbf{y}'$ may require adding a large number of constraints with this approach.

Algorithm 1 provides the pseudocode of a more efficient procedure. Since constraints (3.2d) and (3.2e) are written for each split, we can verify the feasibility of each split independently. For a given split s, we find the feature that has the highest value of $\sum_{j \in J[i]} \lambda_{s,i,j}$ and order the observations that are assigned to the left and to the right of that split in their values in the chosen feature. We then compare the highest value on the left (τ_{i,m_1}) and lowest value on the right (τ_{i,m_2}). If the highest value on the left is greater than or equal to the lowest value on the right ($\tau_{i,m_1} \geq \tau_{i,m_2}$), the current solution is infeasible and we need to add the violated constraints into the model for observations m_1 and m_2 for split s. The violation of the split rule can occur in two ways: Either observation m_1 violates constraint (3.2d) or observation m_2 violates constraint (3.2e) which will force it to the left of split s. Therefore, for split s, we include constraint (3.2d) for observation m_1, constraint (3.2e) for observation m_2 and solve the formulation again. We repeat this procedure until we reach a candidate solution that satisfies constraints (3.2d) and (3.2e) for all splits. The following proposition demonstrates why this constraint generation scheme is more efficient than the one discussed before.

Proposition 9 *Let* $(\boldsymbol{\lambda}, \mathbf{y})$ *be a candidate solution to formulation* (3.2) *which satisfies constraints* (3.2b), (3.2c) *and constraints* (3.2f)-(3.2l). *For a split* s*, assume Algorithm 1 returns constraint* (3.2d) *for an observation* m_1 *and constraint* (3.2e) *for an observation* m_2*. Solving the formulation with these constraints added results in a candidate solution* $(\boldsymbol{\lambda}', \mathbf{y}')$ *such that at least one of the following holds.*

(a) $\mathbf{y}' \neq \mathbf{y}$,

(b) The feature assigned to split s *is different, i.e.,* $\sum_{j \in J[i']} |\lambda_{s,i',j} - \lambda'_{s,i',j}| = 1$ *for some* $i' \in [d]$.

Proposition 9 (see Section B.1.7 in the ecompanion for proof) guarantees that adding constraints according to the procedure described in Algorithm 1 produces a new candidate

solution that differ from the previous one either in the assignment of observations to the leaves found in the previous iteration, $\mathbf{y}$, or in the feature assigned to split s. This implies that in a data set that contains large number of unique values, a constraint generation procedure based on Algorithm 1 can find the optimal solution faster than the approach discussed before, since it avoids considering solutions that only differ in split value.

Algorithm 1 Constraint generation approach

Require: Candidate solution $(\boldsymbol{\lambda}, \mathbf{y})$ satisfying constraints (3.2b), (3.2c), (3.2f)-(3.2k), a split $s \in \textbf{splits}$

Initialize $V \leftarrow \emptyset$

Set $i \leftarrow \arg\max_{i \in [d]} \sum_{j \in J[i]} \lambda_{s,i,j}$

Set $N_L \leftarrow \{m \in [n] \mid y_{m,\ell} = 1 \text{ for } \ell \in \textbf{left}(s)\}$

Set $N_R \leftarrow \{m \in [n] \mid y_{m,\ell} = 1 \text{ for } \ell \in \textbf{right}(s)\}$

Set $m_1 \leftarrow \arg\max_{m \in N_L} \tau_{i,m}$

Set $m_2 \leftarrow \arg\min_{m \in N_R} \tau_{i,m}$

if $\tau_{i,m_1} \geq \tau_{i,m_2}$ **then**

Update $V \leftarrow V \cup \{\text{Constraint (3.2d) for observation } m_1 \text{ and split } s\}$

Update $V \leftarrow V \cup \{\text{Constraint (3.2e) for observation } m_2 \text{ and split } s\}$

return V

3.5 Computational Experiments

In this section, we provide a comparison between the solution times of MIO and the constraint generation-based solution method we proposed. To test the solution methods, we use a collection of data sets chosen from the UCI machine learning repository [35]. All of our numerical experiments are implemented in the Julia technical computing language, version 1.5 [23] using the JuMP package (Julia for Mathematical Programming; see [36]), and executed on a 2017 Apple MacBook Pro with a 3.1GHz Intel i7 quad core CPU and 16GB of

memory.

In decision tree problem, the data sets with larger number of unique values in each feature is harder to solve rather than binary-valued data sets with large number of observations. Therefore, we perform our experiments on data sets which mostly consist of continuous values and have large number of unique numbers in each feature. Table 3.1 provides a summary of the data sets used in this experiment and their characteristics.

For each data set, we generate a collection of problem instances as follows. We run CART as implemented in the `rpart` [80] package in `R` [71] on the complete data set, using default parameters. We then truncate the tree topology to a depth D and solve problem 3.1 with the truncated topology, where we set the rewards $c_{m,a}$ to correspond to binary or multiclass classification (i.e., for each observation m, if the true label is a', then we set $c_{m,a} = 0$ if $a \neq a'$ and we set $c_{m,a'} = 1$). Thus, the decision tree learning problem corresponds to maximizing the number of correctly classified observations. This resulted in a collection of 75 problem instances.

For each problem instance, we test five different approaches: `BD`, which is the (modified) Bertsimas and Dunn [16] model (formulation (3.5)); `AM`, which is our basic formulation (3.2); `AM+CG`, which is our basic formulation (3.2) combined with our constraint generation procedure (Algorithm 1) in Section 3.4; `AM2`, which is our incremental encoding formulation 3.4; and `AM2+CG`, which is our incremental encoding formulation combined with our constraint generation procedure (Algorithm 1). We execute each approach with a time limit of one hour. For each approach, we measure the solution time in seconds, as well as the optimality gap of the solution returned by the formulation. The optimality gap G is defined as $G = 100\% \times (Z_{UB} - Z_{LB})/Z_{UB}$, where Z_{UB} is the tightest upper bound available at termination, while Z_{LB} is the tightest lower bound available at termination (which corresponds to an integer solution).

Tables 3.2 and 3.3 report the solution times of the five approaches across the different problem instances, while Tables 3.4 and 3.5 report the optimality gaps of the five approaches

Data set	n	d	$\lvert\mathcal{A}\rvert$	$\sum_{i=1}^{d} J_i/d$	$\max_i J_i$
Acute-inflammations-1	120	6	2	9.0	44
Acute-inflammations-2	120	6	2	9.0	44
Banknote-authentication	1372	4	2	1255.0	1338
Blood-transfusion	748	4	2	43.8	78
Breast-cancer-coimbra	116	9	2	100.0	116
Breast-cancer-original	683	9	2	9.9	10
Climate-model-crashes	540	18	2	540.0	540
Connectionist-bench-sonar	208	60	2	187.6	208
Contraceptive-method-choice	1473	12	3	6.2	34
Dermatology	358	34	6	5.6	60
Echocardiogram	62	6	2	35.2	56
Ecoli	336	7	8	51.9	82
Haberman-survival	306	3	2	30.7	49
HCV-data	589	12	5	232.0	409
Hepatitis	79	19	2	14.3	59
Indian-liver-patient	579	10	2	102.5	262
Iris	150	4	3	30.8	43
Mammographic-mass	830	12	2	8.3	72
Parkinsons	195	22	2	106.9	194
Seeds	210	7	3	182.3	207
SPECTF	267	44	2	42.9	61
Statlog-Australian-credit	690	14	2	83.4	350
Statlog-German-credit	1000	24	2	11.4	125
Statlog-heart	270	20	2	19.7	144
Thyroid-disease-ann-thyroid	3772	21	3	56.3	324
Thyroid-disease-new-thyroid	215	5	3	66.8	100
Vertebral-column-2C	310	6	2	291.2	305
Vertebral-column-3C	310	6	3	291.2	305
Wall-following-robot-2	5456	2	4	1262.0	1687
Wall-following-robot-4	5456	4	4	1503.0	1779
Wine	178	13	3	98.2	133
Yeast	1484	8	10	51.5	81

Table 3.1: Summary of data sets used in the numerical experiments.

across the problem instances.

From these tables, we obtain several insights into the behavior of the different approaches. First, our approaches compare quite favorably to BD. In those problem instances where BD terminates with an optimal solution in under one hour, our approaches tend to do so as well, but in less time. For example, for the Dermatology data set with $D = 3$, BD requires over 2200 seconds (almost 40 minutes), while our approaches require between roughly two and four minutes. In those instances when BD exhaust the time limit, the optimality gap of the resulting solution is comparable to those obtained by our approaches. In some cases, the optimality gap can be appreciably lower; for example, for Blood-transfusion with $D = 4$, the optimality gap of BD is 21.3%, whereas for our approaches it can be as low as 7.1% (AM2).

Second, comparing our incremental encoding formulation AM2 to our basic formulation AM, we can see these two approaches are comparable. On average, the basic formulation AM obtains a lower average optimality gap over the 75 problem instances than AM2 (17.6% vs. 19.9%). In some instances, AM2 solves more quickly than AM (for example, Indian-liver-patient with $D = 2$), whereas in others, AM solves more quickly (for example, SPECTF with $D = 2$).

Lastly, comparing our AM and AM2 to their counterparts AM+CG and AM2+CG that use our constraint generation procedure from Section 3.4, we can see that Algorithm 1 in general helps to reduce the solution time of the formulations.

3.6 Conclusion

In this paper, we develop a new mixed-integer optimization model for decision tree learning. The proposed model is general in the sense that it can be used for both classification and policy learning purposes. Unlike the existing models in the literature, our formulation does not involve big-M constraints, and thus is more efficient for solvers. We provide several

Data set	D	|**leaves**|	Solution time (s)				
			BD	AM	AM-CG	AM2	AM2+CG
Acute-inflammations-1	2	3	1.1	1.0	3.4	0.6	2.1
Acute-inflammations-1	3	4	0.5	0.3	2.7	0.2	2.5
Acute-inflammations-2	2	3	0.3	0.1	2.7	0.0	1.9
Banknote-authentication	2	4	3600.1	3600.8	3600.6	3600.0	2781.0
Banknote-authentication	3	7	3600.4	3600.9	3600.4	3600.0	3600.1
Banknote-authentication	4	8	3600.0	3600.8	3600.6	3600.0	3600.0
Blood-transfusion	2	3	22.1	47.8	157.2	40.0	19.0
Blood-transfusion	3	4	2455.2	215.7	1106.0	164.4	294.4
Blood-transfusion	4	6	3600.0	3600.0	3600.0	3600.0	3600.0
Breast-cancer-coimbra	2	4	814.0	138.2	514.7	239.0	411.4
Breast-cancer-coimbra	3	6	3600.0	1749.1	3600.0	3600.0	3600.0
Breast-cancer-original	2	4	571.0	89.8	320.6	84.1	116.7
Breast-cancer-original	3	6	3600.0	867.2	3600.0	1129.7	3600.0
Breast-cancer-original	4	7	3600.0	2387.0	3600.0	3600.0	3600.0
Climate-model-crashes	2	4	3600.0	3600.5	3600.9	3600.0	3600.0
Climate-model-crashes	3	6	3600.0	3600.7	3601.7	3600.0	3600.0
Climate-model-crashes	4	7	3600.1	3600.9	3600.8	3600.0	3600.0
Connectionist-bench-sonar	2	4	3600.0	3600.3	3600.9	3600.0	3600.5
Connectionist-bench-sonar	3	7	3600.0	3600.4	3600.1	3600.0	3601.2
Contraceptive-method-choice	2	3	2156.2	532.4	2231.5	300.8	180.1
Contraceptive-method-choice	3	5	3600.1	3600.6	3600.0	3600.5	3600.0
Contraceptive-method-choice	4	7	3600.0	3600.1	3600.0	3600.0	3600.1
Dermatology	2	3	207.4	41.6	39.3	40.3	33.7
Dermatology	3	4	2219.0	119.5	234.5	154.6	126.6
Dermatology	4	5	3600.0	265.0	3600.0	380.5	966.7
Echocardiogram	2	3	4.7	1.1	4.1	0.9	2.8
Ecoli	2	4	2358.1	78.4	194.1	91.9	32.6
Ecoli	3	6	3600.0	1085.1	3600.0	2394.7	1342.8
Ecoli	4	7	3600.0	3600.0	3600.0	3600.0	3600.0
Haberman-survival	2	4	247.9	90.6	73.5	22.5	55.1
Haberman-survival	3	6	3600.0	1392.7	3600.0	1586.8	1263.3
Haberman-survival	4	8	3600.0	3600.0	3600.0	3600.0	3600.0
HCV-data	2	4	3600.0	1911.0	2306.9	2434.7	3600.0
HCV-data	3	6	3600.1	3607.1	3600.0	3600.0	3600.0
Hepatitis	2	3	21.8	2.7	6.2	3.5	7.4
Indian-liver-patient	2	3	3198.8	718.9	1385.7	477.3	562.8
Indian-liver-patient	3	5	3600.1	3600.3	3600.3	3600.0	3600.0
Indian-liver-patient	4	6	3600.0	3600.1	3600.3	3600.0	3600.0
Iris	2	3	12.4	2.8	3.3	0.9	2.6

Table 3.2: Solution time results for UCI data set (part 1 of 2).

Data set	D	$\lvert\mathbf{leaves}\rvert$	Solution time (s)				
			BD	AM	AM-CG	AM2	AM2+CG
Mammographic-mass	2	3	209.8	35.9	196.3	50.4	25.1
Mammographic-mass	3	4	3600.0	134.2	944.4	109.0	722.7
Parkinsons	2	4	3600.0	824.7	2496.1	2683.1	3600.0
Parkinsons	3	5	3600.0	1669.4	3600.0	3600.0	3600.2
Seeds	2	3	26.1	14.8	8.8	16.3	11.5
Seeds	3	4	414.3	166.2	477.0	105.0	109.1
SPECTF	2	3	3600.0	337.7	2515.7	1014.3	1983.6
SPECTF	3	5	3600.0	3600.0	3600.2	3600.0	3600.0
SPECTF	4	6	3600.0	3600.0	3600.1	3600.1	3600.0
Statlog-Australian-credit	2	3	1466.2	412.1	1389.8	371.4	116.4
Statlog-Australian-credit	3	4	3600.0	2193.9	3600.2	2262.3	3600.1
Statlog-Australian-credit	4	5	3600.0	3600.1	3600.7	3600.0	3600.0
Statlog-German-credit	2	3	3600.0	1050.4	2048.7	605.7	408.3
Statlog-German-credit	3	5	3600.0	3600.0	3600.1	3600.0	3600.0
Statlog-German-credit	4	7	3600.1	3600.0	3600.1	3600.0	3600.0
Statlog-heart	2	4	3600.0	237.9	752.6	368.4	865.7
Statlog-heart	3	6	3600.0	3600.1	3600.0	3600.0	3600.0
Statlog-heart	4	7	3600.0	3600.0	3600.0	3600.0	3600.0
Thyroid-disease-ann-thyroid	2	3	3600.0	3600.1	1503.9	3600.1	1096.7
Thyroid-disease-ann-thyroid	3	4	3600.1	3600.2	3675.4	3600.0	3600.0
Thyroid-disease-ann-thyroid	4	5	3600.0	3600.5	3600.2	3600.0	3600.2
Thyroid-disease-new-thyroid	2	3	19.0	8.3	6.6	7.8	4.1
Vertebral-column-2C	2	3	151.5	173.2	156.4	82.1	55.5
Vertebral-column-2C	3	4	3600.0	3600.9	2666.9	3600.0	720.1
Vertebral-column-2C	4	5	3600.0	3600.3	3600.1	3600.0	3600.4
Vertebral-column-3C	2	3	125.9	154.3	127.7	70.8	38.2
Vertebral-column-3C	3	4	2763.8	379.2	920.5	361.2	239.2
Vertebral-column-3C	4	5	3600.0	1348.3	3600.0	3600.0	2713.6
Wall-following-robot-2	2	3	847.7	1038.4	51.1	888.4	129.8
Wall-following-robot-2	3	4	857.8	835.7	61.6	3600.0	121.9
Wine	2	4	793.4	90.6	300.9	247.7	128.2
Wine	3	5	2626.0	492.3	1783.2	1739.9	2421.4
Yeast	2	4	3600.0	3600.1	3600.1	3600.0	3600.4
Yeast	3	7	3600.0	3600.1	3600.0	3600.0	3600.1
Yeast	4	8	3600.1	3600.2	3600.0	3600.0	3600.1

Table 3.3: Solution time results for UCI data set (part 2 of 2).

Data set	D	|**leaves**|	Gap (%) BD	AM	AM-CG	AM2	AM2+CG
Acute-inflammations-1	2	3	0.0	0.0	0.0	0.0	0.0
Acute-inflammations-1	3	4	0.0	0.0	0.0	0.0	0.0
Acute-inflammations-2	2	3	0.0	0.0	0.0	0.0	0.0
Banknote-authentication	2	4	7.3	16.5	–	7.3	0.0
Banknote-authentication	3	7	6.9	35.5	–	39.2	5.8
Banknote-authentication	4	8	6.9	27.8	–	26.1	15.5
Blood-transfusion	2	3	0.0	0.0	0.0	0.0	0.0
Blood-transfusion	3	4	0.0	0.0	0.0	0.0	0.0
Blood-transfusion	4	6	21.3	12.7	17.8	7.1	11.2
Breast-cancer-coimbra	2	4	0.0	0.0	0.0	0.0	0.0
Breast-cancer-coimbra	3	6	14.7	0.0	9.6	9.7	12.1
Breast-cancer-original	2	4	0.0	0.0	0.0	0.0	0.0
Breast-cancer-original	3	6	2.3	0.0	0.7	0.0	1.0
Breast-cancer-original	4	7	2.9	0.0	2.9	2.2	2.3
Climate-model-crashes	2	4	6.3	6.9	–	8.3	6.3
Climate-model-crashes	3	6	6.1	8.3	–	8.0	7.2
Climate-model-crashes	4	7	6.7	10.2	–	8.0	8.1
Connectionist-bench-sonar	2	4	17.8	13.7	–	24.0	22.1
Connectionist-bench-sonar	3	7	15.4	18.7	–	37.0	26.0
Contraceptive-method-choice	2	3	0.0	0.0	0.0	0.0	0.0
Contraceptive-method-choice	3	5	47.0	50.3	–	43.9	39.9
Contraceptive-method-choice	4	7	50.4	49.6	–	50.0	47.9
Dermatology	2	3	0.0	0.0	0.0	0.0	0.0
Dermatology	3	4	0.0	0.0	0.0	0.0	0.0
Dermatology	4	5	15.4	0.0	1.0	0.0	0.0
Echocardiogram	2	3	0.0	0.0	0.0	0.0	0.0
Ecoli	2	4	0.0	0.0	0.0	0.0	0.0
Ecoli	3	6	17.9	0.0	5.3	0.0	0.0
Ecoli	4	7	16.4	7.3	10.4	13.7	3.0
Haberman-survival	2	4	0.0	0.0	0.0	0.0	0.0
Haberman-survival	3	6	17.6	0.0	6.2	0.0	0.0
Haberman-survival	4	8	19.9	18.1	17.4	14.4	14.6
HCV-data	2	4	5.8	0.0	0.0	0.0	1.8
HCV-data	3	6	6.6	6.3	4.8	5.8	7.8
Hepatitis	2	3	0.0	0.0	0.0	0.0	0.0
Indian-liver-patient	2	3	0.0	0.0	0.0	0.0	0.0
Indian-liver-patient	3	5	25.4	25.9	–	25.6	25.6
Indian-liver-patient	4	6	24.9	25.7	–	25.7	25.7
Iris	2	3	0.0	0.0	0.0	0.0	0.0

Table 3.4: Optimality gap results for UCI data set (part 1 of 2).

Data set	D	|**leaves**|	Gap (%)				
			BD	AM	AM-CG	AM2	AM2+CG
Mammographic-mass	2	3	0.0	0.0	0.0	0.0	0.0
Mammographic-mass	3	4	14.8	0.0	0.0	0.0	0.0
Parkinsons	2	4	7.7	0.0	0.0	0.0	8.2
Parkinsons	3	5	6.2	0.0	6.9	6.2	6.2
Seeds	2	3	0.0	0.0	0.0	0.0	0.0
Seeds	3	4	0.0	0.0	0.0	0.0	0.0
SPECTF	2	3	15.5	0.0	0.0	0.0	0.0
SPECTF	3	5	14.2	14.6	–	16.9	13.5
SPECTF	4	6	18.0	14.6	–	16.5	17.6
Statlog-Australian-credit	2	3	0.0	0.0	0.0	0.0	0.0
Statlog-Australian-credit	3	4	12.5	0.0	9.6	0.0	1.5
Statlog-Australian-credit	4	5	13.6	13.4	14.3	13.6	12.3
Statlog-German-credit	2	3	26.8	0.0	0.0	0.0	0.0
Statlog-German-credit	3	5	25.2	26.4	–	25.8	24.0
Statlog-German-credit	4	7	26.3	26.5	–	26.2	25.9
Statlog-heart	2	4	19.8	0.0	0.0	0.0	0.0
Statlog-heart	3	6	16.7	10.8	18.4	13.5	14.4
Statlog-heart	4	7	17.4	15.3	17.8	16.3	13.7
Thyroid-disease-ann-thyroid	2	3	2.1	6.9	0.0	0.9	0.0
Thyroid-disease-ann-thyroid	3	4	2.0	2.0	7.5	3.9	2.1
Thyroid-disease-ann-thyroid	4	5	1.7	3.7	7.5	5.6	7.5
Thyroid-disease-new-thyroid	2	3	0.0	0.0	0.0	0.0	0.0
Vertebral-column-2C	2	3	0.0	0.0	0.0	0.0	0.0
Vertebral-column-2C	3	4	12.4	12.5	0.0	13.5	0.0
Vertebral-column-2C	4	5	12.3	12.0	12.0	13.2	4.7
Vertebral-column-3C	2	3	0.0	0.0	0.0	0.0	0.0
Vertebral-column-3C	3	4	0.0	0.0	0.0	0.0	0.0
Vertebral-column-3C	4	5	12.6	0.0	7.3	8.4	0.0
Wall-following-robot-2	2	3	0.0	0.0	0.0	0.0	0.0
Wall-following-robot-2	3	4	0.0	0.0	0.0	6.0	0.0
Wine	2	4	0.0	0.0	0.0	0.0	0.0
Wine	3	5	0.0	0.0	0.0	0.0	0.0
Yeast	2	4	51.0	37.5	–	60.4	40.7
Yeast	3	7	57.6	65.0	–	57.9	–
Yeast	4	8	58.6	63.2	–	78.1	–

Table 3.5: Optimality gap results for UCI data set (part 2 of 2).

theoretical results on the structure of the formulation and propose relaxation schemes for binary variables that preserve optimality. Theoretically, we establish that our model is a stronger formulation than the existing formulations in the literature. We further propose a constraint generation-based large scale solution method to solve the harder instances of the problem. We evaluate the performance of our formulation and the large scale solution method in comparison to the Bertsimas and Dunn [16] formulation through numeric experiments and show that our formulation outperforms the former formulation significantly in terms of computation time. We also show that in some data sets our proposed solution method can improve the solution time.

CHAPTER 4

Conclusion

In this thesis we have studied how to develop machine learning methods based on optimization for problems in business analytics. We briefly summarize our key contributions in each chapter below.

In Chapter 2, we studied a share-of-choice product design problem which focused on finding the product, defined by its attributes, that maximizes the market share under the assumption that customers follow a logit model of choice. We characterized the complexity of the logit-based share-of-choice problem by showing that even approximating this problem is NP-Hard. We further showed that this problem is NP-Hard even in the special case with two customer segments. Although the logit-based SOCPD has a very challenging nature to solve, we showed that we can obtain an exact solution method by reformulating the problem as a mixed-integer convex program. We further proposed two solution methods for this problem. The first one is based on further reformulating the problem into a mixed-integer conic program, which has a very developed theory of numerical algorithms to solve. The second solution approach is a gradient-based constraint generation algorithm which sequentially approximates the nonlinear functions in the convex formulation using their linear approximations. The second approach transforms the problem into a mixed-integer linear program, which we can solve by using commercial solvers available and constraint generation. In addition to the solution approaches, we proposed some extensions and through numeric experiments, we show how our approaches can obtain high quality solutions to large data instances.

In Chapter 3, we focused on decision trees, which are widely used methods in machine learning and statistics. Decision trees have a wide range of application areas in Business Analytics. Despite the popularity of decision trees, the predominant approach in literature to construct a decision tree is based on heuristics. In this chapter, we proposed a mixed-integer program which can construct optimal decision trees. We provided theoretical results on the structure of the model and showed that it is stronger than precedent formulations in the literature. We further developed a solution method based on constraint generation to scale our formulation to larger problem instances. Finally, we tested our model and large scale method on real life data instances and showed that it is more efficient than the precedent exact methods.

APPENDIX A

Appendix to Chapter 2

A.1 Omitted proofs

A.1.1 Proof of Theorem 1

To prove this result, we will construct a reduction from the maximum independent set (MAX-IS) problem. In the MAX-IS problem, we are given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. An independent set $U \subseteq V$ is a set of vertices such that for any pair of vertices $v, v' \in U$, $v \neq v'$, there does not exist an edge between them, that is, $(v, v') \notin E$. The goal in the MAX-IS problem is to find an independent set whose size is maximal. The MAX-IS problem is known to be NP-Hard to approximate to within a factor $O(n^{1-\epsilon})$ for any $\epsilon > 0$ (see [46]).

In what follows we will construct an approximation-preserving reduction that maps an instance of the MAX-IS problem to an instance of the unconstrained logit-based SOCPD problem. Given a graph $G = (V, E)$, let the number of attributes $n = |V|$, the number of segments $K = n$, and let $V = \{v_1, \ldots, v_n\}$ be an enumeration of the vertices in V. Define the parameters p_L and p_U as

$$p_L = \frac{1}{100n},$$
$$p_U = 1 - \frac{1}{100}.$$

Observe that both p_L and p_U can be regarded as probabilities. Using p_L, p_U, define the

parameters q_L and q_U as the logits corresponding to these probabilities:

$$q_L = \log \frac{p_L}{1 - p_L}$$
$$q_U = \log \frac{p_U}{1 - p_U}$$

Let the utility parameters $\beta_{i,j}$ for $i \in [n]$, $j \in \{0\} \cup [n]$ be defined as follows:

$$\beta_{i,j} = \begin{cases} q_L & \text{if } j = 0, \\ q_U - q_L & \text{if } j = i, \\ q_L - q_U & \text{if } j < i \text{ and } (v_i, v_j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

Note that by construction, the highest possible value that $\sigma(u_i(\mathbf{a}))$ can attain is p_U, which occurs if $a_{i'} = 0$ and $a_i = 1$. Otherwise, for any other $\mathbf{a}$, $u_i(\mathbf{a})$ satisfies $u_i(\mathbf{a}) \leq q_L$, and so $\sigma(u_i(\mathbf{a})) \leq p_L = 1/(100n)$.

Let the weight λ_k of each segment k be set to $1/n$. Finally, let $F : \mathcal{A} \to [0, 1]$ be defined as the share of choice objective function:

$$F(\mathbf{a}) \equiv \frac{1}{n} \sum_{i=1}^{n} \sigma(u_i(\mathbf{a})).$$

To establish the result we need to verify two claims.

1. **Claim 1**. For any independent set $U \subseteq V$, there exists a product $\mathbf{a}$ such that $F(\mathbf{a}) \geq \frac{99}{100n}|U|$.

2. **Claim 2**. For any product $\mathbf{a}$ with share-of-choice given by $F(\mathbf{a})$, there exists an independent set $U \subseteq V$ such that $|U| \geq \lfloor \frac{100n}{99} F(\mathbf{a}) \rfloor$.

Proof of Claim 1. Let $U \subseteq V$ be an independent set. Consider the product vector $\mathbf{a} =$

$(a_1, \ldots, a_n)$ where $a_i = \mathbb{I}\{v_i \in U\}$. For each i such that $v_i \in U$, we have:

$$\begin{aligned} u_i(\mathbf{a}) &= \beta_{i,0} + \sum_{j=1}^{n} \beta_{i,j} a_j \\ &= q_L + \sum_{\substack{j=1: \\ (v_i, v_j) \in E}}^{i-1} (q_L - q_U) a_j + (q_U - q_L) a_i \\ &= q_L + 0 + (q_U - q_L) \cdot 1 \\ &= q_U \end{aligned}$$

where the second equality follows by how the attribute utilities $\beta_{i,j}$ are defined; the third equality follows because U is an independent set, so $a_j = 0$ for all attributes j such that there exists an edge between v_i and v_j; the fourth follows by algebra. Thus, we have:

$$\begin{aligned} F(\mathbf{a}) &= \frac{1}{n} \sum_{i=1}^{n} \frac{\exp(u_i(\mathbf{a}))}{1 + \exp(u_i(\mathbf{a}))} \\ &\geq \frac{1}{n} \cdot \sum_{i: v_i \in U} \frac{\exp(u_i(\mathbf{a}))}{1 + \exp(u_i(\mathbf{a}))} \\ &= \frac{1}{n} \cdot \sum_{i: v_i \in U} \frac{\exp(q_U)}{1 + \exp(q_U)} \\ &= \frac{1}{n} \cdot |U| \cdot p_U \\ &= \frac{99}{100n} \cdot |U|. \end{aligned}$$

Proof of Claim 2. Let $\mathbf{a}$ be an attribute vector. Let us define the set U as follows:

$$U = \{v_i \in V \mid \sigma(u_i(\mathbf{a})) \geq p_U\}.$$

In other words, we retrieve those vertices for which the corresponding segment's purchase probability is at least p_U.

We argue that this set U is an independent set. To see this, let us suppose for the sake of a contradiction that it is not. Then there exist two distinct vertices $v_i, v_{i'} \in U$ such that $(v_i, v_{i'}) \in E$. Without loss of generality, let us assume that $i < i'$. Observe that if we

calculate the logit of segment i', we have

$$\begin{aligned}
u_{i'}(\mathbf{a}) &= \beta_{i',0} + \sum_{j=1}^{n} \beta_{i',j} a_j \\
&= q_L + \sum_{\substack{j=1: \\ (v_j, v_{i'} \in E}}^{i'-1} (q_L - q_U) a_j + (q_U - q_L) a_{i'} \\
&\leq q_L + (q_L - q_U) a_i + (q_U - q_L) a_{i'} \\
&= q_L + (q_L - q_U) \cdot 1 + (q_U - q_L) \cdot 1 \\
&= q_L,
\end{aligned}$$

where the inequality follows because $q_L - q_U < 0$. This implies that

$$\frac{\exp(u_{i'}(\mathbf{a}))}{1 + \exp(u_{i'}(\mathbf{a}))} \leq p_L < p_U.$$

This, however, leads to a contradiction, because $v_{i'}$ was assumed to be in U, which would imply that the corresponding purchase probability of segment i' was higher than p_U. Therefore, it must be the case that U is an independent set.

Now, we derive the desired bound on $|U|$. We have:

$$\begin{aligned}
\left\lfloor \frac{100n}{99} F(\mathbf{a}) \right\rfloor &= \left\lfloor \frac{100n}{99} \cdot \frac{1}{n} \sum_{i=1}^{n} \sigma(u_i(\mathbf{a})) \right\rfloor \\
&= \left\lfloor \frac{100}{99} \cdot \sum_{i: v_i \in U} \sigma(u_i(\mathbf{a})) + \frac{100}{99} \cdot \sum_{i: v_i \notin U} \sigma(u_i(\mathbf{a})) \right\rfloor \\
&\leq \left\lfloor \frac{100}{99} \cdot |U| \cdot \frac{99}{100} + \frac{100}{99} \cdot (n - |U|) \frac{1}{100n} \right\rfloor \\
&\leq \left\lfloor |U| + \frac{100}{99} \cdot n \cdot \frac{1}{100n} \right\rfloor \\
&= \left\lfloor |U| + \frac{1}{99} \right\rfloor \\
&= |U|
\end{aligned}$$

where the first step follows by definition of F; the second step follows by algebra; the third step follows because the floor function is monotonic, and because by definition of the utility

parameters $\{\beta_{i,j}\}$, $\sigma(u_i(\mathbf{a})) \leq p_U = 99/100$ for all i, while for i such that $v_i \notin U$, it is the case that $\sigma(u_i(\mathbf{a})) \leq p_L = 1/(100n)$; the fourth step again follows by monotonicity of the floor function and the fact that $(n - |U|) \leq n$; the fifth step follows by algebra; and the last step follows by the fact that $|U|$ is an integer while $1/99$ is strictly less than 1. $\square$

A.1.2 Proof of Theorem 2

To show this result, we will show that the partition problem, a well-known NP-complete problem [52], can be reduced to the decision form of the logit-based SOCPD problem. The partition problem can be stated as follows:

Partition:

Inputs:

- Integer n;
- integers $c_1, \ldots, c_n$.

Question: Does there exist a set $S \subseteq [n]$ such that $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$?

The decision form of the logit-based SOCPD problem can be stated as follows:

Logit-based SOCPD problem with $K = 2$ (decision form):

Inputs:

- Integer n;
- utility parameters $\beta_{1,0}, \ldots, \beta_{1,n}, \beta_{2,0}, \ldots, \beta_{2,n}$;
- customer type probabilities $\lambda_1, \lambda_2 \geq 0$ such that $\lambda_1 + \lambda_2 = 1$;
- feasible set $\mathcal{A} \subseteq \{0,1\}^n$;
- target share-of-choice value θ.

Question: Does there exist an $\mathbf{a} \in \mathcal{A}$ such that

$$\lambda_1 \sigma(u_1(\mathbf{a})) + \lambda_2 \sigma(u_2(\mathbf{a})) \geq \theta$$

is satisfied?

Given an instance of the partition problem, we construct an instance of the decision form of the logit-based SOCPD problem such that the answer to the partition problem is yes if and only if the answer to the decision form of the logit-based SOCPD problem is yes.

Let $c_1, \ldots, c_n$ be the sizes of the n items in the partition problem. Let $T = \sum_{i=1}^{n} c_i$ be the total of all of the sizes. Note that the equality $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$ implies

$$\begin{aligned} \sum_{i \in S} c_i &= \sum_{i \notin S} c_i \\ \Rightarrow \sum_{i \in S} c_i + \sum_{i \in S} c_i &= \sum_{i \in S} c_i + \sum_{i \notin S} c_i \\ \Rightarrow 2 \sum_{i \in S} c_i &= T \\ \Rightarrow \sum_{i \in S} c_i &= T/2. \end{aligned}$$

Thus, a set S answers the partition problem in the affirmative if and only if $\sum_{i \in S} c_i = T/2$ if and only if $\sum_{i \notin S} c_i = T/2$.

Consider an instance of the decision form of the logit-based SOCPD problem defined as follows. Let $\lambda_1 = \lambda_2 = 0.5$. Let the number of attributes be the same as the number of items n, and let $\mathcal{A} = \{0,1\}^n$. Let $p_U = 0.9$ and $p_L = 0.1$, and define $q_U = \log \frac{p_U}{1-p_U}$ and $q_L = \log \frac{p_L}{1-p_L}$ as the logits corresponding to p_U and p_L respectively. Define the utility parameters as follows:

$$\begin{aligned}
\beta_{1,0} &= q_L + (1 - T/2) \cdot (q_U - q_L), \\
\beta_{1,i} &= (q_U - q_L) \cdot c_i, \quad \forall\ i \in [n], \\
\beta_{2,0} &= q_L + (T/2 + 1) \cdot (q_U - q_L), \\
\beta_{2,i} &= -(q_U - q_L) \cdot c_i, \quad \forall\ i \in [n].
\end{aligned}$$

The utility functions $u_1, u_2 : \mathcal{A} \to \mathbb{R}$ are then

$$\begin{aligned}
u_1(\mathbf{a}) &= \beta_{1,0} + \sum_{i=1}^{n} \beta_{1,i} a_i \\
&= q_L + (1 - T/2) \cdot (q_U - q_L) + \sum_{i=1}^{n} (q_U - q_L) \cdot c_i \cdot a_i \\
&= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^{n} c_i a_i - T/2 + 1 \right], \\
u_2(\mathbf{a}) &= \beta_{2,0} + \sum_{i=1}^{n} \beta_{2,i} a_i \\
&= q_L + (T/2 + 1) \cdot (q_U - q_L) + \sum_{i=1}^{n} -(q_U - q_L) \cdot c_i \cdot a_i \\
&= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^{n} -c_i a_i + T/2 + 1 \right].
\end{aligned}$$

Finally, let $\theta = p_U = 0.9$.

We now show that the answer to the partition problem is yes if and only if the answer to the logit-based SOCPD problem with $K = 2$ is yes.

Partition is yes $\Rightarrow$ *Logit-based SOCPD is yes.* To prove this direction of the equivalence, let S be the set for which $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$. As discussed earlier, this implies that $\sum_{i \in S} c_i = T/2$

and $\sum_{i \notin S} c_i = T/2$. Let the product vector $\mathbf{a} = (a_1, \ldots, a_n)$ be defined as

$$a_i = \mathbb{I}\{i \in S\}.$$

Observe now that:

$$\begin{aligned}
u_1(\mathbf{a}) &= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^{n} c_i a_i - T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot \left[\sum_{i \in S} c_i - T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot [T/2 - T/2 + 1] \\
&= q_L + (q_U - q_L) \cdot 1 \\
&= q_U, \\
u_2(\mathbf{a}) &= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^{n} -c_i a_i + T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot \left[\sum_{i \in S} -c_i + T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot [-T/2 + T/2 + 1] \\
&= q_L + (q_U - q_L) \\
&= q_U.
\end{aligned}$$

This implies that the objective value of $\mathbf{a}$ is

$$\begin{aligned}
&\lambda_1 \sigma(u_1(\mathbf{a})) + \lambda_2 \sigma(u_2(\mathbf{a})) \\
&= 0.5 \cdot \sigma(q_U) + 0.5 \cdot \sigma(q_U) \\
&= (0.5)(0.9) + (0.5)(0.9) \\
&= 0.9,
\end{aligned}$$

which implies that the answer to the decision form of the logit-based SOCPD problem is yes, as required.

Partition is no ⇒ Logit-based SOCPD is no. To prove the other direction of the equivalence, let $\mathbf{a}$ be any product attribute vector. We need to show that the objective value of $\mathbf{a}$ in the logit-based SOCPD problem is strictly less than 0.9. To see this, observe that if we define $S = \{i \in [n] \mid a_i = 1\}$, we obtain a subset of $[n]$. Since the answer to the partition problem is no, we know that $\sum_{i \in S} c_i \neq \sum_{i \notin S} c_i$. This is equivalent to $\sum_{i \in S} c_i \neq T/2$.

There are now two possible cases to consider for where $\sum_{i \in S} c_i$ is in relation to $T/2$. If $\sum_{i \in S} c_i > T/2$, then because the c_i's are integers, this means that $\sum_{i \in S} c_i \geq T/2 + 1$. This implies that the utility of segment 2 for product vector $\mathbf{a}$ can be upper bounded as follows:

$$\begin{aligned}
u_2(\mathbf{a}) &= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^{n} -c_i a_i + T/2 + 1 \right] \\
&= q_L + (q_U - q_L) \cdot \left[\sum_{i \in S} -c_i + T/2 + 1 \right] \\
&\leq q_L + (q_U - q_L) \cdot [-T/2 - 1 + T/2 + 1] \\
&= q_L + (q_U - q_L) \cdot 0 \\
&= q_L
\end{aligned}$$

which implies that the objective value of $\mathbf{a}$ is bounded from above as

$$\begin{aligned}
&\lambda_1 \sigma(u_1(\mathbf{a})) + \lambda_2 \sigma(u_2(\mathbf{a})) \\
&\leq 0.5 \cdot 1 + 0.5 \cdot \sigma(q_L) \\
&= 0.5 + (0.5)(p_L) \\
&= 0.55 \\
&< 0.9.
\end{aligned}$$

Alternatively, if $\sum_{i \in S} c_i < T/2$, then we know that $\sum_{i \in S} c_i \leq T/2 - 1$. This implies that

the utility of segment 1 for $\mathbf{a}$ can be upper bounded as follows:

$$\begin{aligned} u_1(\mathbf{a}) &= q_L + (q_U - q_L) \cdot \left[\sum_{i=1}^{n} c_i a_i - T/2 + 1 \right] \\ &= q_L + (q_U - q_L) \cdot \left[\sum_{i \in S} c_i - T/2 + 1 \right] \\ &\leq q_L + (q_U - q_L) \cdot [T/2 - 1 - T/2 + 1] \\ &= q_L, \end{aligned}$$

which again implies that the objective value of $\mathbf{a}$ is bounded from above as

$$\begin{aligned} &\lambda_1 \sigma(u_1(\mathbf{a})) + \lambda_2 \sigma(u_2(\mathbf{a})) \\ &\leq \lambda_1 \sigma(q_L) + \lambda_2 \cdot 1 \\ &= (0.5)(0.1) + (0.5)(1) \\ &= 0.55 \\ &< 0.9. \end{aligned}$$

This shows that if the answer to the partition problem is no, then the answer to our instance of the decision form of the logit-based SOCPD problem is also no.

Since our instance of the logit-based SOCPD problem can be constructed in polynomial time from the instance of the partition problem, it follows that the logit-based SOCPD problem is NP-Hard even when the number of segments K is equal to 2. □

A.1.3 Proof of Proposition 1

First, we note that this problem has a unique optimal solution since the objective function is strictly concave and the feasible region is a convex set.

To find the optimal solution, we formulate the Lagrangean of this problem:

$$\mathcal{L}(x_{k,1}, x_{k,0}, \mu) = u_k(\mathbf{a}) x_{k,1} + 0 x_{k,0} - x_{k,1} \log(x_{k,1}) - x_{k,0} \log(x_{k,0}) + \mu(x_{k,1} + x_{k,0} - 1)$$

The partial derivatives of $\mathcal{L}$ with respect to $x_{k,1}, x_{k,0}, \mu$ are

$$\frac{\partial}{\partial x_{k,1}}\mathcal{L}(x_{k,1}, x_{k,0}, \mu) = u_k(\mathbf{a}) - \log(x_{k,1}) - 1 + \mu, \tag{A.1}$$

$$\frac{\partial}{\partial x_{k,0}}\mathcal{L}(x_{k,1}, x_{k,0}, \mu) = 0 - \log(x_{k,0}) - 1 + \mu, \tag{A.2}$$

$$\frac{\partial}{\partial \mu}\mathcal{L}(x_{k,1}, x_{k,0}, \mu) = x_{k,1} + x_{k,0} - 1. \tag{A.3}$$

The first-order conditions that must be satisfied by an optimal solution of this problem are that all three partial derivatives are equal to zero. Thus, at optimality, the first order conditions for $x_{k,1}$ and $x_{k,0}$ give us that

$$x^*_{k,1} = \exp(u_k(\mathbf{a})) \cdot \exp(\mu - 1), \tag{A.4}$$

$$x^*_{k,0} = \exp(\mu - 1). \tag{A.5}$$

Using the condition that $x_{k,1} + x_{k,0} - 1 = 0$ or equivalently $x_{k,1} + x_{k,0} = 1$, we get

$$\exp(u_k(\mathbf{a})) \cdot \exp(\mu - 1) + \exp(\mu - 1) = 1$$

which implies that $\exp(\mu - 1) = 1/(\exp(u_k(\mathbf{a})) + 1)$. Substituting this into the expressions for $x^*_{k,1}$ and $x^*_{k,0}$, we obtain that

$$x^*_{k,1} = \frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a})) + 1},$$
$$x^*_{k,0} = \frac{1}{\exp(u_k(\mathbf{a})) + 1}.$$

To verify the objective value of this optimal solution, we have

$$\begin{aligned}
&u_k(\mathbf{a})x^*_{k,1} + 0x^*_{k,0} - x^*_{k,1}\log(x^*_{k,1}) - x^*_{k,0}\log(x^*_{k,0}) \\
&= u_k(\mathbf{a})\frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a})) + 1} - \frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a})) + 1} \cdot \log\left[\frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a})) + 1}\right] \\
&\quad - \frac{1}{\exp(u_k(\mathbf{a})) + 1} \cdot \log\left[\frac{1}{\exp(u_k(\mathbf{a})) + 1}\right] \\
&= u_k(\mathbf{a})\frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a})) + 1} - \frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a})) + 1} \cdot u_k(\mathbf{a}) + \frac{\exp(u_k(\mathbf{a}))}{\exp(u_k(\mathbf{a})) + 1} \cdot \log(1 + \exp(u_k(\mathbf{a}))) \\
&\quad + \frac{1}{1 + \exp(u_k(\mathbf{a}))} \cdot \log(1 + \exp(u_k(\mathbf{a}))) \\
&= \log(1 + \exp(u_k(\mathbf{a}))),
\end{aligned}$$

which completes the proof. □

A.1.4 Proof of Theorem 3

To show this equivalence, we will show that for every $\mathbf{a} \in \mathcal{A}$, the solution $(\mathbf{a}, \mathbf{u}, \mathbf{x})$ where $\mathbf{u}$ and $\mathbf{x}$ are given by

$$u_k = u_k(\mathbf{a}), \quad \forall\ k, \tag{A.6}$$

$$x_{k,1} = \frac{\exp(u_k(\mathbf{a}))}{1 + \exp(u_k(\mathbf{a}))}, \quad \forall\ k \tag{A.7}$$

$$x_{k,0} = \frac{1}{1 + \exp(u_k(\mathbf{a}))}, \quad \forall\ k \tag{A.8}$$

is the only feasible solution of problem (2.5) corresponding to $\mathbf{a}$; stated differently, there is a one-to-one correspondence between product vectors $\mathbf{a}$ and feasible solutions of problem (2.5), and by finding a solution $(\mathbf{a}, \mathbf{u}, \mathbf{x})$ of problem (2.5) that maximizes $\sum_{k=1}^{K} \lambda_k \cdot x_{k,1}$ guarantees that $\mathbf{a}$ maximizes $\sum_{k=1}^{K} \lambda_k \cdot \exp(u_k(\mathbf{a}))/(1 + \exp(u_k(\mathbf{a})))$.

Observe that by construction, the solution $(\mathbf{a}, \mathbf{u}, \mathbf{x})$ automatically satisfies constraints (2.5d), (2.5g) and (2.5b). We thus only need to show that it satisfies constraint (2.5c). Note that the left-hand side of constraint (2.5c) is the objective function of the representative agent model (2.4), where the utility of the product is $u_k(\mathbf{a})$. By Proposition 1, the optimal objective value of this representative agent model is $\log(1 + \exp(u_k(\mathbf{a})))$, which is exactly the right-hand side of constraint (2.5c). Thus, constraint (2.5c) requires that $x_{k,1}$ and $x_{k,0}$ must be chosen to have an objective value at least as great as the optimal objective value; in other words, $(x_{k,1}, x_{k,0})$ is enforced to be an optimal solution of problem (2.4). By Proposition 1, we know that the choice of $x_{k,1}$ and $x_{k,0}$ as in (A.7) and (A.8) is an optimal solution of this representative agent problem, and therefore $x_{k,1}$, $x_{k,0}$ and u_k set as in (A.6) - (A.8) satisfy constraint (2.5c). Moreover, since this solution of the representative agent problem is unique, there can be no other choice of $x_{k,1}$ and $x_{k,0}$ that will satisfy this constraint. This establishes that given $\mathbf{a}$, the solution $(\mathbf{a}, \mathbf{u}, \mathbf{x})$ with $\mathbf{u}$ and $\mathbf{x}$ as defined in (A.6) - (A.8) is the only feasible solution corresponding to $\mathbf{a}$, and thus that the two problems are equivalent. □

A.1.5 Proof of Proposition 2

Let $\mathbf{a}$ be the optimal solution of problem (2.29). We have that

$$\begin{aligned} Z^*_{GM} &= \prod_{k=1}^{K} \left[\frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))} \right]^{\lambda_k} \\ &\leq \sum_{k=1}^{K} \lambda_k \cdot \frac{\exp(u_k(\mathbf{a}))}{1+\exp(u_k(\mathbf{a}))} \\ &\leq Z^*_{AM}, \end{aligned}$$

where the first step follows by the arithmetic-geometric mean inequality and the second step by the definition of Z^*_{AM} as the optimal objective value of (2.1). $\square$

A.1.6 Proof of Theorem 4

Let $\mathbf{x}^* = \mathbf{x}(\mathbf{a}^*)$ and $\hat{\mathbf{x}} = \mathbf{x}(\hat{\mathbf{a}})$. To prove the result we proceed in three steps.

Step 1: The first step in our proof is to show that if there exist nonnegative constants $\overline{\alpha}$ and $\underline{\alpha}$ such that g satisfies

$$\underline{\alpha} f(\mathbf{x}) \leq g(\mathbf{x}) \leq \overline{\alpha} f(\mathbf{x}) \tag{A.9}$$

for all $\mathbf{x} \in \mathcal{X}$, then $\hat{\mathbf{x}}$ satisfies

$$f(\hat{\mathbf{x}}) \geq (\underline{\alpha}/\overline{\alpha}) \cdot f(\mathbf{x}^*). \tag{A.10}$$

To establish this, we will first bound the quantity $f(\mathbf{x}^*) - f(\hat{\mathbf{x}})$. We have

$$\begin{aligned} f(\mathbf{x}^*) - f(\hat{\mathbf{x}}) &= [f(\mathbf{x}^*) - g(\mathbf{x}^*)] + [g(\mathbf{x}^*) - g(\hat{\mathbf{x}})] + [g(\hat{\mathbf{x}}) - f(\hat{\mathbf{x}})] \\ &\leq f(\mathbf{x}^*) - g(\mathbf{x}^*) + g(\hat{\mathbf{x}}) - f(\hat{\mathbf{x}}) \\ &\leq f(\mathbf{x}^*) - \underline{\alpha} f(\mathbf{x}^*) + \overline{\alpha} f(\hat{\mathbf{x}}) - f(\hat{\mathbf{x}}) \\ &= (1 - \underline{\alpha}) f(\mathbf{x}^*) - (1 - \overline{\alpha}) f(\hat{\mathbf{x}}) \\ &= (1 - \overline{\alpha} + \overline{\alpha} - \underline{\alpha}) f(\mathbf{x}^*) - (1 - \overline{\alpha}) f(\hat{\mathbf{x}}) \\ &= (1 - \overline{\alpha})(f(\mathbf{x}^*) - f(\hat{\mathbf{x}})) + (\overline{\alpha} - \underline{\alpha}) f(\mathbf{x}^*) \end{aligned}$$

where the first step follows by algebra; the second step follows since $g(\mathbf{x}^*) \leq g(\hat{\mathbf{x}})$, which is true by the definition of $\hat{\mathbf{x}}$ as the vector of choice probabilities for an optimal product $\hat{\mathbf{a}}$ for the function $g(\mathbf{x}(\mathbf{a}))$; the third step follows by (A.9); and the remaining steps by algebra.

Observe that by re-arranging the inequality

$$f(\mathbf{x}^*) - f(\hat{\mathbf{x}}) \leq (1 - \overline{\alpha})(f(\mathbf{x}^*) - f(\hat{\mathbf{x}})) + (\overline{\alpha} - \underline{\alpha}) f(\mathbf{x}^*) \tag{A.11}$$

we obtain that

$$\overline{\alpha}[f(\mathbf{x}^*) - f(\hat{\mathbf{x}})] \leq (\overline{\alpha} - \underline{\alpha}) f(\mathbf{x}^*). \tag{A.12}$$

Since $\overline{\alpha}$ is nonnegative, dividing through by $\overline{\alpha}$ we obtain

$$f(\mathbf{x}^*) - f(\hat{\mathbf{x}}) \leq \frac{(\overline{\alpha} - \underline{\alpha})}{\overline{\alpha}} f(\mathbf{x}^*), \tag{A.13}$$

and re-arranging, we obtain

$$\begin{aligned} f(\hat{\mathbf{x}}) &\geq \left[1 - \frac{\overline{\alpha} - \underline{\alpha}}{\overline{\alpha}}\right] f(\mathbf{x}^*) \\ &= (\underline{\alpha}/\overline{\alpha}) \cdot f(\mathbf{x}^*), \end{aligned}$$

which is the desired result.

Step 2: We now establish explicit values for the constants $\overline{\alpha}$ and $\underline{\alpha}$. Recall that by the arithmetic-geometric mean inequality, $g(\mathbf{x}) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$. Therefore, a valid choice of $\overline{\alpha}$ is 1.

For $\underline{\alpha}$, we proceed as follows. Consider the ratio $f(\mathbf{x})/g(\mathbf{x})$. For any $\mathbf{x}$, we have

$$\begin{aligned}
\frac{f(\mathbf{x})}{g(\mathbf{x})} &= \frac{\sum_{k=1}^{K} \lambda_k x_k}{\prod_{k=1}^{K} x_k^{\lambda_k}} \\
&= \sum_{k=1}^{K} \lambda_k \cdot x_k^{1-\lambda_k} \cdot \prod_{k' \neq k} x_{k'}^{-\lambda_{k'}} \\
&\leq \sum_{k=1}^{K} \lambda_k \cdot U^{1-\lambda_k} \cdot \prod_{k' \neq k} L^{-\lambda_{k'}} \\
&= \sum_{k=1}^{K} \lambda_k \cdot U^{1-\lambda_k} \cdot L^{-\sum_{k' \neq k} \lambda_{k'}} \\
&= \sum_{k=1}^{K} \lambda_k \cdot U^{1-\lambda_k} \cdot L^{\lambda_k - 1} \\
&= \sum_{k=1}^{K} \lambda_k \left(\frac{U}{L}\right)^{1-\lambda_k},
\end{aligned}$$

where the first step follows by the definitions of f and g; the second by algebra; the third by the fact that the function $h(x) = x^{1-\lambda_k}$ is increasing in x (since $1 - \lambda_k \geq 0$), and that the function $\bar{h}(x) = x^{-\lambda_{k'}}$ is decreasing in x (since $-\lambda_k \leq 0$); the fourth by algebra; the fifth by recognizing that $\sum_{k'=1}^{K} \lambda_k = 1$, which implies that $\lambda_k - 1 = -\sum_{k' \neq k} \lambda_{k'}$; and the last by algebra. This implies that a valid choice of $\underline{\alpha}$ is

$$\underline{\alpha} = \frac{1}{\sum_{k=1}^{K} \lambda_k \left(\frac{U}{L}\right)^{1-\lambda_k}}. \tag{A.14}$$

Step 3: We conclude the proof by combining Steps 1 and 2. In particular, by using $\overline{\alpha} = 1$ and $\underline{\alpha} = [\sum_{k=1}^{K} \lambda_k \left(U/L\right)^{1-\lambda_k}]^{-1}$, we obtain that

$$f(\mathbf{x}(\hat{\mathbf{a}})) \geq \frac{1}{\sum_{k=1}^{K} \lambda_k \left(\frac{U}{L}\right)^{1-\lambda_k}} \cdot f(\mathbf{x}(\mathbf{a}^*)),$$

as required. □

A.2 Additional details for numerical experiments

A.2.1 Attributes for real data instances in Section 2.5.2

Tables A.1, A.2, A.3 and A.4 display the attributes and attribute levels for the `bank`, `candidate`, `immigrant` and `timbuk2` datasets, respectively.

Attribute	Levels
Interest Rate	High Fixed Rate, Medium Fixed Rate, Low Fixed Rate, Medium Variable Rate
Rewards	1, 2, 3, 4
Annual Fee	High, Medium, Low
Bank	Bank A, Bank B, Out of State Bank
Rebate	Low, Medium, High
Credit Line	Low, High
Grace Period	Short, Long

Table A.1: Attributes for `bank` dataset.

Attribute	Levels
Age	36, 45, 52, 60, 68, 75
Military Service	Did Not serve, Served
Religion	None, Jewish, Catholic, Mainline Protestant, Evangelical Protestant, Mormon
College	No BA, Baptist College, Community College, State University, Small College, Ivy League University
Income	32K, 54K, 65K, 92K, 210K, 5.1M
Profession	Business Owner, Lawyer, Doctor, High School Teacher, Farmer, Car Dealer
Race/Ethnicity	White, Native American, Black, Hispanic, Caucasian, Asian American
Gender	Male, Female

Table A.2: Attributes for `candidate` dataset.

Attribute	Levels
Education	No Formal, 4th Grade, 8th Grade, High School, Two-Year College, College Degree, Graduate Degree
Gender	Female, Male
Origin	Germany, France, Mexico, Philippines, Poland, India, China, Sudan, Somalia, Iraq
Application Reason	Reunite With Family, Seek Better Job, Escape Persecution
Profession	Janitor, Waiter, Child Care Provider, Gardener, Financial Analyst, Construction Worker, Teacher, Computer Programmer, Nurse, Research Scientist, Doctor
Job Experience	None, 1-2 Years, 3-5 Years, 5+ Years
Job Plans	Contract With Employer, Interviews With Employer, Will Look For Work, No Plans To Look For Work
Prior Trips to US	Never, Once As Tourist, Many Times As Tourist, Six Months With Family, Once Without Authorization
Language	Fluent English, Broken English, Tried English But Unable, Used Interpreter

Table A.3: Attributes for `immigrant` dataset.

Attribute	Levels
Price	$70, $75, $80, $85, $90, $95, $100
Size	Normal, Large
Color	Black, Red
Logo	No, Yes
Handle	No, Yes
PDA Holder	No, Yes
Cellphone Holder	No, Yes
Velcro Flap	No, Yes
Protective Boot	No, Yes

Table A.4: Attributes for `timbuk2` dataset.

A.2.2 Hierarchical Bayesian model specification

For our hierarchical Bayesian model, we assume that each respondent's partworth vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ is drawn as

$$\boldsymbol{\beta} \sim N(\bar{\boldsymbol{\beta}}, \mathbf{V}_{\boldsymbol{\beta}}),$$

where $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a multivariate normal distribution with mean μ and covariance matrix $\boldsymbol{\Sigma}$. The distributions of the mean $\bar{\boldsymbol{\beta}}$ and covariance matrix $\mathbf{V}_{\boldsymbol{\beta}}$ are then specified as

$$\bar{\boldsymbol{\beta}} \sim N(\mathbf{0}, \alpha \mathbf{V}_{\boldsymbol{\beta}}),$$
$$\mathbf{V}_{\boldsymbol{\beta}} \sim IW(\nu, \mathbf{V}),$$

where $IW(\nu, \mathbf{W})$ denotes an inverse Wishart distribution with degrees of freedom ν and scale matrix $\mathbf{W}$. This model specification is implemented in `bayesm`, using the `rhierBinLogit` function. We use `bayesm`'s defaults for ν, $\mathbf{V}$ and α.

A.2.3 Additional constraints for `immigrant` dataset

As discussed in Section 2.5.2, we define $\mathcal{A}$ with some additional constraints, which we describe here:

- If the immigrant's profession attribute is set to "doctor", "research scientist", "computer programmer" or "financial analyst", then the immigrant's education attribute is set to "college degree" or "graduate degree".
- If the immigrant's profession attribute is set to "teacher" or "nurse", then the immigrant's education attribute is set to "high school", "two-year college", "college degree" or "graduate degree".
- If the immigrant's application reason attribute is set to "escape persecution", then the country of origin attribute is set to Sudan, Somalia or Iraq.

- Either the immigrant's application reason attribute is set to "seek better job" or the immigrant's job plan attribute is set to "no plans to work", but they cannot both be set in this way.

A.2.4 Competitive offerings for Section 2.5.2

Tables A.5, A.6, A.7 and A.8 display the attributes of the competitive offerings for the `bank`, `candidate`, `immigrant` and `timbuk2` datasets, respectively. We note that for `timbuk2`, we follow the same competitive offerings used in other optimization work that has used this dataset [13, 20, 21].

Attribute	Outside Option 1	Outside Option 2	Outside Option 3
Interest Rate: High fixed rate			
Interest Rate: Medium fixed rate			
Interest Rate: Low fixed rate			
Interest Rate: Medium variable rate			
Rewards: 1			
Rewards: 2			
Rewards: 3			
Rewards: 4			
Annual Fee: High			
Annual Fee: Medium			
Annual Fee: Low			
Bank: Bank A			
Bank: Bank B			
Bank: Out of state bank			
Rebate: Low			
Rebate: Medium			
Rebate: High			
Credit Line: Low			
Credit Line: High			
Grace Period: Short			
Grace Period: Long			

Table A.5: Outside options for `bank` dataset problem instances.

Attribute	Outside Option 1	Outside Option 2	Outside Option 3
Age: 36			
Age: 45			
Age: 52			
Age: 60			
Age: 68			
Age: 75			
Military Service: Did not serve			
Military Service: Served			
Religion: None			
Religion: Jewish			
Religion: Catholic			
Religion: Mainline protestant			
Religion: Evangelical protestant			
Religion Mormon			
College: No BA			
College: Baptist college			
College: Community college			
College: State university			
College: Small college			
College: Ivy League university			
Income: 32K			
Income: 54K			
Income: 65K			
Income: 92K			
Income: 210K			
Income 5.1M			
Profession: Business owner			
Profession: Lawyer			
Profession: Doctor			
Profession: High school teacher			
Profession: Farmer			
Profession: Car dealer			
Race/Ethnicity: White			
Race/Ethnicity: Native American			
Race/Ethnicity: Black			
Race/Ethnicity: Hispanic			
Race/Ethnicity: Caucasian			
Race/Ethnicity: Asian American			
Gender: Male			
Gender: Female			

Table A.6: Outside options for `candidate` dataset problem instances.

Attribute	Outside Option 1	Outside Option 2	Outside Option 3
Education: No formal			
Education: 4th grade			
Education: 8th grade			
Education: High school			
Education: Two-year college			
Education: College degree			
Education: Graduate degree			
Gender: Female			
Gender: Male			
Origin: Germany			
Origin: France			
Origin: Mexico			
Origin: Philippines			
Origin: Poland			
Origin: India			
Origin: China			
Origin: Sudan			
Origin: Somalia			
Origin: Iraq			
Application Reason: Reunite with family			
Application Reason: Seek better job			
Application Reason: Escape persecution			
Profession: Janitor			
Profession: Waiter			
Profession: Child care provider			
Profession: Gardener			
Profession: Financial analyst			
Profession: Construction worker			
Profession: Teacher			
Profession: Computer programmer			
Profession: Nurse			
Profession: Research scientist			
Profession: Doctor			
Job Experience: None			
Job Experience: 1-2 years			
Job Experience: 3-5 years			
Job Experience: 5+ years			
Job Plans: Contract with employer			
Job Plans: Interviews with employer			
Job Plans: Will look for work			
Job Plans: No plans to look for work			
Prior Trips to U.S.: Never			
Prior Trips to U.S.: Once as tourist			
Prior Trips to U.S.: Many times as tourist			
Prior Trips to U.S.: Six months with family			
Prior Trips to U.S.: Once without authorization			
Language: Fluent English			
Language: Broken English			
Language: Tried English but unable			
Language: Used interpreter			

Table A.7: Outside options for `immigrant` dataset problem instances.

Attribute	Outside Option 1	Outside Option 2	Outside Option 3
Price: $70			
Price: $75			
Price: $80			
Price: $85			
Price: $90			
Price: $95			
Price: $100			
Size: Large			
Color: Red			
Logo: Yes			
Handle: Yes			
PDA Holder: Yes			
Cellphone Holder: Yes			
Mesh Pocket: Yes			
Velcro Flap: Yes			
Protective Boot: Yes			

Table A.8: Outside options for `timbuk2` dataset problem instances. (For ease of comparison, only one level of each binary attribute is shown.)

APPENDIX B

Appendix to Chapter 3

B.1 Omitted Proofs

B.1.1 Proof of Proposition 3

We rewrite the problem as follows:

$$\begin{aligned} &\text{maximize} && \sum_{\ell \in \textbf{leaves}} G_\ell(\mathbf{y}, \boldsymbol{\lambda}) && \text{(B.1)} \\ &\text{subject to} && \text{constraints (3.2b)} - \text{(3.2e)}, \text{(3.2i)}, \text{(3.2l)}, \end{aligned}$$

where $G_\ell(\mathbf{y}, \boldsymbol{\lambda})$ is defined as the optimal value of the following subproblem:

$$\begin{aligned} G_\ell(\mathbf{y}, \boldsymbol{\lambda}) = \text{maximize} \quad & \sum_{m=1}^{M} \sum_{a \in \mathcal{A}} c_{m,a} w_{m,\ell,a} && && \text{(B.2a)} \\ \text{subject to} \quad & \sum_{a \in \mathcal{A}} \pi_{\ell,a} = 1, && && \text{(B.2b)} \\ & y_{m,\ell} = \sum_{a \in \mathcal{A}} w_{m,\ell,a}, && \forall m \in [n], && \text{(B.2c)} \\ & w_{m,\ell,a} \leq \pi_{\ell,a}, && \forall m \in [n], \quad a \in \mathcal{A}, && \text{(B.2d)} \\ & \pi_{\ell,a} \geq 0, && \forall a \in \mathcal{A}, && \text{(B.2e)} \\ & w_{m,\ell,a} \geq 0, && \forall m \in [n], \quad a \in \mathcal{A}. && \text{(B.2f)} \end{aligned}$$

We will show that there exists an optimal solution where $w_{m,\ell,a}$ and $\pi_{\ell,a}$ variables have binary values. Observe that, by constraints (B.2c) and (B.2d), we have $w_{m,\ell,a} \leq \min\{\pi_{\ell,a}, y_{m,\ell}\}$

for all $m \in [n]$ and $a \in \mathcal{A}$. Also, by constraint (B.2b), $\pi_{\ell,a} \leq 1$ for all $a \in \mathcal{A}$. Thus, if $y_{m,\ell} = 0$, we have $w_{m,\ell,a} = 0$ and if $y_{m,\ell} = 1$, we have $w_{m,\ell,a} \leq \pi_{\ell,a}$ for all $a \in \mathcal{A}$. Observe that we can construct an optimal solution in the following way:

$$w_{m,\ell,a} = \begin{cases} \pi_{\ell,a} & \text{if } y_{m,\ell} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Note that, this is feasible because of constraint (B.2b). Moreover, assume $w_{m',\ell,a'} < \pi_{\ell,a'}$ for some $m' \in [n]$ and $a' \in \mathcal{A}$ where $y_{m',\ell} = 1$. We cannot increase the objective function value by increasing any other $w_{m,\ell,a}$ as they are bounded by $\pi_{\ell,a}$ and we have $\sum_{a\in\mathcal{A}} \pi_{\ell,a} = 1$ by constraint (B.2b). Therefore, we can only obtain the maximum objective function value by having $w_{m',\ell,a'} = \pi_{\ell,a'}$. Substituting this in the objective function, we obtain

$$\begin{aligned} G_\ell(\mathbf{y}, \boldsymbol{\lambda}) &= \sum_{m=1}^{M} \sum_{a\in\mathcal{A}} c_{m,a} w_{m,\ell,a} \\ &= \sum_{\substack{m=1: \\ y_{m,\ell}=1}}^{M} \sum_{a\in\mathcal{A}} c_{m,a} \pi_{\ell,a} \\ &= \sum_{a\in\mathcal{A}} \sum_{\substack{m=1: \\ y_{m,\ell}=1}}^{M} c_{m,a} \pi_{\ell,a} \\ &= \sum_{a\in\mathcal{A}} \pi_{\ell,a} \sum_{\substack{m=1: \\ y_{m,\ell}=1}}^{M} c_{m,a}. \end{aligned}$$

Let

$$a' = \arg\max_{a} \sum_{\substack{m=1: \\ y_{m,\ell}=1}}^{M} c_{m,a}$$

and we define π^* such that

$$\pi_{\ell,a} = \begin{cases} 1 & \text{if } a = a', \\ 0 & \text{otherwise.} \end{cases}$$

Note that, π^* is a feasible solution to the subproblem. Since $\sum_{a\in\mathcal{A}} \pi_{\ell,a} = 1$, we also have

$$\sum_{a\in\mathcal{A}} \pi^*_{\ell,a} \sum_{\substack{m=1:\\ y_{m,\ell}=1}}^{M} c_{m,a} = \sum_{\substack{m=1:\\ y_{m,\ell}=1}}^{M} c_{m,a'} \geq \sum_{a\in\mathcal{A}} \pi_{\ell,a} \sum_{\substack{m=1:\\ y_{m,\ell}=1}}^{M} c_{m,a}.$$

Thus, π^* is an optimal solution for the problem, which shows that there exists an optimal solution to the subproblem where w and π are integers.

Next, we show that, when $\mathbf{y}$ is binary, decision variables $\boldsymbol{\lambda}$ also take integer values by showing that the set of feasible $\boldsymbol{\lambda}$ is a polyhedron defined by a totally unimodular contraint matrix.

Let $\mathbf{y} \in \{0,1\}$. Observe that, the set of constraints on $\boldsymbol{\lambda}$ variables, constraints (3.2c)-(3.2e), can be decomposed for each split variable. Thus, for each split variable $s \in \textbf{splits}$ we have the following set of constraints on $\boldsymbol{\lambda}$,

$$\sum_{i=1}^{d} \sum_{j=1}^{J_i} \lambda_{s,i,j} \leq 1, \tag{B.3a}$$

$$-\sum_{i=1}^{d} \sum_{j=1}^{J_i} \lambda_{s,i,j} \leq -1, \tag{B.3b}$$

$$-\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} \leq -\sum_{\ell\in\textbf{left}(s)} y_{m,\ell}, \quad \forall m \in [n], \tag{B.3c}$$

$$\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} \leq 1 - \sum_{\ell\in\textbf{right}(s)} y_{m,\ell}, \quad \forall m \in [n],, \tag{B.3d}$$

$$\lambda_{s,i,j} \geq 0, \quad \forall i \in [d], j \in J[i]. \tag{B.3e}$$

where constraint (3.2c) is expressed by constraints (B.3a) and (B.3b) in the inequality form and constraint (3.2d) is rearranged by subtracting 1 from both sides and substituting $1 - \sum_{i=1}^{d} \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} = \sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j}$ by constraint (3.2c).

We show that, the matrix formed by these constraint is totally unimodular. We use Λ^i_s to denote the vector of $\{\lambda_{s,i,j}\}_{j\in\{0\}\cup[J_i]}$ for $i \in [d]$ and $s \in \textbf{splits}$. Let $\boldsymbol{\lambda}_s$ be the vector of $\{\Lambda^i_s\}_{i\in[d]}$ for $s \in \textbf{splits}$. Then, we can rewrite constraints (B.3) in matrix form in the following way:

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \dots & \mathbf{1} \\ \mathbf{-1} & \mathbf{-1} & \dots & \mathbf{-1} \\ \mathcal{C}^1 & \mathcal{C}^2 & \dots & \mathcal{C}^d \\ \mathcal{D}^1 & \mathcal{D}^2 & \dots & \mathcal{D}^d \end{pmatrix} \left(\boldsymbol{\lambda}_s\right) \leq \begin{pmatrix} R_a \\ R_b \\ R_c \\ R_d \end{pmatrix}, \tag{B.4}$$

$$\boldsymbol{\lambda}_s \geq 0. \tag{B.5}$$

where $\mathcal{C}^i, \mathcal{D}^i \in \mathbb{R}^{m\times(J_i+1)}$ for all $i \in [d]$ correspond to the coefficient matrices of the left-hand sides of constrains (B.3c) and (B.3d). These are defined as follows. Let $\mathcal{C}^i_{m,j}$ denote the element in m^{th} row and j^{th} column in matrix $\mathcal{C}^i$ for each $i \in [d]$. Then, we have $\mathcal{C}^i_{m,j} = -\mathbb{I}_{\{j\geq\tau_{i,m}\}}$, where $\mathbb{I}_{\{.\}}$ is the indicator function. Similarly, let $\mathcal{D}^i_{m,j}$ be the element in m^{th} row and j^{th} column in matrix $\mathcal{D}^i$ for each $i \in [d]$, where $\mathcal{D}^i_{m,j} = \mathbb{I}_{\{j\geq\tau_{i,m}\}}$.

Let $\mathbf{T}$ denote the coefficient matrix formed by constraints (B.3), that is,

$$\mathbf{T} = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \dots & \mathbf{1} \\ \mathbf{-1} & \mathbf{-1} & \dots & \mathbf{-1} \\ \mathcal{C}^1 & \mathcal{C}^2 & \dots & \mathcal{C}^d \\ \mathcal{D}^1 & \mathcal{D}^2 & \dots & \mathcal{D}^d \end{pmatrix}.$$

We apply the following row operations to show $\mathbf{T}$ is totally unimodular. We add the first row to the second row and make all elements in the second row zero. Observe that, we have $\mathcal{C}^i = -\mathcal{D}^i$ for all $i \in [d]$. By adding the rows of $\mathcal{D}^i$ to $\mathcal{C}^i$, we can transform each $\mathcal{C}^i$ into zero matrices. Let $\mathbf{T}'$ be the new matrix we obtained after performing elementary row operations on matrix $\mathbf{T}$. For notational convenience, we also interchange the rows so that $\mathbf{T}'$ is given by

$$\mathbf{T}' = \begin{pmatrix} \mathcal{D}^1 & \mathcal{D}^2 & \dots & \mathcal{D}^d \\ \mathbf{1} & \mathbf{1} & \dots & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{pmatrix}.$$

Now, we use the following theorem (see [22]) to show that $\mathbf{T}'$ is totally unimodular.

Theorem 6 ***(Theorem 3.2 from [22])****. A matrix* $\boldsymbol{A}$ *is totally unimodular if and only if each collection* $\boldsymbol{J}$ *of columns of* $\boldsymbol{A}$ *can be partitioned into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries 0, +1, and -1.*

To represent the collection of columns, we will assume that we are given a set of columns which correspond to a subset of columns of matrices $\mathcal{D}^i$ for each $i \in [d]$:

$$S_i \subseteq \{0, \dots, J_i\}, \quad \forall i \in [d].$$

Now, we will show that given S_i for all $i \in [d]$, there exists two disjoint sets Q^+ and Q^- such that

$$\left| \sum_{k \in Q^+} \mathbf{T}'_{h,k} - \sum_{k \in Q^-} \mathbf{T}'_{h,k} \right| \leq 1, \quad \text{for } h = 1, \dots, m+1.$$

Now, we describe how to construct sets Q^+ and Q^-. Without loss of generality, we assume that $\{0\} \notin S_i$ for all $i \in [d]$ (otherwise, we can include column 0 to either one of Q^+ or Q^-, since column 0 is a zero vector). Let $\sigma(k)$ be the index $j \in \{1, \dots, J_i\}$ that column k corresponds to in matrix $\mathcal{D}^i$ for $k = 1, \dots, |S_i|$ for all $i \in [d]$. Moreover, let $k^i_m = \min\{k \in S_i \mid \sigma(k) \geq \tau_{i,m}\}$ for $m \in [n]$. Starting from $i = 1$, we implement the following steps.

Step 1. For each $k \in S_i$, if k is odd, put k^{th} column of matrix $\mathbf{T}'$ in Q^+. Otherwise, put k^{th} column in Q^-. Let $S^+_i \subseteq \cup^i_{p=1} S_p$ contain columns of $\cup^i_{p=1} S_p$ that we put in Q^+ and $S^-_i \subseteq \cup^i_{p=1} S_p$ contain the columns that we put in Q^-. Then, for $h = 1, \dots, m+1$, we have the following cases.

- Let $|S_i|$ be odd.

$$\sum_{k\in S_i^+} \mathbf{T}'_{h,k} - \sum_{k\in S_i^-} \mathbf{T}'_{h,k} = 1, \quad \forall h \in \{m \in [n] \mid k_m^1 \text{ is odd}\},$$
$$\sum_{k\in S_i^+} \mathbf{T}'_{h,k} - \sum_{k\in S_i^-} \mathbf{T}'_{h,k} = 0, \quad \forall h \in \{m \in [n] \mid k_m^1 \text{ is even}\},$$
$$\sum_{k\in S_i^+} \mathbf{T}'_{m+1,k} - \sum_{k\in S_i^-} \mathbf{T}'_{m+1,k} = 1.$$

- Let $|S_i|$ be even.

$$\sum_{k\in S_i^+} \mathbf{T}'_{h,k} - \sum_{k\in S_i^-} \mathbf{T}'_{h,k} = 0, \quad \forall h \in \{m \in [n] \mid k_m^1 \text{ is odd}\},$$
$$\sum_{k\in S_i^+} \mathbf{T}'_{h,k} - \sum_{k\in S_i^-} \mathbf{T}'_{h,k} = -1, \quad \forall h \in \{m \in [n] \mid k_m^1 \text{ is even}\},$$
$$\sum_{k\in S_i^+} \mathbf{T}'_{m+1,k} - \sum_{k\in S_i^-} \mathbf{T}'_{m+1,k} = 0.$$

Step 2. Assume S_{i+1} is such that $|S_i| + |S_{i+1}|$ is even. For each $k \in S_{i+1}$, we follow the opposite of the procedure we applied to partition columns in S_i, that is, if the columns with odd indices are put in Q^+ for S_i, we put columns with even indices in Q^+. Otherwise, vice versa. Then, we obtain

$$\sum_{k\in S_i^+} \mathbf{T}'_{m+1,k} - \sum_{k\in S_i^-} \mathbf{T}'_{m+1,k} = 0.$$

Assume S_{i+1} is such that $|S_i| + |S_{i+1}|$ is odd. Then, we apply the same procedure we followed for S_i to partition columns into Q^+ and Q^- for S_{i+1} and obtain

$$\sum_{k\in S_i^+} \mathbf{T}'_{h,k} - \sum_{k\in S_i^-} \mathbf{T}'_{h,k} = 1, \quad \forall h \in \{m \in [n] \mid k_m^1 \text{ is odd}\},$$
$$\sum_{k\in S_i^+} \mathbf{T}'_{h,k} - \sum_{k\in S_i^-} \mathbf{T}'_{h,k} = -1, \quad \forall h \in \{m \in [n] \mid k_m^1 \text{ is even}\},$$
$$\sum_{k\in S_i^+} \mathbf{T}'_{m+1,k} - \sum_{k\in S_i^-} \mathbf{T}'_{m+1,k} = 1.$$

Step 3. If $|S_i| + |S_{i+1}|$ is even, go to Step 1 to partition columns in S_{i+2}. Otherwise, for each $k \in S_{i+2}$, apply the opposite of the procedure we used for S_{i+1} to partition columns for S_{i+2}. Then, we analyze the following two cases.

- $|S_{i+2}|$ is odd.

$$\sum_{k \in S_i^+} \mathbf{T}'_{h,k} - \sum_{k \in S_i^-} \mathbf{T}'_{h,k} = 0, \quad \forall h \in \{m \in [n] \mid k_m^i \text{ is odd}\},$$
$$\sum_{k \in S_i^+} \mathbf{T}'_{h,k} - \sum_{k \in S_i^-} \mathbf{T}'_{h,k} = -1, \quad \forall h \in \{m \in [n] \mid k_m^i \text{ is even}\},$$
$$\sum_{k \in S_i^+} \mathbf{T}'_{m+1,k} - \sum_{k \in S_i^-} \mathbf{T}'_{m+1,k} = 0.$$

- $|S_{i+2}|$ is even.

$$\sum_{k \in S_i^+} \mathbf{T}'_{h,k} - \sum_{k \in S_i^-} \mathbf{T}'_{h,k} = 1, \quad \forall h \in \{m \in [n] \mid k_m^i \text{ is odd}\},$$
$$\sum_{k \in S_i^+} \mathbf{T}'_{h,k} - \sum_{k \in S_i^-} \mathbf{T}'_{h,k} = 0, \quad \forall h \in \{m \in [n] \mid k_m^i \text{ is even}\},$$
$$\sum_{k \in S_i^+} \mathbf{T}'_{m+1,k} - \sum_{k \in S_i^-} \mathbf{T}'_{m+1,k} = 1.$$

Then, we go to Step 2 to partition columns for S_{i+3}.

We terminate this procedure when we partitioned the columns of S_i for all $i = 1, \ldots, d$ into Q^+ or Q^-. Notice that, at the end of each step, we obtain two disjoint sets S_i^+ and S_i^- such that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries only in $\{-1, 0, 1\}$. Thus, regardless of when we terminate the procedure, Proposition 6 is satisfied as we have $S_d^+ = Q^+$ and $S_d^- = Q^-$. □

B.1.2 Proof of Theorem 5

Let $(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{w}, \boldsymbol{\pi}) \in \mathcal{F}_{AM}$ be a feasible solution for the LP relaxation of formulation (3.2) and $(\mathbf{M}, \bar{\mathbf{N}}, \mathbf{N}, \boldsymbol{\alpha}, \mathbf{f}, \mathbf{b}, \mathbf{d}) \in \mathbf{proj}_{(\mathbf{M}, \bar{\mathbf{N}}, \mathbf{N}, \boldsymbol{\alpha}, \mathbf{f}, \mathbf{b}, \mathbf{d})}(\mathcal{F}'_{AM})$. We show that $(\mathbf{M}, \bar{\mathbf{N}}, \mathbf{N}, \boldsymbol{\alpha}, \mathbf{f}, \mathbf{b}, \mathbf{d})$ is

a feasible solution for the LP relaxation of formulation (3.5), i.e., $(\mathbf{M}, \bar{\mathbf{N}}, \mathbf{N}, \boldsymbol{\alpha}, \mathbf{f}, \mathbf{b}, \mathbf{d}) \in \mathcal{F}_{BD}$.

To establish that constraint (3.5b) holds, let us fix $a \in \mathcal{A}$ and $\ell \in \textbf{leaves}$. If we substitute $\bar{N}_\ell = \sum_{m=1}^{n} y_{m,\ell}$, M_ℓ and $N_{a,\ell} = \frac{1}{2}\sum_{m=1}^{n}(1+\bar{Y}_{m,a})y_{m,\ell}$, we need to prove that

$$M_\ell = \bar{N}_\ell - \sum_{m=1}^{n}\sum_{a' \in \mathcal{A}} c_{m,a'} \min\{y_{m,\ell}, \pi_{\ell,a'}\} \geq \bar{N}_\ell - \frac{1}{2}\sum_{m=1}^{n}(1+\bar{Y}_{m,a})y_{m,\ell} - n(1-\pi_{\ell,a}),$$

which is equivalent to

$$\sum_{m=1}^{n}\sum_{a \in \mathcal{A}} c_{m,a} \min\{y_{m,\ell}, \pi_{\ell,a}\} \leq \frac{1}{2}\sum_{m=1}^{n}(1+\bar{Y}_{m,a})y_{m,\ell} + n(1-\pi_{\ell,a}).$$

Observe that $c_{m,a} = \sum_{m=1}^{n}(1+\bar{Y}_{m,a})/2$; thus, we need to show that

$$\sum_{m=1}^{n}\sum_{a' \in \mathcal{A}} c_{m,a'} \min\{y_{m,\ell}, \pi_{\ell,a'}\} \leq \sum_{m=1}^{n} c_{m,a} y_{m,\ell} + n(1-\pi_{\ell,a}).$$

We can rewrite this expression as

$$\sum_{m=1}^{n}\sum_{a \in \mathcal{A}} c_{m,a} \min\{y_{m,\ell}, \pi_{\ell,a}\} \leq \sum_{m=1}^{n} c_{m,a} y_{m,\ell} + \sum_{m=1}^{n}(1-\pi_{\ell,a}),$$

which is satisfied, if

$$\sum_{a' \in \mathcal{A}} c_{m,a'} \min\{y_{m,\ell}, \pi_{\ell,a'}\} \leq c_{m,a} y_{m,\ell} + (1-\pi_{\ell,a})$$

holds for all $m \in [n]$. Note that, to maintain the equivalence of formulation (3.2) and (3.5), we assume that $c_{m,a} = 1$ is satisfied by only one $a \in \mathcal{A}$ for all $m \in [n]$. For an observation $m \in [n]$, let $c_{m,\tilde{a}} = 1$ for some $\tilde{a} \in \mathcal{A}$. Then, for observation $m \in [n]$, it is sufficient to show that

$$\min\{y_{m,\ell}, \pi_{\ell,\tilde{a}}\} \leq c_{m,a} y_{m,\ell} + (1-\pi_{\ell,a}),$$

for all $a \in \mathcal{A}$ and $\ell \in \textbf{leaves}$. Then, for all $\ell \in \textbf{leaves}$, we analyze the following cases:

- **Case 1:** $a = \tilde{a}$. Since, $\min\{y_{m,\ell}, \pi_{\ell,\tilde{a}}\} \leq y_{m,\ell}$ and $1 - \pi_{\ell,a} \geq 0$, $\min\{y_{m,\ell}, \pi_{\ell,\tilde{a}}\} \leq y_{m,\ell} + (1-\pi_{\ell,a})$ holds.

- **Case 2:** $a \neq \tilde{a}$. Then, $c_{m,a} = 0$. Observe that

$$\min\{y_{m,\ell}, \pi_{\ell,\tilde{a}}\} \leq \pi_{\ell,\tilde{a}} \leq \sum_{a' \in \mathcal{A}} \pi_{\ell,a'} - \pi_{\ell,a} = 1 - \pi_{\ell,a},$$

  by constraint (3.2f) and $1 \geq \pi_{\ell,a} \geq 0$.

Therefore, we conclude that constraint (3.5b) is satisfied.

Constraint (3.5d) is satisfied, since for all $m \in [n]$ and $\ell \in \textbf{leaves}$, $y_{m,\ell} \geq \sum_{a \in \mathcal{A}} c_{m,a} \min\{y_{m,\ell}, \pi_{\ell,a}\}$, which implies $\bar{N}_\ell = \sum_{m=1}^n y_{m,\ell} \geq \sum_{m=1}^n \sum_{a \in \mathcal{A}} c_{m,a} \min\{y_{m,\ell}, \pi_{\ell,a}\}$.

Constraints (3.5e) and (3.5f) are automatically satisfied because of how $\bar{N}_\ell$ and $N_{a,\ell}$ are constructed.

To show constraint (3.5g), fix an $m \in [n]$, $\ell \in \textbf{leaves}$ and $s \in A_R(\ell)$. Observe that since $s \in A_R(\ell)$, we automatically have that $\ell \in \textbf{right}(s)$. We now upper bound the right hand

side of constraint (3.5g):

$$
\begin{aligned}
& b_s - 1 + (1+\epsilon) y_{m,\ell} \\
&= \sum_{i=1}^{d} \sum_{j} \omega_{i,j} \lambda_{s,i,j} - 1 + (1+\epsilon) \cdot y_{m,\ell} \\
&\leq \sum_{i=1}^{d} \sum_{j} \omega_{i,j} \lambda_{s,i,j} - 1 + (1+\epsilon) \cdot \sum_{\ell' \in \mathbf{right}(s)} y_{m,\ell'} \\
&\leq \sum_{i=1}^{d} \sum_{j} \omega_{i,j} \lambda_{s,i,j} - 1 + (1+\epsilon) \cdot \left[1 - \sum_{i=1}^{d} \sum_{j : j \geq t_{i,m}} \lambda_{s,i,j} \right] \\
&= \sum_{i=1}^{d} \sum_{j} \omega_{i,j} \lambda_{s,i,j} - \sum_{i=1}^{d} \sum_{j} \lambda_{s,i,j} + (1+\epsilon) \sum_{i=1}^{d} \sum_{j : j < t_{i,m}} \lambda_{s,i,j} \\
&= \sum_{i=1}^{d} \sum_{j} \omega_{i,j} \lambda_{s,i,j} - \sum_{i=1}^{d} \sum_{j} \lambda_{s,i,j} + (1+\epsilon) \sum_{i=1}^{d} \sum_{j} \mathbb{I}\{j < \tau_{i,m}\} \lambda_{s,i,j} \qquad \text{(B.6)} \\
&= \sum_{i=1}^{d} \sum_{j} \left[\omega_{i,j} - 1 + (1+\epsilon) \mathbb{I}\{j < \tau_{i,m}\} \right] \cdot \lambda_{s,i,j} \qquad \text{(B.7)} \\
&\leq \sum_{i=1}^{d} \sum_{j} X_j^m \lambda_{s,i,j} \qquad \text{(B.8)} \\
&= \sum_{i=1}^{d} f_{s,i} X_i^m. \qquad \text{(B.9)}
\end{aligned}
$$

In the above sequence, the first equality follows from the definition of b_s. The first inequality follows since $\ell \in \mathbf{right}(s)$ (and thus $y_{m,\ell} \leq \sum_{\ell' \in \mathbf{right}(s)} y_{m,\ell'}$). The second inequality follows by re-arranging constraint 3.2e. The second equality follows by using constraint (3.2c). The third equality follows by the definition of the indicator function $\mathbb{I}\{\cdot\}$. The fourth equality follows by algebra. The final inequality follows by observing that each coefficient of $\lambda_{s,i,j}$ is

bounded by X_i^m; in particular, when $j < \tau_{i,m}$, the definition of τ and ω gives us that

$$\begin{aligned}
&\omega_{i,j} - 1 + (1+\epsilon)\mathbb{I}\{j < \tau_{i,m}\} \\
&= \omega_{i,j} - 1 + (1+\epsilon) \\
&= \omega_{i,j} + \epsilon \\
&\leq \omega_{i,\tau_{i,m}} \\
&= X_i^m,
\end{aligned}$$

and when $j \geq \tau_{i,m}$, we have

$$\begin{aligned}
&\omega_{i,j} - 1 + (1+\epsilon)\mathbb{I}\{j < \tau_{i,m}\} \\
&= \omega_{i,j} - 1 \\
&\leq X_i^m,
\end{aligned}$$

which holds because each X_i^m and $\omega_{i,j}$ belongs to the interval $[0, 1]$. (Recall from Section 3.3.1 that the features are assumed to be normalized so that each X_i^m satisfies $0 \leq X_i^m \leq 1$.) The last equality (B.9) then simply follows from our construction of the $f_{s,i}$ variables.

To show constraint (3.5h), we proceed in a similar fashion as for constraint (3.5g). Let m $\in [n]$, $\ell \in$ **leaves** and $s \in A_L(\ell)$. Since $s \in A_L(\ell)$, we immediately have that $\ell \in$ **left**(s).

We now lower bound the right hand side of constraint (3.5h):

$$b_s + 1 - y_{m,\ell} \tag{B.10}$$

$$= \sum_{i=1}^{d} \sum_{j} \omega_{i,j} \lambda_{s,i,j} + 1 - y_{m,\ell} \tag{B.11}$$

$$\geq \sum_{i=1}^{d} \sum_{j} \omega_{i,j} \lambda_{s,i,j} + 1 - \sum_{\ell' \in \mathbf{left}(s)} y_{m,\ell'} \tag{B.12}$$

$$\geq \sum_{i=1}^{d} \sum_{j} \omega_{i,j} \lambda_{s,i,j} + \sum_{i=1}^{d} \sum_{j : j < \tau_{i,m}} \lambda_{s,i,j} \tag{B.13}$$

$$= \sum_{i=1}^{d} \sum_{j} \left[\omega_{i,j} + \mathbb{I}\{j < \tau_{i,m}\} \right] \lambda_{s,i,j} \tag{B.14}$$

$$\geq \sum_{i=1}^{d} \sum_{j} X_i^m \lambda_{s,i,j} \tag{B.15}$$

$$= \sum_{i=1}^{d} f_{s,i} \lambda_{s,i,j}. \tag{B.16}$$

The steps above follow along similar lines as constraint (3.5g). The first inequality follows since $\ell \in \mathbf{left}(s)$; the second inequality follows by constraint (3.2d); and the final inequality again follows because when $j < \tau_{i,m}$, we have

$$\omega_{i,j} + \mathbb{I}\{j < \tau_{i,m}\} = \omega_{i,j} + 1 \geq X_i^m$$

(since X_i^m and $\omega_{i,j}$ are both in $[0, 1]$) and when $j \geq \tau_{i,m}$, we have

$$\omega_{i,j} + \mathbb{I}\{j < \tau_{i,m}\} = \omega_{i,j} \geq X_i^m.$$

Constraint (3.5i) is the same as constraint (3.2b). Constraint (3.5j) holds by the definition of $f_{s,i}$ and constraint (3.2c). Constraint (3.5k) holds because of the definition of b_s as $\sum_{i=1}^{d} \sum_{j} \omega_{i,j} \lambda_{s,i,j}$, and the fact that each $\omega_{i,j}$ lies in $[0, 1]$. $\square$

B.1.3 Proof of Proposition 5

We have shown that given $\mathbf{y} \in \{0,1\}$, the feasible region remains the same when binary variables λ are relaxed in Proposition 3. Thus, we first show that all extreme points have integer $\mathbf{y}$ values if we relax all binary variables in the single observation case. This implies that $\boldsymbol{\lambda}$ variables must be integers as well by Proposition 3. We then show that $\mathbf{w}$ and $\boldsymbol{\pi}$ also have integer values.

Assume to the contrary that there exists an extreme point where $y_\ell \in (0,1)$ for some $\ell \in \textbf{leaves}$. $\sum_{\ell \in \textbf{leaves}} y_\ell = 1$ implies that we have at least two leaves with fractional y variables. Let $\ell_p, \ell_r \in \textbf{leaves}$ be such that $y_{\ell_p}, y_{\ell_r} \in (0,1)$. We define

$$\begin{aligned} y^1_{\ell_p} &= y_{\ell_p} + \epsilon, \\ y^1_{\ell_r} &= y_{\ell_r} - \epsilon, \\ y^2_{\ell_p} &= y_{\ell_p} - \epsilon, \\ y^2_{\ell_r} &= y_{\ell_r} + \epsilon, \end{aligned}$$

where $\epsilon > 0$. Additionally, we define

$$\begin{aligned} w^1_{\ell_p,a_p} &= w_{\ell_p,a_p} + \epsilon, \\ w^1_{\ell_r,a_r} &= w_{\ell_r,a_r} - \epsilon, \\ w^2_{\ell_p,a_p} &= w_{\ell_p,a_p} - \epsilon, \\ w^2_{\ell_r,a_r} &= w_{\ell_r,a_r} + \epsilon, \end{aligned}$$

where $a_p, a_r \in \mathcal{A}$ satisfy $w_{\ell_p,a_p} > 0$ and $w_{\ell_r,a_r} > 0$, respectively. We construct π^1, π^2 in the following way. If $w_{\ell_p,a_p} < \pi_{\ell_p,a_p}$ and $w_{\ell_r,a_r} < \pi_{\ell_r,a_r}$, we set

$$\pi^1 = \pi^2 = \pi.$$

Otherwise, for some $a'_p \neq a_p$ such that $\pi_{\ell_p,a'_p} > 0$, we set

$$\pi^1_{\ell_p,a_p} = \pi_{\ell_p,a_p} + \epsilon,$$
$$\pi^1_{\ell_p,a'_p} = \pi_{\ell_p,a'_p} - \epsilon,$$
$$\pi^2_{\ell_p,a_p} = \pi_{\ell_p,a_p} - \epsilon,$$
$$\pi^2_{\ell_p,a'_p} = \pi_{\ell_p,a'_p} + \epsilon.$$

Similarly, for some $a'_r \neq a_r$ such that $\pi_{\ell_r,a'_r} > 0$, we set

$$\pi^1_{\ell_r,a_r} = \pi_{\ell_r,a_r} - \epsilon,$$
$$\pi^1_{\ell_r,a'_r} = \pi_{\ell_r,a'_r} + \epsilon,$$
$$\pi^2_{\ell_r,a_r} = \pi_{\ell_r,a_r} + \epsilon,$$
$$\pi^2_{\ell_r,a'_r} = \pi_{\ell_r,a'_r} - \epsilon.$$

We now construct λ^1 and λ^2 for each split $s \in \textbf{splits}$.

Case 1: Assume that $\ell_p \in \textbf{left}(s)$ and $\ell_r \in \textbf{right}(s)$ for some $s \in \textbf{splits}$. We analyze the following two cases.

- $\sum_{i=1}^d \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} \leq 1 - \sum_{\ell \in \textbf{left}(s)} y_{m,\ell}$ and $\sum_{i=1}^d \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} \leq 1 - \sum_{\ell \in \textbf{right}(s)} y_{m,\ell}$. We define
$$\lambda^1_{s,i} = \lambda^2_{s,i} = \lambda_{s,i}, \quad \forall i \in [d].$$
- $\sum_{i=1}^d \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} = 1 - \sum_{\ell \in \textbf{left}(s)} y_{m,\ell}$ or $\sum_{i=1}^d \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} = 1 - \sum_{\ell \in \textbf{right}(s)} y_{m,\ell}$. Without loss of generality we may assume $\sum_{i=1}^d \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} = 1 - \sum_{\ell \in \textbf{left}(s)} y_{m,\ell}$. For some $i' \in [d]$, $j' \in J[i']$ such that $\lambda_{s,i',j'} > 0$, we construct
$$\lambda^1_{s,i',j'} = \lambda_{s,i',j'} + \epsilon,$$
$$\lambda^2_{s,i',j'} = \lambda_{s,i',j'} - \epsilon.$$

Case 2: Consider $s \in \textbf{splits}$ such that $y_{\ell_p}, y_{\ell_r} \in \textbf{left}(s)$ or $y_{\ell_p}, y_{\ell_r} \in \textbf{right}(s)$. We can set

$$\lambda^1_{s,i,j} = \lambda^2_{s,i,j} = \lambda_{s,i,j} \quad \forall i \in [d], \quad j \in J[i].$$

Case 3: Consider $s \in \textbf{splits}$ such that $y_{\ell_p} \in \textbf{left}(s)$ and $y_{\ell_r} \notin \textbf{left}(s), \textbf{right}(s)$. Note that we assume for all $\ell \in \textbf{right}(s)$, we have $y_\ell = 0$. Otherwise, we could pick ℓ_r as the $\ell \in \textbf{right}(s)$ that satisfies $y_\ell \in (0,1)$ and construct solutions as in Case 1.

- $\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} < 1 - \sum_{\ell \in \textbf{left}(s)} y_{m,\ell}$. We can set

$$\lambda^1_{s,i,j} = \lambda^2_{s,i,j} = \lambda_{s,i,j}, \quad \forall i \in [d], j \in J[i].$$

- $\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} = 1 - \sum_{\ell \in \textbf{left}(s)} y_{m,\ell}$. Note that we then have $\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} < 1 - \sum_{\ell \in \textbf{right}(s)} y_{m,\ell}$. For some $i' \in [d]$ and $j' \in J[i']$ such that $\lambda_{s,i',j'} > 0$, we construct

$$\lambda^1_{s,i',j'} = \lambda_{s,i',j'} + \epsilon,$$
$$\lambda^2_{s,i',j'} = \lambda_{s,i',j'} - \epsilon.$$

Case 4: Consider $s \in \textbf{splits}$ such that $y_{\ell_p} \in \textbf{right}(s)$ and $y_{\ell_r} \notin \textbf{left}(s), \textbf{right}(s)$. Observe that this is symmetric with Case 3. Therefore, $\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} < 1 - \sum_{\ell \in \textbf{right}(s)} y_{m,\ell}$ is analogous to $\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m}} \lambda_{s,i,j} < 1 - \sum_{\ell \in \textbf{left}(s)} y_{m,\ell}$ in Case 3. If we have $\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{s,i,j} = 1 - \sum_{\ell \in \textbf{right}(s)} y_{m,\ell}$, For some $i' \in [d]$ and $j' \in J[i']$ such that $\lambda_{s,i',j'} > 0$, we construct

$$\lambda^1_{s,i',j'} = \lambda_{s,i',j'} - \epsilon,$$
$$\lambda^2_{s,i',j'} = \lambda_{s,i',j'} + \epsilon.$$

In all of the cases above, $(\mathbf{y}, \boldsymbol{\lambda}, \mathbf{w}, \boldsymbol{\pi})$ can be written as a convex combination of $(\mathbf{y}^1, \boldsymbol{\lambda}^1, \mathbf{w}^1, \boldsymbol{\pi}^1)$ and $(\mathbf{y}^2, \boldsymbol{\lambda}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$. Since $(\mathbf{y}^1, \boldsymbol{\lambda}^1, \mathbf{w}^1, \boldsymbol{\pi}^1)$ and $(\mathbf{y}^2, \boldsymbol{\lambda}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$ are feasible, this contradicts with the assumption that $(\mathbf{y}, \boldsymbol{\lambda}, \mathbf{w}, \boldsymbol{\pi})$ is an extreme point.

Now, we assume that there exists an extreme point $(\mathbf{y}, \boldsymbol{\lambda}, \mathbf{w}, \boldsymbol{\pi})$ such that $w_{\ell,a'} \in (0,1)$ for $a' \in \mathcal{A}$. Since y_ℓ cannot be continuous, there must exist $a'' \in \mathcal{A}$ such that $w_{\ell,a''} \in (0,1)$

and $a'' \neq a'$. Then, we construct $(\mathbf{y}^1, \boldsymbol{\lambda}^1, \mathbf{w}^1, \boldsymbol{\pi}^1)$ and $(\mathbf{y}^2, \boldsymbol{\lambda}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$ such that

$$
\begin{aligned}
w^1_{\ell,a'} &= w_{\ell,a'} + \epsilon, \\
w^1_{\ell,a''} &= w_{\ell,a''} - \epsilon, \\
w^2_{\ell,a'} &= w_{\ell,a'} - \epsilon, \\
w^2_{\ell,a''} &= w_{\ell,a''} + \epsilon, \\
\pi^1_{\ell,a'} &= \pi_{\ell,a'} + \epsilon, \\
\pi^1_{\ell,a''} &= \pi_{\ell,a''} - \epsilon, \\
\pi^2_{\ell,a'} &= \pi_{\ell,a'} - \epsilon, \\
\pi^2_{\ell,a''} &= \pi_{\ell,a''} + \epsilon.
\end{aligned}
$$

Observe that $(\mathbf{y}^1, \boldsymbol{\lambda}^1, \mathbf{w}^1, \boldsymbol{\pi}^1)$ and $(\mathbf{y}^2, \boldsymbol{\lambda}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$ we constructed are feasible and $(\mathbf{y}, \boldsymbol{\lambda}, \mathbf{w}, \boldsymbol{\pi})$ can be written as a convex combination of $(\mathbf{y}^1, \boldsymbol{\lambda}^1, \mathbf{w}^1, \boldsymbol{\pi}^1)$ and $(\mathbf{y}^2, \boldsymbol{\lambda}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$.

Finally, we assume that there exists an extreme point $(\mathbf{y}, \boldsymbol{\lambda}, \mathbf{w}, \boldsymbol{\pi})$ such that $\pi_{\ell,a'} \in (0, 1)$ for $a' \in \mathcal{A}$. By constraint (3.2f), there must exist $a'' \in \mathcal{A}$ such that $\pi_{\ell,a''} \in (0, 1)$ and $a'' \neq a'$. Then, we can construct feasible $(\mathbf{y}^1, \boldsymbol{\lambda}^1, \mathbf{w}^1, \boldsymbol{\pi}^1)$ and $(\mathbf{y}^2, \boldsymbol{\lambda}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$ as described above. Thus, this also contradicts with $(\mathbf{y}, \boldsymbol{\lambda}, \mathbf{w}, \boldsymbol{\pi})$ being an extreme point. □

B.1.4 Proof of Proposition 6

We consider the following observation from Iris data set.

$$
X^1_1 = 5.1, X^1_2 = 3.5, X^1_3 = 1.4, X^1_4 = 0.2, Y^1 = 1,
$$

Assume we have a decision tree of depth 2 with two splits. Let $\mathbf{splits} = \{1, 2\}$ and $\mathbf{leaves} = \{3, 4, 5\}$. Also, let $\mathcal{A} = \{1, 2, 3\}$. It can be verified that there exists an extreme

point to [16] formulation such that

$$\pi_{3,1} = 1/3, \pi_{3,2} = 1/3, \pi_{3,3} = 1/3,$$
$$\pi_{4,1} = 1/3, \pi_{4,2} = 1/3, \pi_{4,3} = 1/3,$$
$$\pi_{5,1} = 1/2, \pi_{5,2} = 0, \pi_{5,3} = 1/2.$$

B.1.5 Proof of Proposition 7

Let $\boldsymbol{\pi}$ be binary. We show that even if we relax all the other variables, $\boldsymbol{\lambda}$, $\mathbf{y}$ and $\mathbf{w}$ will have binary values in all extreme points in a depth 1 tree.

Note that, by Proposition 4, it suffices to show that $\boldsymbol{\lambda}$ is integral. Assume that there exists an extreme point $(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{w}, \boldsymbol{\pi})$ such that $\lambda_{s,i',j'} \in (0,1)$ for some $i' \in [d]$ and $j' \in J[i']$. Then, by constraint (3.2c), there exists $i'' \in [d]$ and $j'' \in J[i'']$ such that $\lambda_{s,i'',j''} \in (0,1)$. We construct two feasible solutions $(\boldsymbol{\lambda}^1, \mathbf{y}^1, \mathbf{w}^1, \boldsymbol{\pi}^1)$ and $(\boldsymbol{\lambda}^2, \mathbf{y}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$ as follows. For $i', i'' \in [d]$ and $j', j'' \in [d]$ and for a sufficiently small ϵ such that $\epsilon > 0$, we set

$$\lambda^1_{s,i',j'} = \lambda_{s,i',j'} + \epsilon,$$
$$\lambda^1_{s,i'',j''} = \lambda_{s,i'',j''} - \epsilon,$$
$$\lambda^2_{s,i'',j''} = \lambda_{s,i',j'} - \epsilon,$$
$$\lambda^2_{s,i'',j''} = \lambda_{s,i'',j''} + \epsilon.$$

For each $m \in [n]$, we analyze the following cases.

Case 1: $\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m}} \lambda_{1,i,j} < 1 - \sum_{\ell \in \mathbf{left}(1)} y_{m,\ell}$ and $\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{1,i,j} < 1 - \sum_{\ell \in \mathbf{right}(1)} y_{m,\ell}$. For $\epsilon > 0$, we set

$$y^1_{m,\ell} = y^2_{m,\ell} = y_{m,\ell}, \quad \forall \ell \in \mathbf{leaves},$$
$$w^1_{m,\ell,a} = w^2_{m,\ell,a} = w_{m,\ell,a}, \quad \forall \ell \in \mathbf{leaves}, \quad a \in \mathcal{A}.$$

Case 2: $\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m}} \lambda_{1,i,j} = 1 - \sum_{\ell \in \mathbf{left}(1)} y_{m,\ell}$ or $\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m}} \lambda_{1,i,j} = 1 - \sum_{\ell \in \mathbf{right}(1)} y_{m,\ell}$.

Without loss of generality, we assume $\sum_{i=1}^{d}\sum_{j:j<\tau_{i,m}}\lambda_{1,i,j} = 1 - \sum_{\ell\in\mathbf{left}(1)} y_{m,\ell}$. Then, we set consider the following sub-cases.

- $j' < \tau_{i',m}$ and $j'' < \tau_{i'',m}$. Then, we set

$$y^1_{m,\ell} = y^2_{m,\ell} = y_{m,\ell}, \quad \forall \ell \in \mathbf{left}(1),$$
$$w^1_{m,\ell,a} = w^2_{m,\ell,a} = w_{m,\ell,a}, \quad \forall \ell \in \mathbf{left}(1), \quad a \in \mathcal{A}.$$

- $j' < \tau_{i',m}$ and $j'' \geq \tau_{i'',m}$. Choose $\ell' \in \mathbf{left}(1)$ such that $y_{m,\ell'} > 0$ and $\ell'' \in \mathbf{right}(1)$ such that $y_{m,\ell''} > 0$. Also, choose a' and a'' such that $w_{m,\ell',a'} > 0$ and $w_{m,\ell'',a''} > 0$. Then, we set

$$y^1_{m,\ell'} = y^1_{m,\ell'} + \epsilon,$$
$$y^1_{m,\ell''} = y^1_{m,\ell''} - \epsilon,$$
$$y^2_{m,\ell'} = y^2_{m,\ell'} - \epsilon,$$
$$y^2_{m,\ell''} = y^2_{m,\ell''} + \epsilon,$$
$$w^1_{m,\ell',a'} = w^1_{m,\ell',a'} + \epsilon,$$
$$w^1_{m,\ell'',a''} = w^1_{m,\ell'',a''} - \epsilon,$$
$$w^2_{m,\ell',a'} = w^2_{m,\ell',a''} - \epsilon,$$
$$w^2_{m,\ell'',a''} = w^2_{m,\ell'',a''} + \epsilon.$$

- $j' \geq \tau_{i',m}$ and $j'' \geq \tau_{i'',m}$. This case is symmetric with the first case.

- $j' \geq \tau_{i',m}$ and $j'' < \tau_{i'',m}$. This case is symmetric with the second case.

Note that, if we also have $\sum_{i=1}^{d}\sum_{j:j\geq\tau_{i,m}}\lambda_{1,i,j} = 1 - \sum_{\ell\in\mathbf{right}(1)} y_{m,\ell}$, we can construct the corresponding variables in $(\boldsymbol{\lambda}^1, \mathbf{y}^1, \mathbf{w}^1, \boldsymbol{\pi}^1)$ and $(\boldsymbol{\lambda}^2, \mathbf{y}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$ in a similar way.

Notice that, $(\boldsymbol{\lambda}^1, \mathbf{y}^1, \mathbf{w}^1, \boldsymbol{\pi}^1)$ and $(\boldsymbol{\lambda}^2, \mathbf{y}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$ are feasible for formulation (3.2) and $(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{w}, \boldsymbol{\pi}) = 1/2(\boldsymbol{\lambda}^1, \mathbf{y}^1, \mathbf{w}^1, \boldsymbol{\pi}^1) + 1/2(\boldsymbol{\lambda}^2, \mathbf{y}^2, \mathbf{w}^2, \boldsymbol{\pi}^2)$ for all of the cases above. This contradicts with $(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{w}, \boldsymbol{\pi})$ being an extreme point. Thus, in all extreme points $\boldsymbol{\lambda}$ variables

are integral. By proposition 4, this implies that extreme points are integral in all variables. □

B.1.6 Proof of Proposition 8

We consider the following subset of Iris data set that consists of $n = 5$ observations.

$$X_1^1 = 4.4, X_2^1 = 2.9, X_3^1 = 1.4, X_4^1 = 0.2, Y^1 = 1,$$
$$X_1^2 = 5.5, X_2^2 = 2.6, X_3^2 = 4.4, X_4^2 = 1.2, Y^1 = 2,$$
$$X_1^3 = 7.7, X_2^3 = 3.8, X_3^3 = 6.7, X_4^3 = 2.2, Y^1 = 3,$$
$$X_1^4 = 6.6, X_2^4 = 2.9, X_3^4 = 4.6, X_4^4 = 1.3, Y^1 = 2,$$
$$X_1^5 = 6.0, X_2^5 = 3.0, X_3^5 = 4.8, X_4^5 = 1.8, Y^1 = 3.$$

It can be verified that in a tree where $|\mathbf{splits}| = 1$, there exists an extreme point for [16] formulation such that the following holds.

$$f_{1,1} = 0.426857, f_{2,1} = 0.573143,$$
$$b_1 = 0.142286,$$
$$\pi_{2,2} = 1, \pi_{3,3} = 1.$$

□

B.1.7 Proof of Proposition 9

Let $(\boldsymbol{\lambda}', \mathbf{y}')$ be a feasible to formulation (3.2) which satisfies constraints (3.2b), (3.2c) and constraints (3.2f)-(3.2l). For a split s, let $i' = \arg\max_{i\in[d]} \sum_{j\in J[i]} \lambda_{s,i,j}$, $m_1 = \arg\max_{m:\sum_{\ell\in\mathbf{left}(s)} y'_{m,\ell}=1} \tau_{i,m}$ and $m_2 = \arg\min_{m:\sum_{\ell\in\mathbf{right}(s)} y''_{m,\ell}=1} \tau_{i,m}$. Assume $\tau_{i,m_1} \geq \tau_{i,m_2}$. Then, we add the following constraints to the formulation according to Algorithm 1.

$$\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m_1}} \lambda_{s,i,j} \leq 1 - \sum_{\ell \in \mathbf{left}(s)} y_{m_1,\ell} \quad \forall s \in \mathbf{splits}, \tag{B.17}$$

$$\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m_2}} \lambda_{s,i,j} \leq 1 - \sum_{\ell \in \mathbf{right}(s)} y_{m_2,\ell} \quad \forall s \in \mathbf{splits}. \tag{B.18}$$

Let $(\boldsymbol{\lambda}'', \mathbf{y}'')$ be the new candidate solution we obtain after adding constraints (B.17) and (B.18) to the formulation. Assume to the contrary that, $(\boldsymbol{\lambda}'', \mathbf{y}'')$ are such that $y' = y''$ and $\sum_{j\in J[i]} |\lambda'_{s,i,j} - \lambda''_{s,i,j}| \neq 1$ for all $i \in [d]$. Notice that, $\sum_{j\in J[i]} |\lambda'_{s,i,j} - \lambda''_{s,i,j}| = 1$ implies that $\boldsymbol{\lambda}''$ differs from $\boldsymbol{\lambda}'$ in the feature assigned to split s. Therefore, $\boldsymbol{\lambda}''$ should differ from $\boldsymbol{\lambda}'$ in the split value assigned to split s in feature i'. Without loss of generality, let $j'' \in J[i']$ be the split value used in split s according to $\boldsymbol{\lambda}''$, i.e., $\lambda''_{s,i',j''} > 0$. We analyze the following cases.

- $j'' < \tau_{i',m_1}$: Then, we must have $\sum_{i=1}^{d} \sum_{j:j<\tau_{i,m_1}} \lambda''_{s,i,j} > 0$, which implies $\sum_{\ell\in\mathbf{left}(s)} y''_{m_1,\ell} = 0$ according to constraint (B.17). This gives a contradiction, since we assumed $\mathbf{y}'' = \mathbf{y}'$ and $\sum_{\ell\in\mathbf{left}(s)} y'_{m_1,\ell} = 1$.

- $j'' \geq \tau_{i',m_1}$: Since $\tau_{i',m_1} \geq \tau_{i',m_2}$, we have $\sum_{i=1}^{d} \sum_{j:j\geq\tau_{i,m_2}} \lambda''_{s,i,j} > 0$. Then, constraint (B.18) implies that $\sum_{\ell\in\mathbf{right}(s)} y''_{m_2,\ell} = 0$. This gives a contradiction, since we assumed $\mathbf{y}'' = \mathbf{y}'$ and $\sum_{\ell\in\mathbf{right}(s)} y'_{m_2,\ell} = 1$.

Therefore, we prove that at least one of the following must hold.

(a) $\mathbf{y}'' \neq \mathbf{y}'$,

(b) The feature assigned to split s is different, i.e., $\sum_{j\in J[i]} |\lambda'_{s,i',j} - \lambda''_{s,i',j}| = 1$ for some $i' \in [d]$.

□

REFERENCES

[1] S. Aghaei, M. J. Azizi, and P. Vayanos. Learning optimal and fair decision trees for non-discriminative decision-making. *arXiv preprint arXiv:1903.10598*, 2019.

[2] S. Aghaei, A. Gomez, and P. Vayanos. Learning optimal classification trees: Strong max-flow formulations. *arXiv preprint arXiv:2002.09142*, 2020.

[3] İ. Akçakuş and V. V. Mišić. Exact logit-based product design. *arXiv preprint arXiv:2106.15084*, 2021.

[4] G. M. Allenby and J. L. Ginter. Using extremes to design products and segment markets. *Journal of Marketing Research*, 32(4):392–403, 1995.

[5] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, pages 1–37, 2020.

[6] S. P. Anderson, A. De Palma, and J.-F. Thisse. A representative consumer theory of the logit model. *International Economic Review*, pages 461–466, 1988.

[7] A. Aouad, A. N. Elmachtoub, K. J. Ferreira, and R. McNellis. Market segmentation trees. *Manufacturing & Service Operations Management*, 2023.

[8] A. Aouad, V. Farias, R. Levi, and D. Segev. The approximability of assortment optimization under ranking preferences. *Operations Research*, 66(6):1661–1669, 2018.

[9] A. Atamtürk, G. Berenguer, and Z.-J. M. Shen. A conic integer programming approach to stochastic joint location-inventory problems. *Operations Research*, 60(2):366–381, 2012.

[10] P. V. Balakrishnan and V. S. Jacob. Genetic algorithms for product design. *Management Science*, 42(8):1105–1117, 1996.

[11] G.-Y. Ban and C. Rudin. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1):90–108, 2019.

[12] O. Bastani, C. Kim, and H. Bastani. Interpretability via model extraction. *arXiv preprint arXiv:1706.09773*, 2017.

[13] A. Belloni, R. Freund, M. Selove, and D. Simester. Optimizing product line designs: Efficient methods and comparisons. *Management Science*, 54(9):1544–1552, 2008.

[14] K. P. Bennett and J. A. Blue. Optimal decision trees. *Rensselaer Polytechnic Institute Math Report*, 214:24, 1996.

[15] H. Y. Benson and U. Sağlam. *Mixed-Integer Second-Order Cone Programming: A Survey*, chapter Chapter 2, pages 13–36. INFORMS, 2013.

[16] D. Bertsimas and J. Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[17] D. Bertsimas, J. Dunn, G. C. Velmahos, and H. M. Kaafarani. Surgical risk is not linear: derivation and validation of a novel, user-friendly, and machine-learning-based predictive optimal trees in emergency surgery risk (potter) calculator. *Annals of surgery*, 268(4):574–583, 2018.

[18] D. Bertsimas, N. Kallus, A. M. Weinstein, and Y. D. Zhuo. Personalized diabetes management using electronic medical records. *Diabetes care*, 40(2):210–217, 2017.

[19] D. Bertsimas, A. King, and R. Mazumder. Best subset selection via a modern optimization lens. *Annals of Statistics*, 44(2):813–852, 2016.

[20] D. Bertsimas and V. V. Mišić. Robust product line design. *Operations Research*, 65(1):19–37, 2017.

[21] D. Bertsimas and V. V. Mišić. Exact first-choice product line optimization. *Operations Research*, 67(3):651–670, 2019.

[22] D. Bertsimas and R. Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas Belmont, 2005.

[23] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[24] R. E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pages 107–121, 2012.

[25] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[26] F. Bravo and Y. Shaposhnik. Mining optimal policies: A pattern recognition approach to model analysis. *INFORMS Journal on Optimization*, 2(3):145–166, 2020.

[27] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[28] J. D. Camm, J. J. Cochran, D. J. Curry, and S. Kannan. Conjoint optimization: An exact branch-and-bound algorithm for the share-of-choice problem. *Management Science*, 52(3):435–447, 2006.

[29] 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.

[30] L. Chen, L. He, and Y. H. Zhou. An exponential cone programming approach for managing electric vehicle charging. *Available at SSRN 3548028*, 2021.

[31] Y.-C. Chen and V. V. Mišić. Decision forest: A nonparametric approach to modeling irrational choice. *Management Science*, 68(10):7090–7111, 2022.

[32] D. F. Ciocan and V. V. Mišić. Interpretable optimal stopping. *Management Science*, forthcoming, 2020.

[33] C. Coey, M. Lubin, and J. P. Vielma. Outer approximation with conic certificates for mixed-integer convex problems. *Mathematical Programming Computation*, pages 1–45, 2020.

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

[35] D. Dua and C. Graff. UCI machine learning repository, 2017.

[36] I. Dunning, J. Huchette, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[37] S. Esmeir and S. Markovitch. Anytime learning of decision trees. *Journal of Machine Learning Research*, 8(May):891–933, 2007.

[38] V. F. Farias and A. A. Li. Learning preferences with side information. *Management Science*, 65(7):3131–3149, 2019.

[39] J. B. Feldman and H. Topaloglu. Revenue management under the markov chain choice model. *Operations Research*, 65(5):1322–1342, 2017.

[40] G. Feng, X. Li, and Z. Wang. On the relation between several discrete choice models. *Operations research*, 65(6):1516–1525, 2017.

[41] K. J. Ferreira, B. H. A. Lee, and D. Simchi-Levi. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, 18(1):69–88, 2016.

[42] G. Gallego and H. Topaloglu. Assortment optimization. In *Revenue Management and Pricing Analytics*, pages 129–160. Springer, 2019.

[43] P. E. Green, A. M. Krieger, and Y. Wind. *Buyer Choice Simulators, Optimizers, and Dynamic Models*, pages 169–199. Springer US, Boston, MA, 2004.

[44] O. Günlük, J. Kalagnanam, M. Menickelly, and K. Scheinberg. Optimal decision trees for categorical data via integer programming. *arXiv preprint arXiv:1612.03225*, 2018.

[45] J. Hainmueller, D. J. Hopkins, and T. Yamamoto. Causal inference in conjoint analysis: Understanding multidimensional choices via stated preference experiments. *Political analysis*, 22(1):1–30, 2014.

[46] J. Hastad. Clique is hard to approximate within n/sup 1-/spl epsiv. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 627–636. IEEE, 1996.

[47] J. Hofbauer and W. H. Sandholm. On the global convergence of stochastic fictitious play. *Econometrica*, 70(6):2265–2294, 2002.

[48] J. Huchette and J. P. Vielma. Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. *arXiv preprint arXiv:1708.00050*, 2017.

[49] 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.

[50] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

[51] P. Jaillet, G. G. Loke, and M. Sim. Strategic manpower planning under uncertainty. *Available at SSRN 3168168*, 2018.

[52] D. S. Johnson and M. R. Garey. *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, 1979.

[53] N. Kallus. Recursive partitioning for personalization using observational data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1789–1798. JMLR. org, 2017.

[54] R. Kohli, K. Boughanmi, and V. Kohli. Randomized algorithms for lexicographic inference. *Operations Research*, 67(2):357–375, 2019.

[55] R. Kohli and R. Krishnamurti. A heuristic approach to product design. *Management Science*, pages 1523–1533, 1987.

[56] R. Kohli and R. Krishnamurti. Optimal product design using conjoint analysis: Computational complexity and algorithms. *European Journal of Operational Research*, 40(2):186–195, 1989.

[57] R. Kohli and R. Sukumar. Heuristics for product-line design using conjoint analysis. *Management Science*, 36(12):1464–1478, 1990.

[58] M. Liu, Z. Pan, K. Xu, and D. Manocha. New formulation of mixed-integer conic programming for globally optimal grasp planning. *IEEE Robotics and Automation Letters*, 5(3):4663–4670, 2020.

[59] M. Lubin, Y. Dvorkin, and L. Roald. Chance constraints for improving the security of ac optimal power flow. *IEEE Transactions on Power Systems*, 34(3):1908–1917, 2019.

[60] M. Lubin, J. P. Vielma, and I. Zadik. Mixed-integer convex representability. *arXiv preprint arXiv:1706.05135*, 2017.

[61] M. Lubin, E. Yamangil, R. Bent, and J. P. Vielma. Polyhedral approximation in mixed-integer convex optimization. *Mathematical Programming*, 172(1):139–168, 2018.

[62] H.-Y. Mak, Y. Rong, and Z.-J. M. Shen. Infrastructure planning for electric vehicles with battery swapping. *Management Science*, 59(7):1557–1575, 2013.

[63] 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.

[64] V. V. Mišić. Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624, 2020.

[65] Mosek ApS. Mosek modeling cookbook, 2021.

[66] Mosek ApS. Mosek optimization suite, 2021.

[67] S. Murthy and S. Salzberg. Lookahead and pathology in decision tree induction. In *IJCAI*, pages 1025–1033. Citeseer, 1995.

[68] S. W. Norton. Generating better decision trees. In *IJCAI*, volume 89, pages 800–805, 1989.

[69] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[70] J. R. Quinlan. C4. 5: Programming for machine learning. *Morgan Kauffmann*, 38:48, 1993.

[71] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.

[72] L. Rokach and O. Maimon. Decision trees. In *Data mining and knowledge discovery handbook*, pages 165–192. Springer, 2005.

[73] P. E. Rossi. `bayesm`: Bayesian inference for marketing/micro-econometrics. r package version 3.1-4, 2019.

[74] P. Rusmevichientong, D. Shmoys, C. Tong, and H. Topaloglu. Assortment optimization under the multinomial logit model with random choice parameters. *Production and Operations Management*, 23(11):2023–2039, 2014.

[75] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[76] R. Schmalensee and J.-F. Thisse. Perceptual maps and the optimal location of new products: An integrative essay. *International Journal of Research in Marketing*, 5(4):225–249, 1988.

[77] C. Schön. On the optimal product line selection problem with price discrimination. *Management Science*, 56(5):896–902, 2010.

[78] L. Shi, S. Ólafsson, and Q. Chen. An optimization framework for product design. *Management Science*, 47(12):1681–1692, 2001.

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

[80] T. Therneau and B. Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2018. R package version 4.1-13.

[81] O. Toubia, D. I. Simester, J. R. Hauser, and E. Dahan. Fast polyhedral adaptive conjoint estimation. *Marketing Science*, 22(3):273–303, 2003.

[82] K. E. Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.

[83] M. Udell and S. Boyd. Maximizing a sum of sigmoids. *Optimization and Engineering*, pages 1–25, 2013.

[84] S. Verwer and Y. Zhang. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1625–1632, 2019.

[85] J. P. Vielma. Mixed integer linear programming formulation techniques. *Siam Review*, 57(1):3–57, 2015.

[86] 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.

[87] 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.

[88] X. Wang, J. D. Camm, and D. J. Curry. A branch-and-price approach to the share-of-choice product line design problem. *Management Science*, 55(10):1718–1728, 2009.

[89] Z. Zhou, S. Athey, and S. Wager. Offline multi-action policy learning: Generalization and optimization. *arXiv preprint arXiv:1810.04778*, 2018.

[90] T. Zhu, J. Xie, and M. Sim. Joint estimation and robustness optimization. *Management Science*, 2021.