OPERATIONS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0030-364X | EISSN 1526-5463 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000

Exact first-choice product line optimization

Dimitris Bertsimas
Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology; 77 Massachusetts Avenue, E40-147, Cambridge, MA, 02139, USA; dbertsim@mit.edu

Velibor V. Mišić
Anderson School of Management, University of California, Los Angeles; 110 Westwood Plaza, Los Angeles, CA, 90095, USA; velibor.misic@anderson.ucla.edu

A fundamental problem faced by firms is that of product line design: given a set of candidate products that may be offered to a collection of customers, what subset of those products should be offered so as to maximize the profit that is realized when customers make purchases according to their preferences? In this paper, we consider the product line design problem when customers choose according to a first-choice rule and present a new mixed-integer optimization formulation of the problem. We theoretically analyze the strength of our formulation and show that it is stronger than alternative formulations that have been proposed in the literature, thus contributing to a unified understanding of the different formulations for this problem. We also present a novel solution approach for solving our formulation at scale, based on Benders decomposition, which exploits the surprising fact that Benders cuts for both the relaxation and the integer problem can be generated in a computationally efficient manner. We demonstrate the value of our formulation and Benders decomposition approach through two sets of experiments. In the first, we use synthetic instances to show that our formulation is computationally tractable and can be solved an order of magnitude faster for small to medium scale instances than the alternate, previously proposed formulations. In the second, we consider a previously studied product line design instance based on a real conjoint data set, involving over 3000 candidate products and over 300 respondents. We show that this problem, which required a week of computation time to solve in prior work, is solved by our approach to full optimality in approximately ten minutes.

Key words: product line design; first-choice models; mixed-integer optimization; Benders decomposition.
History: Submitted August 16, 2017; revised June 28, 2018; accepted September 29, 2018.

1. Introduction

A key problem in marketing is to decide on what products to offer in the presence of customers who choose. The problem can be generally stated as follows: given a set of candidate products and a collection of customer types with different preferences for those products, what is the subset of those products that we should offer so as to maximize our profit when the customers make purchases consistent with their preferences? This problem is known as the *product line design* (PLD) problem.

The PLD problem is a notoriously difficult problem for two reasons. First, because the decision involves selecting a subset of products, the number of possible solutions to the problem is exponentially large. Second, it is not straightforward to select a good subset of products when customers exhibit substitution: adding a low profit product to attract customers from one segment may cannibalize demand from a higher profit product that is favored by a different segment, leading to lower overall profits. One cannot thus select products independently of each other.

A specific subclass of the PLD problem that has been studied significantly in the marketing literature is when customers behave according to a first-choice model. In this model, the customer population is described through a finite collection of customer types. Each customer type is endowed with a ranking of the set of candidate products. When offered a specific subset of products, all customers of the given type deterministically choose the product from the subset that is most

preferred (their first choice) according to their ranking. Such a model is extremely general, in that it can model customers with compensatory decision processes (i.e., customers evaluate the utility of each product and select the one with the highest utility) and non-compensatory decision processes (for example, customers evaluate products feature by feature according to a lexicographic rule).

In this paper, we consider a new methodology for solving the first-choice PLD problem. Our methodology is based on a mixed-integer optimization formulation of the problem, and as such is capable of providing provably optimal solutions at a large scale. Our specific contributions are as follows:

1. We propose a new MIO model of the PLD problem under a first-choice model of customer behavior. We theoretically analyze the formulation and compare it to three alternative formulations: the ranking-based formulation of Belloni et al. (2008), the utility-based formulation of McBride and Zufryden (1988) and an alternate utility formulation that, to the best of our knowledge, has not been proposed before, but improves on the formulation of McBride and Zufryden (1988). We show that the linear optimization (LO) relaxation of our formulation is tighter than that of all three alternative formulations. Moreover, we show that in the case of a single customer segment, our formulation is always integral (all extreme points of the LO relaxation are feasible for the integer problem), whereas the other formulations in general are not. In this way, our paper contributes to a unified theoretical understanding of different MIO formulations for the first-choice PLD problem.

2. We propose a novel solution method for solving our MIO model that is based on Benders decomposition. Leveraging the structure of the problem, we show that Benders cuts for integer solutions of the problem can be obtained in closed form. Surprisingly, we also show that Benders cuts for fractional solutions can be obtained in a similar way by applying a special greedy algorithm. In both cases, cuts can be obtained without using an LO solver, making the method computationally attractive for solving large-scale instances of the problem.

3. We computationally demonstrate the value of our methodology through two sets of experiments. In the first set, we use simulated data to show that our formulation can be solved to provable optimality in significantly less time and is significantly tighter than the other three formulations. In the second set of experiments, we use a real conjoint data set involving laptop bags from Toubia et al. (2003), and re-visit a product line design problem based on this data set first studied in Belloni et al. (2008). This problem involves 3584 products and 330 respondent utility functions, and required one week of computation time to solve to full optimality at the time of Belloni et al. (2008). We show that using our Benders solution approach, the same problem can be solved to full optimality in approximately ten minutes.

The rest of this paper is organized as follows. In Section 2, we provide a review of some of the related literature. In Section 3, we present our mixed-integer optimization model of the first-choice PLD problem, and provide a theoretical comparison of our model to other models for this problem. In Section 4, we propose our solution methodology based on Benders decomposition and prove results on the structure of Benders cuts for integer and fractional solutions. In Section 5, we provide the results of our numerical experiments with both simulated and real data. In Section 6, we conclude.

2. Literature review

The problem of product line design (PLD) has been well-studied in marketing. We briefly review some of the literature in this area; for a more detailed overview, we refer the reader to the literature review of Bertsimas and Mišić (2017). The PLD problem has been studied extensively under the utility-based first-choice model, where segments of customers deterministically choose the option that provides the maximum utility from the offered product line; examples include Green and Krieger (1985), Kohli and Krishnamurti (1987) McBride and Zufryden (1988), Dobson and Kalish (1988) Kohli and Sukumar (1990), Green and Krieger (1993), Belloni et al. (2008). In addition, the PLD problem has also been studied under probabilistic choice models, where each segment probabilistically chooses from the available options; examples include Chen and Hausman (2000), Schön

(2010a,b) and Kraus and Yano (2003). In addition, some papers have studied other dimensions of the problem. Luo (2011) considers PLD from a perspective that considers both marketing and engineering criteria and their interplay. Bertsimas and Mišić (2017) considers a robust optimization framework for product line decisions that allows a firm to account for both parameter and structural uncertainty in the customer choice model.

Within this literature, many papers consider heuristics; few consider directly formulating and solving the problem as an integer optimization (IO) problem. Closest to our paper are the papers of McBride and Zufryden (1988) and Belloni et al. (2008). The paper of McBride and Zufryden (1988) proposes an IO formulation of the utility-based first-choice PLD problem, where the utilities of each customer enter directly into the formulation. The paper of Belloni et al. (2008) is primarily concerned with performing a systematic comparison of different heuristics for the PLD problem; however, to objectively measure the quality of these heuristics, the paper formulates and solves the first-choice PLD problem as an IO problem. Due to the scale of the problem, the paper considers a custom solution approach that is primarily based on Lagrangean relaxation.

Our paper builds on and advances the state-of-the-art relative to these two papers in two regards. First, we show theoretically in Section 3.4 that our formulation is a tighter formulation than those of McBride and Zufryden (1988) and Belloni et al. (2008): that is, the feasible region of the LO relaxation of our formulation is contained within the feasible regions of the relaxations of McBride and Zufryden (1988) and Belloni et al. (2008). This is of great practical importance, as tighter MIO formulations of a problem are generally faster to solve. Indeed, we show in Section 5.1 using simulated data that for large instances, our formulation can be solved at least an order of magnitude faster than the formulations of McBride and Zufryden (1988) and Belloni et al. (2008).

Second, we recognize an important feature of our formulation, which is that the formulation can be divided into two sets of variables: a "global" set of variables, which dictates the set of products to be offered, and "local" sets of variables that specify which product is chosen by each segment. We leverage this structure to propose a solution algorithm based on Benders decomposition. To the best of our knowledge, such a solution approach has not been proposed for first-choice PLD models previously. Furthermore, the result that the optimal solution to each segment's subproblem can be obtained by a greedy algorithm when the global variables are fractional is unexpected and surprising; we are not aware of a similar result for other PLD models. Lastly, as mentioned in the introduction, we use this approach to solve the same product line problem for the same data set as in Belloni et al. (2008), and show that our approach can solve the problem optimality in roughly ten minutes, whereas the formulation and solution approach of Belloni et al. (2008) required one week of computation time.

Outside of the PLD literature, the problem studied in this paper is also found in the assortment optimization literature. In assortment optimization, the goal is to select a set of products to offer, where the products are products that have already been offered, as opposed to hypothetical products. Within the operations research and operations management communities, this problem has been studied under a wide variety of choice models, such as the multinomial logit (MNL) model (Talluri and van Ryzin 2004), the robust MNL model (Rusmevichientong and Topaloglu 2012), the nested logit model (Davis et al. 2014, Li et al. 2015), the latent class MNL model (Bront et al. 2009, Rusmevichientong et al. 2014) and the Markov chain choice model (Blanchet et al. 2016, Feldman and Topaloglu 2017).

With regard to first-choice/ranking-based models, there has been some previous work on solution methodologies for this problem. Specifically, the papers of Aouad et al. (2017) and Aouad et al. (2015) consider the problem of assortment optimization under ranking preferences, where the former paper studies the computational complexity of the problem and provides general approximation algorithms, while the latter paper studies the problem for specific families of first-choice models and develops efficient dynamic programming-based solution methods. The earlier paper of Honhon et al. (2012) also develops solution methods for specific families of first-choice models. Our approach differs from this prior work in several important ways. First, our approach is agnostic to

the type of first-choice model: our MIO formulation does not require any assumptions on the collection of customer types and the nature of the rankings over the products. Second, our approach is based on mixed-integer optimization; thus, it is capable of providing solutions that are provably optimal, as opposed to solutions that only come with an approximation guarantee. Moreover, by formulating the problem as an integer optimization problem, we can leverage both advances in solution software that are manifested in state-of-the-art solvers such as Gurobi (Gurobi Optimization, Inc. 2015), as well as generally-applicable solution methods developed in the operations research community (such as Benders decomposition, which we will see in Section 5.3 allows us to solve the problem to provable optimality at a very large scale).

3. Method

In this section, we describe our PLD model. We begin in Section 3.1 by describing the first-choice model and the first-choice PLD problem abstractly. In Section 3.2 we present our MIO formulation for finding the optimal product line under this choice model. In Section 3.3, we describe three alternate formulations for the first-choice PLD problem, and in Section 3.4, we theoretically compare our MIO formulation to these alternate formulations. Finally, in Section 3.5, we theoretically analyze the formulations for the special case when there is only a single customer segment.

3.1. Background

We assume that there are n candidate products, indexed from 1 to n, that may be included in the product line. We use the index 0 to denote the no-purchase alternative (the possibility that the customer does not purchase any of the products that we offer). Together, we refer to the set $\{0,1,2,\ldots,n\}$ – the set of products together with the no-purchase alternative – as the *options* that are available. We assume that a customer will select exactly one of the available options. We also assume that the no-purchase alternative 0 is always available to the customer.

We assume that we have K rankings $\sigma^1,\ldots,\sigma^K$ over the options $\{0,1,2,\ldots,n\}$. Each ranking $\sigma^k : \{0,1,2,\ldots,n\} \to \{0,1,2,\ldots,n\}$ is a bijection that assigns each option to a rank in $\{0,1,2,\ldots,n\}$. The value $\sigma^k(i)$ indicates the rank of option i; $\sigma^k(i) < \sigma^k(j)$ indicates that i is more preferred to j under the ranking σ^k. We assume that given a set of products $S \subseteq \{1,2,\ldots,n\}$, a customer that follows the ranking σ^k will select the option i from the set $S \cup \{0\}$ with the lowest rank, i.e., the option $\arg\min_{i\in S\cup\{0\}} \sigma^k(i)$. We can think of each k as a segment or a customer type, and σ^k as the choice behavior that customers of that customer type/segment will follow.

We use λ^k to denote the probability that a random customer makes a choice according to the ranking σ^k; we use $\boldsymbol{\lambda}$ to denote the probability mass function (PMF) over the set of rankings $\{\sigma^1,\ldots,\sigma^K\}$. The probability λ^k is the probability that a customer is of type k or equivalently, the relative size of the segment k. For a given product line $S \subseteq \{1,2,\ldots,n\}$, the probability $\mathbb{P}(i \,|\, S)$ that a random customer selects option $i \in \{0,1,2,\ldots,n\}$ given that the available set of products was S is given by

$$\mathbb{P}(i \,|\, S) = \sum_{k=1}^{K} \lambda^k \cdot \mathbb{I}\{i = \arg\min_{i'\in S\cup\{0\}} \sigma^k(i')\},$$

where $\mathbb{I}\{\cdot\}$ is the indicator function ($\mathbb{I}\{A\}$ is 1 if A is true and 0 otherwise).

The above choice model based on rankings is very general; we describe a few ways in which such a model may arise.

1. **Utility-based models**. A common choice model used in the PLD literature is to assume that customers maximize utility: each customer evaluates the utility of each option (one of the products in the product line or the no-purchase option), and chooses the option that yields the highest utility. In practice, such a model is formed by collecting conjoint data and estimating an attribute-level utility function using ordinary least squares or polyhedral methods (Toubia et al. 2003, 2004) for each of the K customer types. When the products are described by attributes, the

resulting utility function of each customer type k allows us to compute a utility u_j^k for each option $j \in \{0, 1, \ldots, n\}$. When offered S, each customer type will then choose the option that yields the highest utility, which is given by $\arg\max_{j \in S \cup \{0\}} u_j^k$. Such a model of choice can be represented in the above framework by defining the ranking of each customer type k to be the ranking that is consistent with the utilities of customer type k, that is,

$$\sigma^k(i) < \sigma^k(j) \text{ if and only if } u_i^k > u_j^k.$$

2. **Lexicographic rules**. A separate stream of research in marketing has considered the use of lexicographic rules as a model for describing choice behavior. In such a model, each customer has an ordering of the product attributes from most to least important. Given a set of products, the customer selects the product that has the most important attribute; ties are broken according to the second most important attribute, third most important attribute, and so on. Such models can be estimated by a variety of approaches (see for example Yee et al. 2007, Kohli and Jedidi 2015, Kohli et al. 2018); for an excellent overview, the reader is referred to the paper of Kohli et al. (2018). A lexicographic rule defines a fixed ranking over the products; as such, lexicographic choice models are subsumed within the ranking-based choice model presented above.

3. **Aggregate estimation**. Outside of PLD, we also note that the above choice model is significant in the context of assortment optimization. In particular, Farias et al. (2013) pioneered the use of the above model for making choice predictions from limited aggregate transaction data for historical product assortments. Their approach involves considering all $(n+1)!$ rankings of the $n+1$ options, and finding the worst-case probability distribution $\boldsymbol{\lambda}$ such that $\mathbb{P}(i \,|\, S)$ is consistent with the transaction data for those assortments S that have been previously offered. This approach can be considered a nonparametric approach, as any choice model for the n products that is based on random utility maximization (such as the MNL model, the nested logit model and the latent class MNL model) can be represented as a probability distribution over rankings. Since the paper of Farias et al. (2013), other research has considered approaches for estimating $\boldsymbol{\lambda}$ and a collection of rankings $\sigma^1, \ldots, \sigma^K$ so as to fit some aggregate data (van Ryzin and Vulcano 2015, Mišić 2016). A ranking-based model estimated from such an approach can be used directly in the optimization problem; for more details, see Mišić (2016).

In the approach that we will present in Section 3.2, we do not make any assumptions on the rankings $\sigma^1, \ldots, \sigma^K$ and their probabilities $\lambda^1, \ldots, \lambda^K$.

We now define our optimization problem abstractly. Let π_i be the profit of option i; we assume that the profit π_0 of the no-purchase alternative is exactly zero. Then the expected per-customer profit from offering the product line S is denoted by $\Pi(S)$ and is given by

$$\begin{aligned}\Pi(S) &= \sum_{i \in S} \pi_i \cdot \mathbb{P}(i \,|\, S) \\ &= \sum_{i \in S} \pi_i \cdot \left(\sum_{k=1}^{K} \lambda^k \cdot \mathbb{I}\{i = \arg\min_{i' \in S \cup \{0\}} \sigma^k(i')\} \right).\end{aligned}$$

Having defined the first-choice model we will be using, we now define the optimization problem. Let $\mathcal{P}(\{1, \ldots, n\})$ be the power set of $\{1, \ldots, n\}$, i.e., the collection of all subsets of $\{1, \ldots, n\}$, and let $\mathcal{S} \subseteq \mathcal{P}(\{1, \ldots, n\})$ be a collection of permissible product lines. The problem we wish to solve is to find the set of products S^* from $\mathcal{S}$ that maximizes the expected profit under the above first-choice model:

$$S^* = \arg\max_{S \in \mathcal{S}} \Pi(S). \tag{1}$$

The above optimization problem is known to be theoretically intractable. Kohli and Krishnamurti (1989) showed that it is NP-Hard in general. Later, Aouad et al. (2017) showed that it is NP-Hard to find a solution that is within $O(n^{1-\epsilon})$ of optimal for any $\epsilon > 0$.

3.2. Mixed-integer optimization model

To solve this problem, we will formulate it as an MIO problem. For each product $i \in \{1,\ldots,n\}$, let x_i be a binary decision variable that is 1 if product i is included in the product line, and 0 otherwise. Note that given $\mathbf{x} \in \{0,1\}^n$, we can recover the set of products $S(\mathbf{x})$ that correspond to $\mathbf{x}$ as $S(\mathbf{x}) = \{i \in \{1,\ldots,n\} \,|\, x_i = 1\}$. For each option $i \in \{0,1,\ldots,n\}$, let y_i^k be a decision variable that is 1 if option i is chosen under the kth ranking, and 0 otherwise.

Before defining our problem, we make the following assumption about the set of permissible product lines $\mathcal{S}$.

ASSUMPTION 1. *There exists a matrix $\mathbf{C} \in \mathbb{R}^{m\times n}$ and vector $\mathbf{d} \in \mathbb{R}^m$ such that the set of permissible product lines $\mathcal{S}$ is encoded by all binary vectors $\mathbf{x}$ that satisfy $\mathbf{Cx} \leq \mathbf{d}$, that is,*

$$\mathcal{S} = \{S \subseteq \{1,\ldots,n\} \,|\, S = S(\mathbf{x}) \text{ for some } \mathbf{x} \in \{0,1\}^n \text{ such that } \mathbf{Cx} \leq \mathbf{d}\}.$$

This assumption assumes that the set $\mathcal{S}$ can be written in terms of $\mathbf{x}$ through the system of inequalities $\mathbf{Cx} \leq \mathbf{d}$. This is a relatively mild assumption, as one can model a wide range of business requirements through such a system of inequalities; we discuss some examples at the end of this section after stating our formulation.

The problem, in its most basic form, can then be formulated as follows.

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{k=1}^{K}\sum_{i=1}^{n} \pi_i \cdot \lambda^k \cdot y_i^k && \text{(2a)}\\
\text{subject to} \quad & \sum_{i=0}^{n} y_i^k = 1, \quad \forall\, k \in \{1,\ldots,K\}, && \text{(2b)}\\
& y_i^k \leq x_i, \quad \forall\, k \in \{1,\ldots,K\},\ i \in \{1,\ldots,n\}, && \text{(2c)}\\
& \sum_{j:\sigma^k(j)>\sigma^k(i)} y_j^k \leq 1 - x_i, \quad \forall\, k \in \{1,\ldots,K\},\ i \in \{1,\ldots,n\}, && \text{(2d)}\\
& \sum_{j:\sigma^k(j)>\sigma^k(0)} y_j^k = 0, \quad \forall\, k \in \{1,\ldots,K\}, && \text{(2e)}\\
& \mathbf{Cx} \leq \mathbf{d}, && \text{(2f)}\\
& x_i \in \{0,1\}, \quad \forall\, i \in \{1,\ldots,n\}, && \text{(2g)}\\
& y_i^k \geq 0, \quad \forall\, k \in \{1,\ldots,K\},\ i \in \{0,1,\ldots,n\}. && \text{(2h)}
\end{aligned}$$

In order of appearance, the constraints have the following meaning. Constraint (2b) ensures that under each ranking, exactly one choice is made. Constraint (2c) ensures that under ranking k, product i can only be chosen if product i is included in the product line. Constraint (2d) ensures that, if product i is included in the product line, then none of the options that are less preferred to i under ranking σ^k may be chosen under ranking k (the y_j^k variables for all j that are less preferred are forced to zero). Constraint (2e) is a similar constraint, but pertaining to the no-purchase option: those options that are less preferred to the no-purchase option 0 may not be selected and their y_j^k values are forced to zero. Constraint (2f) ensures that the product line encoded by $\mathbf{x}$ is selected from the set of permissible product lines $\mathcal{S}$ and follows the firm's business requirements. The penultimate constraint ensures that the x_i's are binary. The last constraint ensures that each y_i^k is nonnegative. The objective function is the expected per-customer profit of the product line encoded by the x_i variables. Note that the marginal profit π_i of each product i is known and fixed; it is not a decision variable.

It is worth commenting on several important aspects of this formulation. First, note that the formulation does not require the y_i^k variables to be binary. This is because for fixed binary values

of x_i, constraints (2b)–(2e) ensure that the y_i^k values are forced to the correct binary values. Thus, as a result, the formulation has only n binary variables (the x_i variables).

Second, constraint (2f) allows us to enforce a variety of different constraints on the product line. We provide some examples below.

1. **Lower and upper bounds on the size of the product line.** Due to resource constraints, the firm may wish the number of products offered to be between certain upper and lower bounds. If U and L are upper and lower bounds on the number of products in the product line, then this requirement can be algebraically modeled as

$$L \leq \sum_{i=1}^{n} x_i \leq U.$$

In matrix-vector form, letting $\mathbf{1}$ denote a column n-vector of ones, we can write this as

$$\begin{bmatrix} \mathbf{1}^T \\ -\mathbf{1}^T \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} U \\ -L \end{bmatrix}.$$

2. **Precedence constraints.** The firm may require that if a product i is included, then a product i' must be included. This can be modeled through the inequality $x_i \leq x_{i'}$. Letting $\mathbf{e}_j$ be the jth unit column n-vector, we can write this constraint in matrix-vector form as

$$\left[\mathbf{e}_i^T - \mathbf{e}_{i'}^T\right] \mathbf{x} \leq [\,0\,]$$

We remark that if there are no requirements constraining the product line (i.e., $\mathcal{S} = \mathcal{P}(\{1,\ldots,n\})$), then we can model this using the constraint $\mathbf{C}\mathbf{x} \leq \mathbf{d}$ by setting $\mathbf{C} = [\mathbf{0}^T]$ and $\mathbf{d} = [\,0\,]$, where $\mathbf{0}$ is a column n-vector of zeros.

3.3. Alternate formulations

In this section, we describe three other MIO formulations for modeling the first-choice PLD problem. The first formulation we will consider is the product line design problem formulation that was studied in Belloni et al. (2008):

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{k=1}^{K} \sum_{i=1}^{n} \pi_i \cdot \lambda^k \cdot y_i^k && \text{(3a)}\\
\text{subject to} \quad & \sum_{i=0}^{n} y_i^k = 1, \quad \forall\ k \in \{1,\ldots,K\}, && \text{(3b)}\\
& y_i^k \leq x_i, \quad \forall\ k \in \{1,\ldots,K\},\ i \in \{1,\ldots,n\}, && \text{(3c)}\\
& y_j^k \leq 1 - x_i, \quad \forall\ k \in \{1,\ldots,K\},\ i,j \in \{1,\ldots,n\},\ \sigma^k(j) > \sigma^k(i), && \text{(3d)}\\
& y_j^k = 0, \quad \forall\ k \in \{1,\ldots,K\}, j \in \{1,\ldots,n\},\ \sigma^k(j) > \sigma^k(0), && \text{(3e)}\\
& \mathbf{C}\mathbf{x} \leq \mathbf{d}, && \text{(3f)}\\
& x_i \in \{0,1\}, \quad \forall\ i \in \{1,\ldots,n\}, && \text{(3g)}\\
& y_i^k \geq 0, \quad \forall\ k \in \{1,\ldots,K\},\ i \in \{0,1,\ldots,n\}. && \text{(3h)}
\end{aligned}$$

This formulation is similar to our problem (2); the key difference is that for a fixed option ($i \in \{1,\ldots,\}$ or the no-purchase option), the preference constraints (2d) and (2e) in our formulation aggregate all lower ranked options j, whereas in (3) each lower ranked j is handled in a disaggregated way, through a separate constraint.

The two other formulations we will study involve transforming the ranking preferences into utilities. In particular, for each customer type $k \in \{1,\ldots,K\}$ and each option $i \in \{0,\ldots,n\}$, let u_i^k denote the utility of option i to customer type k. The u_i^k values can take any values that are consistent with the assumptions below.

ASSUMPTION 2. *Each u_i^k is nonnegative.*

ASSUMPTION 3. *For a given k, the u_i^k values satisfy $u_i^k > u_j^k$ whenever $\sigma^k(i) < \sigma^k(j)$; stated differently, option i has higher utility than option j if and only if i is preferred to j (the rank of i is lower than the rank of j).*

Assumption 3 implies that each customer type k chooses the option that yields them the greatest utility, which is equivalent to customers choosing the option that has the lowest rank according to σ^k. We will use $L^k = \min_{0\leq j\leq n} u_j^k$ and $U^k = \max_{0\leq j\leq n} u_j^k$ to denote the minimum and maximum values, respectively, of the option utilities of each customer type k.

We now consider our first utility-based formulation, which is given below.

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{k=1}^{K}\sum_{i=1}^{n} \pi_i \cdot \lambda^k \cdot y_i^k && \text{(4a)}\\
\text{subject to} \quad & \sum_{i=0}^{n} y_i^k = 1, \quad \forall\ k \in \{1,\dots,K\}, && \text{(4b)}\\
& y_i^k \leq x_i, \quad \forall\ k \in \{1,\dots,K\},\ i \in \{1,\dots,n\}, && \text{(4c)}\\
& \sum_{j=0}^{n} u_j^k y_j^k \geq (u_i^k - L^k)x_i + L^k, \quad \forall\ k \in \{1,\dots,K\},\ i \in \{1,\dots,n\}, && \text{(4d)}\\
& \sum_{j=0}^{n} u_j^k y_j^k \geq u_0^k, \quad \forall\ k \in \{1,\dots,K\}, && \text{(4e)}\\
& \mathbf{C}\mathbf{x} \leq \mathbf{d}, && \text{(4f)}\\
& x_i \in \{0,1\}, \quad \forall\ i \in \{1,\dots,n\}, && \text{(4g)}\\
& y_i^k \geq 0, \quad \forall\ k \in \{1,\dots,K\},\ i \in \{0,1,\dots,n\}. && \text{(4h)}
\end{aligned}$$

In this formulation, preferences are represented through constraints (4d) and (4e). The left-hand side of constraint (4d) represents the utility of the option chosen by customer type k; the right-hand side is equal to the utility of product i, if product i is selected for the product line, and L^k otherwise, in which case the constraint is vacuous. In words, the constraint requires that the choice of customer type k indeed provides the greatest utility among those products in the product line. Constraint (4e) is a similar constraint for the no-purchase option. All other constraints are the same as in problem (2). Note that, like problems (2) and (3), the y_i^k variables need not be defined as binary, as they will automatically be forced to their correct binary values. To the best of our knowledge, this formulation has not been proposed in the literature; the justification for why we are considering this formulation will be deferred until after we present the final formulation.

We now consider our second utility-based formulation, which was first proposed in McBride and Zufryden (1988).

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{k=1}^{K}\sum_{i=1}^{n} \pi_i \cdot \lambda^k \cdot y_i^k && \text{(5a)}\\
\text{subject to} \quad & \sum_{i=0}^{n} y_i^k = 1, \quad \forall\ k \in \{1,\dots,K\}, && \text{(5b)}\\
& y_i^k \leq x_i, \quad \forall\ k \in \{1,\dots,K\},\ i \in \{1,\dots,n\}, && \text{(5c)}\\
& u_i^k x_i \geq u_j^k x_j - U^k(1-y_i^k), \quad \forall\ k \in \{1,\dots,K\},\ i,j \in \{1,\dots,n\},\ i \neq j, && \text{(5d)}\\
& u_i^k x_i \geq u_0^k - U^k(1-y_i^k), \quad \forall\ k \in \{1,\dots,K\},\ i \in \{1,\dots,n\}, && \text{(5e)}\\
& u_0^k \geq u_j^k x_j - U^k(1-y_0^k), \quad \forall\ k \in \{1,\dots,K\},\ j \in \{1,\dots,n\}, && \text{(5f)}
\end{aligned}$$

$$\mathbf{Cx} \leq \mathbf{d}, \tag{5g}$$
$$x_i \in \{0,1\}, \quad \forall\, i \in \{1,\ldots,n\}, \tag{5h}$$
$$y_i^k \in \{0,1\}, \quad \forall\, k \in \{1,\ldots,K\},\ i \in \{0,1,\ldots,n\}. \tag{5i}$$

In this formulation, preferences are represented through constraints (5d) - (5f). To understand constraint (5d), suppose that $y_i^k = 1$ and thus, $x_i = 1$. If product j is in the product line, then $x_j = 1$, and the constraint is satisfied only if $u_i^k \geq u_j^k$, i.e., the utility of i, which is chosen by customer k, is more than that of j. On the other hand, if product j is not in the product line, then $x_j = 0$; the right hand side is then simply 0, and the constraint is vacuous (recall that all utilities are assumed to be nonnegative). This constraint thus requires that whatever option is selected (i.e., for which the corresponding $y_i^k = 1$) is indeed the one that maximizes the utility over the products in the product line. Constraints (5e) and (5f) are similar constraints for handling the no-purchase option. Note that unlike problems (2), (3) and (4), the y_i^k variables must be binary in this formulation.

3.4. Theoretical comparison of formulations

In the previous section, we defined several alternate formulations of the first-choice PLD problem. This leads us to a natural question: how do these formulations compare? Is one formulation better than the others? The answer to this question comes from considering the relaxations of these formulations. We briefly review some central concepts in mixed-integer optimization that will motivate our theoretical results; the interested reader is referred to Bertsimas and Weismantel (2005) for more details.

The LO relaxation of a MIO problem is obtained by replacing every constraint of the form $x_i \in \{0,1\}$ with a constraint of the form $0 \leq x_i \leq 1$; in words, we relax the binary variables so that they are continuous and may take any real value between 0 and 1. The resulting problem is a LO problem, as opposed to being a MIO problem. The feasible region of the relaxation of a MIO model always contains the feasible region of the MIO model, and the optimal value of the relaxation of an MIO model with a maximization objective is always an upper bound on the true optimal value of the MIO model.

Typically, for a given optimization problem (such as the first-choice PLD problem), there may be several MIO models that are valid formulations of the problem, but that differ in their relaxations. The best possible MIO formulation is one where the relaxation exactly coincides with the convex hull of the integer feasible solutions of the MIO formulation. Such a formulation is called *ideal* or *integral*, because all extreme points of the relaxation correspond to integer feasible solutions, so one can solve the MIO formulation by simply solving its relaxation. Integral formulations are often impractical, however, because they may be of an exponential size.

In the absence of an integral formulation, one may hope to find a formulation whose relaxation is as close as possible to the convex hull of the integer feasible solutions. A *strong* or *tight* MIO formulation is one where the feasible region of the relaxation is close to the convex hull of the integer solutions; a *weak* or *loose* formulation is one where the feasible region is a loose approximation to the convex hull of the integer solutions. A strong formulation is desirable because typically, strong formulations can be solved more quickly than loose formulations. Given two formulations A and B, where $\mathcal{F}_A$ and $\mathcal{F}_B$ are the feasible regions of their relaxations, we say that A is at least as strong as B if $\mathcal{F}_A \subseteq \mathcal{F}_B$. By comparing two relaxations with respect to subset containment, we can determine which formulation is stronger.

From this perspective of formulation strength, let us now provide the justification for considering problem (4). Let $\mathcal{F}_{Utility}$ and $\mathcal{F}_{MZ}$ denote the feasible regions of the LO relaxations of problems (4) and (5), respectively. The justification for considering problem (4) comes from the following proposition, which states that problem (4) is at least as strong as problem (5) (see Section EC.1.1 of the electronic companion for the proof).

PROPOSITION 1. *For any choice of u_i^k satisfying Assumptions 2 and 3, $\mathcal{F}_{Utility} \subseteq \mathcal{F}_{MZ}$.*

We now arrive to the main result of this section. Let $\mathcal{F}_{BM}$ and $\mathcal{F}_{BFSS}$ denote the feasible regions of the relaxations of our problem (2) and problem (3), respectively. Theorem 1 below (see Section EC.1.2 of the ecompanion for the proof) compares the LO relaxations of problems (3), (4) and (5).

THEOREM 1. *For any choice of u_i^k satisfying assumptions 2 and 3, we have:*
(a) $\mathcal{F}_{BM} \subseteq \mathcal{F}_{BFSS}$;
(b) $\mathcal{F}_{BM} \subseteq \mathcal{F}_{Utility}$;
(c) $\mathcal{F}_{BM} \subseteq \mathcal{F}_{MZ}$.

Theorem 1 is significant because it asserts that out of the four formulations – problems (2), (3), (4) and (5) – our formulation (2) is the strongest. We will see in Section 5.2 that our formulation can be solved in significantly less time than the other three formulations, as one would predict from this theorem.

Let Z_{BM}^*, Z_{BFSS}^*, $Z_{Utility}^*$ and Z_{MZ}^* denote the optimal values of the LO relaxation of problems (2), (3), (4) and (5), respectively. Since all four formulations share the same objective function in terms of $(\mathbf{x}, \mathbf{y})$, an immediate corollary of the above is that the LO relaxation bound of our formulation is tighter than the others.

COROLLARY 1. *For any choice of u_i^k satisfying assumptions 2 and 3, we have:*
(a) $Z_{BM}^* \leq Z_{BFSS}^*$;
(b) $Z_{BM}^* \leq Z_{Utility}^*$; *and*
(c) $Z_{BM}^* \leq Z_{MZ}^*$.

We will see in Section 5.2 that the difference between our relaxation bound Z_{BM}^* and the others can be substantial.

We conclude this section by stating two additional results that relate the other three formulations to each other. We have already seen one (Proposition 1). The first additional result states that problem (3) of Belloni et al. (2008) is stronger than problem (5) of McBride and Zufryden (1988).

PROPOSITION 2. *For any choice of u_i^k satisfying assumptions 2 and 3, $\mathcal{F}_{BFSS} \subseteq \mathcal{F}_{MZ}$.*

We provide the proof in Section EC.1.3 of the ecompanion. Proposition 2, together with Theorem 1 and Proposition 1, implies that problem (5) is the weakest formulation of the problem. The second additional result states that in general, the feasible regions of the relaxations of problems (3) and (4) are not contained in each other.

PROPOSITION 3. *In general, $\mathcal{F}_{BFSS} \not\subseteq \mathcal{F}_{Utility}$ and $\mathcal{F}_{Utility} \not\subseteq \mathcal{F}_{BFSS}$.*

The proof of both non-containments (see Section EC.1.4 of the ecompanion) follows through a simple instance with $K = 1$.

Overall, the results above provide us with the following unified perspective on all four formulations:

1. Problem (2) (this paper) is the strongest formulation.
2. Problem (5) of McBride and Zufryden (1988) is the weakest formulation.
3. Problem (3) of Belloni et al. (2008) and the utility-based problem (4) are each weaker than (2) and stronger than (5), but neither is stronger than the other.

3.5. Theoretical comparison when $K = 1$

It is also interesting to consider under what conditions our formulation (2) is integral, that is, every extreme point solution of the LO relaxation is an integer feasible solution the integer problem (2); thus, by solving the LO relaxation, we we solve the integer problem. In general, for $K \geq 2$, our formulation is not integral, as we will verify in our numerical results in Section 5.1. However, for the special case of $K = 1$ (i.e., only one customer type), we have the following result.

PROPOSITION 4. *When $K = 1$ and constraint* (2f) *is removed, $\mathcal{F}_{BM}$ is integral, i.e., every extreme point $(\mathbf{x}, \mathbf{y})$ of $\mathcal{F}_{BM}$ satisfies $\mathbf{x} \in \{0,1\}^n$.*

The proof of Proposition 4 (see Section EC.1.5 of the ecompanion) follows by showing that the constraint matrix which defines the set $\mathcal{F}_{BM}$ is totally unimodular.

In contrast, the other formulations that we have considered above do not enjoy this property, as established by the following proposition.

PROPOSITION 5. *When $K = 1$, $\mathcal{F}_{BFSS}$, $\mathcal{F}_{Utility}$ and $\mathcal{F}_{MZ}$ are not integral in general.*

We prove the above result by providing specific instances (i.e., specific choices of n and σ) for which the feasible region is not integral. The proof is provided in Section EC.1.6 of the ecompanion.

Propositions 4 and 5 provide an important complement to Theorem 1 that underscores the value of formulation (2) relative to existing formulations: in the simplest case of only a single customer type, problem (2) is integral, whereas the alternate formulations are not. We shall see later in our numerical experiments that even when $K \gg 1$, the integrality gap – the relative difference between the optimal value of the integer problem and the LO relaxation of problem (2) – can be very low.

4. Large-scale solution via Benders decomposition

While problem (2) is a modestly-sized formulation of the problem and its relaxation is tighter than that of alternative formulations, it can still be challenging to solve directly when the number of products n or the number of rankings K is large. At the same time, formulation (2) is structured in a special way that allows us to apply the method of Benders decomposition. Before presenting our results on adapting Benders decomposition to our product line formulation, we provide a brief overview of Benders decomposition. The interested reader is referred to Chapter 6.5 of Bertsimas and Tsitsiklis (1997) or Chapter II.3.7 of Nemhauser and Wolsey (1988) for the basic theory of the method, and to the paper of Rahmaniani et al. (2017) for a review of recent theoretical advances and applications.

Benders decomposition is a technique for solving large-scale linear and mixed-integer optimization problems where one can divide the decision variables into two sets of variables, $\mathbf{x}$ and $\mathbf{y}$, where the $\mathbf{y}$ variables are continuous. The idea of Benders decomposition is to use LO duality theory to eliminate the $\mathbf{y}$ variables and to represent their effect in the formulation with a family of constraints in terms of the $\mathbf{x}$ variables. The resulting formulation in terms of the $\mathbf{x}$ variables is called the *master* problem. The family of constraints obtained from eliminating $\mathbf{y}$ is usually too large to explicitly enumerate in the formulation. Thus, to solve the master problem, one must typically iteratively generate the constraints. These constraints are generated by solving the *subproblem*, which is the problem of optimizing the $\mathbf{y}$ variables in the original problem where the $\mathbf{x}$ variables are fixed to their current values. This approach is termed a decomposition because the $\mathbf{x}$ and the $\mathbf{y}$ variables are effectively decoupled.

Benders decomposition is particularly attractive for at least three reasons. First, it can be faster than direct solution on large problems. Second, some formulations are simply too large to be explicitly represented in computer memory, and a large scale optimization approach like Benders decomposition is the only option in such situations. Indeed, Benders decomposition has had significant practical success in solving large-scale optimization problems arising in areas such as multicommodity distribution system design (Geoffrion and Graves 1974), airline scheduling (Cordeau et al. 2001) and hub location (Contreras et al. 2011). Third, it is often the case that the subproblem has a special structure that allows it to be solved easily, either in closed form or through a specialized algorithm. For example, in Contreras et al. (2011), it is shown that the subproblem is a collection of independent semi-assignment problems, and that the dual solution can be obtained in closed form by exploiting complementary slackness. By exploiting the structure of the subproblem, it is possible to generate constraints for the master problem in a computationally efficient manner.

Benders decomposition applies naturally to the first-choice PLD problem because the variables in our model (2) can be divided in two: the $\mathbf{x}$ variables, which specify which products are included

in the product line, and the $\mathbf{y}^k$ variables, which specify how customers of type k choose from the selected product line. One can thus specify the master problem as that of selecting the product line $\mathbf{x}$, and a collection of K subproblems, one to determine the choice $\mathbf{y}^k$ of each customer type k. To the best of our knowledge, Benders decomposition has not been previously applied in PLD.

This section is organized as follows. In Section 4.1, we first present a Benders reformulation of our product line problem (2) into a master problem and a collection of subproblems, and show that the subproblems can be solved in closed form when the current master solution is binary. In Section 4.2, we then show that the subproblems can be solved efficiently when the current master solution is fractional, through a greedy algorithm. Finally, in Section 4.3, we summarize the complete method.

4.1. Benders reformulation of the integer problem

We begin by re-writing problem (2) as

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{t}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda^k t_k && \text{(6a)}\\
\text{subject to} \quad & t_k \leq G_k(\mathbf{x}), \quad \forall\, k \in \{1,\ldots,K\}, && \text{(6b)}\\
& \mathbf{C}\mathbf{x} \leq \mathbf{d}, && \text{(6c)}\\
& x_i \in \{0,1\}, \quad \forall\, i \in \{1,\ldots,n\}, && \text{(6d)}
\end{aligned}$$

where $G_k(\mathbf{x})$ is defined as the optimal value of the following LO problem:

$$\begin{aligned}
G_k(\mathbf{x}) = \underset{\mathbf{y}^k}{\text{maximize}} \quad & \sum_{i=1}^{n} \pi_i \cdot y_i^k && \text{(7a)}\\
\text{subject to} \quad & \sum_{i=0}^{n} y_i^k = 1, && \text{(7b)}\\
& y_i^k \leq x_i, \quad \forall\, i \in \{1,\ldots,n\}, && \text{(7c)}\\
& \sum_{j:\sigma^k(j)>\sigma^k(i)} y_j^k \leq 1 - x_i, \quad \forall\, i \in \{1,\ldots,n\}, && \text{(7d)}\\
& \sum_{j:\sigma^k(j)>\sigma^k(0)} y_j^k = 0, && \text{(7e)}\\
& y_i^k \geq 0, \quad \forall\, i \in \{0,1,\ldots,n\}. && \text{(7f)}
\end{aligned}$$

As alluded to in Section 3.2, when $\mathbf{x}$ is binary, the constraints in problem (7) automatically force the y_j^k variables to their correct binary values, reflecting the choice of ranking k for the product line encoded by $\mathbf{x}$. This result is formalized as Proposition 6.

PROPOSITION 6. *Let $k \in \{1,\ldots,K\}$, $\mathbf{x} \in \{0,1\}^n$ and $S = \{i \in \{1,\ldots,n\} \,|\, x_i = 1\}$. An optimal solution of problem (7) for ranking k and product line encoded by $\mathbf{x}$ is given by*

$$y_i^k = \begin{cases} 1, \text{ if } i = \arg\min_{j \in S \cup \{0\}} \sigma^k(j), \\ 0, \text{ otherwise.} \end{cases} \tag{8}$$

The proof of Proposition 6 (see Section EC.1.7 of the ecompanion) follows by showing that the $\mathbf{y}$ solution prescribed by the proposition constitutes the *only* feasible solution for problem (7). When $\mathbf{x}$ is binary, Proposition 6 allows us to assert that problem (7) is feasible and bounded. Therefore,

by LO strong duality, the optimal value of problem (7) is equal to the optimal value of its dual problem:

$$
\begin{aligned}
G_k(\mathbf{x}) = \underset{\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k}{\text{minimize}} \quad & \gamma^k + \sum_{j=1}^n \alpha_j^k x_j + \sum_{i=1}^n \beta_i^k \cdot (1 - x_i) && \text{(9a)} \\
\text{subject to} \quad & \gamma^k + \alpha_j^k + \sum_{i\,:\,\sigma^k(i) < \sigma^k(j)} \beta_i^k \geq \pi_j, \quad \forall\ j \in \{1, \ldots, n\}, && \text{(9b)} \\
& \gamma^k + \sum_{i\,:\,\sigma^k(i) < \sigma^k(0)} \beta_i^k \geq 0, && \text{(9c)} \\
& \boldsymbol{\alpha}^k, \boldsymbol{\beta}^k \geq \mathbf{0}. && \text{(9d)}
\end{aligned}
$$

Letting A_k indicate the feasible set of $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ for the dual subproblem (9) of ranking k, we can then re-write problem (6) as

$$
\begin{aligned}
\underset{\mathbf{x}, \mathbf{t}}{\text{maximize}} \quad & \sum_{k=1}^K \lambda^k t_k && \text{(10a)} \\
\text{subject to} \quad & t_k \leq \gamma^k + \sum_{i=1}^n \alpha_i^k \cdot x_i + \sum_{i=1}^n \beta_i^k \cdot (1 - x_i), \quad \forall\ k \in \{1, \ldots, K\},\ (\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k) \in A_k, && \text{(10b)} \\
& \mathbf{C}\mathbf{x} \leq \mathbf{d}, && \text{(10c)} \\
& x_i \in \{0, 1\}, \quad \forall\ i \in \{1, \ldots, n\}. && \text{(10d)}
\end{aligned}
$$

The above problem is a problem that is amenable to constraint generation. For a given integer feasible solution $(\mathbf{x}, \mathbf{t})$, we check for each ranking k whether there is a $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ for which the constraints are violated. If so, we add the constraint, and solve again; otherwise, we conclude that the current integer feasible solution is optimal.

To determine whether there is a $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ for which constraint (10b) is violated, we have to solve problem (9) and compare its optimal value to t_k. It turns out that, like the primal problem (7), the dual subproblem (9) can also be solved in a very simple way when $\mathbf{x}$ is binary.

PROPOSITION 7. *Let* $k \in \{1, \ldots, K\}$, $\mathbf{x} \in \{0, 1\}^n$ *and* $S = \{i \in \{1, \ldots, n\} \,|\, x_i = 1\}$. *Let* $i^* = \arg\min_{j \in S \cup \{0\}} \sigma^k(j)$ *and let* π^* *be defined as*

$$
\pi^* = \begin{cases} \pi_{i^*}, & \text{if } i^* \neq 0, \\ 0, & \text{if } i^* = 0. \end{cases}
$$

An optimal solution of problem (9) *for ranking* k *and product line encoded by* $\mathbf{x}$ *is given by*

$$
\begin{aligned}
\alpha_i^k &= \max\left\{ \pi_i - \gamma^k - \sum_{j\,:\,\sigma^k(j) < \sigma^k(i)} \beta_j^k,\ 0 \right\}, && \text{(11a)} \\
\beta_i^k &= \begin{cases} \max_{i' \in S} \pi_{i'} - \pi^*, & \text{if } i = i^*, \\ 0, & \text{otherwise}, \end{cases} && \text{(11b)} \\
\gamma^k &= \pi^*. && \text{(11c)}
\end{aligned}
$$

The proof of Proposition 7, which can be found in Section EC.1.8 of the electronic companion, follows by checking that the proposed solution is feasible and attains the same objective value as the optimal solution of the primal problem.

The significance of Proposition 7 is that it furnishes us with a simple procedure for testing whether an integer solution $(\mathbf{x}, \mathbf{t})$ to problem (10) violates any constraints or not. Namely, for

each k, we generate the $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ solution prescribed by Proposition 7 and compare its objective value in problem (9) to t_k. If t_k is equal or lower than this objective value, then the kth family of constraints is not violated; otherwise, if t_k is greater than this objective value, then we have identified a violated constraint and we add it to the formulation.

In our implementation of Benders decomposition for the integer problem, we add constraints to problem (10) in the so-called "lazy" fashion. This means that one solves problem (10) using a branch-and-bound solver, and checks for violated constraints at each integer solution generated in the branch-and-bound tree; if a violated constraint is found, the constraint is added to every node in the branch-and-bound tree, the integer solution is discarded and the node is re-solved. This differs from the "classical" Benders decomposition approach. In the classical approach, one solves the restricted master of problem (10) to full optimality with the current set of constraints, adds any violated constraints, and solves the problem again with the new set of constraints, repeating until one can no longer find violated constraints. The lazy approach is advantageous because one does not repeatedly solve the problem from scratch. Instead, one only solves the problem once, and constraints are only added as they are needed in the branch-and-bound tree.

4.2. Benders reformulation of the relaxed problem

We can use similar reasoning to obtain a Benders reformulation of the LO relaxation of problem (2). The resulting formulation is given below.

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{t}}{\text{maximize}} \quad & \sum_{k=1}^{K} \lambda^k t_k && \text{(12a)}\\
\text{subject to} \quad & t_k \leq \gamma^k + \sum_{i=1}^{n} \alpha_i^k \cdot x_i + \sum_{i=1}^{n} \beta_i^k \cdot (1 - x_i), \quad \forall\, k \in \{1,\ldots,K\},\ (\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k) \in A_k, && \text{(12b)}\\
& \mathbf{C}\mathbf{x} \leq \mathbf{d}, && \text{(12c)}\\
& 0 \leq x_i \leq 1, \quad \forall\, i \in \{1,\ldots,n\}. && \text{(12d)}
\end{aligned}$$

Problem (12), like the integer problem (10), can also be solved using constraint generation; we formally describe our constraint generation procedure in Section EC.2.2 of the electronic companion. This problem is valuable to consider, for two reasons. First, solving this problem will yield a valid upper bound on the true integer optimal value; this gives us a way to measure the suboptimality of a candidate integer solution. Second, when we solve problem (12), we can use the constraints that have been generated to warm-start the solution process of the integer problem (10).

One may wonder if it is possible to obtain Benders cuts efficiently for fractional solutions $\mathbf{x}$, so as to solve problem (12) and potentially warm-start the integer problem (10) with a set of high-quality cuts. Surprisingly, it turns out that this is the case; one can solve problems (7) and (9) using a two-phase greedy algorithm that first constructs a primal solution, followed by a dual solution. We use this algorithm to generate cuts in the LO relaxation phase of our overall Benders approach.

For ease of exposition, let us redefine $\mathbf{x}$ as $\mathbf{x} \in [0,1]^{n+1}$, where $\mathbf{x} = (x_0, x_1, \ldots, x_n)$; for convenience, we take $x_0 = 1$, and we also let $\pi_0 = 0$. Let σ be the ranking; we drop the k superscript used to indicate the customer type to simplify the exposition. With these definitions, the primal and dual subproblems are given below.

$$\begin{aligned}
\text{(Primal)} \quad \underset{\mathbf{y}}{\text{maximize}} \quad & \sum_{i=0}^{n} \pi_i \cdot y_i && \text{(13a)}\\
\text{subject to} \quad & \sum_{i=0}^{n} y_i = 1, && \text{(13b)}\\
& y_i \leq x_i, \quad \forall\, i \in \{0,1,\ldots,n\}, && \text{(13c)}
\end{aligned}$$

$$\sum_{j:\sigma(j)>\sigma(i)} y_j \leq 1 - x_i, \quad \forall\, i \in \{0,1,\ldots,n\}, \tag{13d}$$

$$y_i \geq 0, \quad \forall\, i \in \{0,1,\ldots,n\}. \tag{13e}$$

$$\text{(Dual)} \quad \underset{\boldsymbol{\alpha},\boldsymbol{\beta},\gamma}{\text{minimize}} \quad \gamma + \sum_{j=0}^{n} \alpha_j x_j + \sum_{i=0}^{n} \beta_i \cdot (1 - x_i) \tag{14a}$$

$$\text{subject to} \quad \gamma + \alpha_j + \sum_{i\,:\,\sigma(i)<\sigma(j)} \beta_i \geq \pi_j, \quad \forall\, j \in \{0,1,\ldots,n\}, \tag{14b}$$

$$\boldsymbol{\alpha}, \boldsymbol{\beta} \geq \mathbf{0}. \tag{14c}$$

The overall algorithm that we use to solve these problems has a primal and a dual phase. Algorithm 1 provides the pseudocode for the primal phase, while Algorithm 2 provides the pseudocode of the dual phase.

Algorithm 1 Primal phase of greedy algorithm for solving problems (13) and (14)

Require: Ordering $\tau : \{0,1,\ldots,n\} \to \{0,1,\ldots,n\}$ such that $\pi_{\tau(0)} \geq \pi_{\tau(1)} \geq \cdots \geq \pi_{\tau(n)}$.
  Set $y_j = 0$ for all $j \in \{0,1,\ldots n\}$.
  Set $C = \tau(n)$.
  Set $A \leftarrow \emptyset$, $B \leftarrow \emptyset$, $B_{main} \leftarrow \emptyset$.
  for $s = 0,1,\ldots,n$ **do**
    Set $q_1 \leftarrow x_{\tau(s)}$.
    Set $q_2 \leftarrow \min_{i:\sigma(i)<\sigma(\tau(s))} \left\{1 - x_i - \sum_{j:\sigma(j)>\sigma(i)} y_j\right\}$.
    Set $q_3 \leftarrow 1 - \sum_{j=0}^{n} y_j$.
    Set $q^* \leftarrow \min\{q_1, q_2, q_3\}$.
    Set $y_{\tau(s)} \leftarrow q^*$.
    if $q^* = q_3$ **then**
      {C event}
      Set $C \leftarrow \tau(s)$.
      break
    else
      if $q^* = q_2$ **then**
        {B event}
        Set $i^* = \arg\min_{i:\sigma(i)<\sigma(\tau(s))} \left\{1 - x_i - \sum_{j:\sigma(j)>\sigma(i)} y_j\right\}$.
        if $i^* \notin B_{main}$ **then**
          Set $B \leftarrow B \cup (i^*, \tau(s))$.
          Set $B_{main} \leftarrow B_{main} \cup \{i^*\}$.
        end if
      else
        {A event}
        Set $A \leftarrow A \cup \{\tau(s)\}$.
      end if
    end if
  end for

Before we present our formal result on the correctness of these two algorithms, it is worthwhile to give a high level overview of each algorithm. Algorithm 1 is the primal phase of the overall

Algorithm 2 Dual phase of greedy algorithm for solving problems (13) and (14)

Require: Sets A, B, B_{main}, integer C from primal phase.
  Initialize $\alpha_i = 0$ for all $i \in \{0, 1, \ldots, n\}$; $\beta_i = 0$ for all $i \in \{0, 1, \ldots, n\}$.
  Set $\gamma \leftarrow \pi_C$.
  Define $f : B_{main} \to \{0, 1, \ldots, n\}$ such that $(i, f(i)) \in B$.
  Sort $B_{main} = \{i_1, \ldots, i_{|B_{main}|}\}$ so that $\sigma(i_1) < \sigma(i_2) < \cdots < \sigma(i_{|B_{main}|})$.
  for $t = 1, 2, \ldots, |B_{main}|$ **do**
    Set $\beta_{i_t} = \pi_{f(i_t)} - \gamma - \sum_{t'=1}^{t-1} \beta_{i_{t'}}$.
  end for
  for $i \in A$ **do**
    Set $\alpha_i = \pi_i - \gamma - \sum_{j:\sigma(j)<\sigma(i)} \beta_j$.
  end for

algorithm. In this phase, we order the options according to their marginal profits, using the function τ – i.e.,

$$\pi_{\tau(0)} \geq \pi_{\tau(1)} \geq \cdots \geq \pi_{\tau(n)}.$$

By convention, we set $\tau(n) = 0$ (the no-purchase option). We then proceed through the options, from the highest profit option (this is $\tau(0)$) to the lowest profit option (this is $\tau(n)$, which is the no-purchase option). At each stage, we set $y_{\tau(s)}$ to the highest possible value it can be so that none of the constraints of problem (13) are violated. After setting the value of $y_{\tau(s)}$, we effectively record which constraint was tight using the sets A, B_{main}, B and the integer C. This involves several checks:

- We check first if the $\mathbf{y}$ solution sums to 1. If it does, we say that a C event has occurred; we set the integer $C = \tau(s)$. We also terminate the primal phase, because we know there is no need to check any of the remaining options $\tau(s+1), \ldots, \tau(n)$ (they cannot be anything other than zero).
- If $\mathbf{y}$ does not sum to 1, we then check if one of the preference constraints (13d) has become binding. If so, we say that a B event has occurred; we record which constraint became binding (this is i^*) and what option was being checked when it occurred (this is $\tau(s)$). We add the pair $(i^*, \tau(s))$ to B, and we add i^* to B_{main}; we only do this if i^* is not in B_{main} already. Note that when the arg min defining i^* is not unique, we take the one that has the lowest value of σ.
- If in the above check, one of the constraints (13d) did not become binding, then it must be that one of the constraints (13c) became binding. We say that an A event has occurred, and we add the option $\tau(s)$ that we checked to the set A.

Upon termination, we have a primal solution $\mathbf{y}$, sets A, B, B_{main}, and integer C. We then run the dual phase (Algorithm 2). In the dual phase, we first set γ, using C (γ is the dual variable corresponding to the unit sum constraint); we then set $\boldsymbol{\beta}$ using B, B_{main} and γ (the set B_{main} contains those indices for which β_i can be nonnegative); and finally, we set $\boldsymbol{\alpha}$ using A, $\boldsymbol{\beta}$ and γ. In Algorithm 1, the set B keeps track of the pairs $(i^*, \tau(s))$ that were encountered in Algorithm 1; i^* indicates the preference inequality that became binding, while $\tau(s)$ is the coordinate of $\mathbf{y}$ that we set in Algorithm 1 that made that inequality become binding. The function f encodes the pairs in B; if i^* is in B_{main}, then i^* is the index of a preference inequality that became binding, and $f(i^*)$ is the option that was being modified in Algorithm 1. We use the function f to determine what the β_j values should be.

We now establish that Algorithms 1 and 2 are indeed correct; they produce primal feasible and dual feasible solutions, respectively, and these solutions are optimal.

THEOREM 2. *Let* $\mathbf{y}$ *be the solution produced by Algorithm 1, and* $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ *be the solution produced by Algorithm 2. Then:*

- $\mathbf{y}$ *is feasible for the primal problem* (13)*;*
- $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ *is feasible for the dual problem* (14)*; and*

- *the two solutions are optimal for their respective problems.*

The proof of this result (see Section EC.1.9 of the ecompanion) is rather technical; it involves carefully reasoning about the steps of the two algorithms to show that both solutions are feasible and that they satisfy complementary slackness.

We illustrate how Algorithms 1 and 2 work through an example.

EXAMPLE 1. Consider an instance of the primal and dual subproblem with $n = 9$ products; including the no-purchase option, there are thus $n+1 = 10$ options in total. Assume that the ranking σ is as follows:

$$\sigma(1) < \sigma(2) < \sigma(3) < \sigma(4) < \sigma(5) < \sigma(0) < \sigma(6) < \sigma(7) < \sigma(8) < \sigma(9).$$

Assume also the following for the values of x_i and π_i (recall that $x_0 = 1$, $\pi_0 = 0$):

i	1	2	3	4	5	6	7	8	9	0
x_i	0.35	0.20	0.35	0.25	0.15	0.45	0.50	0.50	0.05	1.00
π_i	11	32	71	59	90	50	81	95	85	0

We first execute Algorithm 1, which creates the primal solution. Table 1 below shows the iterations of Algorithm 1, the computations that occur in each iteration and the values to which the y_j variables are set. After completing the primal phase, we execute Algorithm 2, which creates the dual solution. Table 2 below shows the steps of Algorithm 2, the computations that occur in each step and the values to which the dual variables $\alpha_0, \ldots, \alpha_n, \beta_0, \ldots, \beta_n, \gamma$ are set.

Figure 1 visualizes the primal and dual solutions before running Algorithms 1 and 2. The left-hand plot shows the options and their corresponding x_i values, while the right-hand side shows the options and their profit values π_i (the thick black horizontal lines).

Figure 2 visualizes the primal and dual solutions after Algorithms 1 and 2. The left-hand plot shows the values of the non-zero y_j values set in Algorithm 1. The text above each option indicates the order in which the options are traversed $(\tau(0), \tau(1), \ldots, \tau(n))$ and the events (A event/B event at i^*/C event) that occur. The right hand plot shows the dual solution: at each option i, the height of the colored bars indicates the value of $\alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j + \gamma$, which is the left-hand side of the dual constraint (14b), while the thick horizontal lines indicate π_i, which is on the right-hand side of the dual constraint. For all options i, the colored bars are at or above the horizontal lines, indicating that the solution satisfies constraint (14b).

The objective value of the primal solution is

$$\begin{aligned}\sum_{j=0}^{n} \pi_j y_j &= \pi_1 \times y_1 + \pi_3 \times y_3 + \pi_4 \times y_4 + \pi_5 \times y_5 \\ &= 11 \times 0.35 + 71 \times 0.35 + 59 \times 0.15 + 90 \times 0.15 \\ &= 51.05.\end{aligned}$$

The objective value of the dual solution is

$$\begin{aligned}&\sum_{j=0}^{n} x_j \cdot \alpha_j + \sum_{j=0}^{n} (1 - x_j) \cdot \beta_j + \gamma \\ &= x_3 \times \alpha_3 + x_5 \times \alpha_5 + (1 - x_1) \times \beta_1 + (1 - x_0) \times \beta_0 + \gamma \\ &= 0.35 \times 12 + 0.15 \times 31 + (1 - 0.35) \times 48 + (1 - 1) \times 36 + 11 \\ &= 51.05,\end{aligned}$$

which is identical to the primal solution, as expected. Figure 3 visualizes this equivalence by representing the objective as an area.

Table 1 Iterations and computations of the primal phase of the greedy algorithm (Algorithm 1) for Example 1.

Iteration	Values of q_1, q_2, q_3	Computations
(Initialization)	–	$y_0, \ldots, y_n \leftarrow 0$ $A, B, B_{main} \leftarrow \emptyset$
$\tau(0) = 8$	$q_1 = 1,\ q_2 = 0,\ q_3 = 0.5$	B event at $i^* = 0$ $B \leftarrow B \cup \{(0, 8)\}$ $B_{main} \leftarrow B_{main} \cup \{0\}$ $y_8 \leftarrow 0$
$\tau(1) = 5$	$q_1 = 1,\ q_2 = 0.65,\ q_3 = 0.15$	A event $A \leftarrow A \cup \{5\}$ $y_5 \leftarrow 0.15$
$\tau(2) = 9$	$q_1 = 0.85,\ q_2 = 0,\ q_3 = 0.05$	B event at $i^* = 0$ $y_9 \leftarrow 0$
$\tau(3) = 7$	$q_1 = 0.85,\ q_2 = 0,\ q_3 = 0.5$	B event at $i^* = 0$ $y_7 \leftarrow 0$
$\tau(4) = 3$	$q_1 = 0.85,\ q_2 = 0.5,\ q_3 = 0.35$	A event $A \leftarrow A \cup \{3\}$ $y_3 \leftarrow 0.35$
$\tau(5) = 4$	$q_1 = 0.5,\ q_2 = 0.15,\ q_3 = 0.25$	B event at $i^* = 1$ $B \leftarrow B \cup \{(1, 4)\}$ $B_{main} \leftarrow B_{main} \cup \{1\}$ $y_4 \leftarrow 0.15$
$\tau(6) = 6$	$q_1 = 0.35,\ q_2 = 0,\ q_3 = 0.45$	B event at $i^* = 1$ $y_6 \leftarrow 0$
$\tau(7) = 2$	$q_1 = 0.35,\ q_2 = 0,\ q_3 = 0.20$	B event at $i^* = 1$ $y_2 \leftarrow 0$
$\tau(8) = 1$	$q_1 = 0.35,\ q_3 = 0.35$	C event $C \leftarrow 1$ $y_1 \leftarrow 0.35$ **break**

4.3. Overall algorithmic approach

Our overall Benders solution approach operates in two phases and is summarized below:

1. **LO Relaxation Phase**. We solve the relaxation of problem (10) using constraint generation. In the process, we will have generated a set of cuts $\bar{A}_k$ for each customer type k.

2. **MIO Relaxation Phase**. We solve the integer version of problem (10) using lazy constraint generation. In addition, we warm-start the integer problem using the cuts $\bar{A}_k$ generated from the LO relaxation phase.

We conclude this section by commenting on two aspects of our Benders formulation. First, we emphasize that Benders decomposition is an exact method. In particular, the Benders reformulation (10) is an exact reformulation of the original MIO model (2): solving problem (10) is equivalent to solving problem (2), and will lead to a globally optimal solution of the first-choice PLD problem (1). In addition, constraint generation is an exact method for solving the Benders problem (10) and for solving the Benders LO relaxation (12). For completeness, we provide exact definitions of classical constraint generation algorithms for solving the relaxed Benders problem and the integer Benders problem in Sections EC.2.1 and EC.2.2 of the electronic companion, respectively. In addi-

Table 2 Steps and computations of the dual phase of the greedy algorithm (Algorithm 2) for Example 1.

Step	Computations
Initialization	$\alpha_0, \ldots, \alpha_n \leftarrow 0$ $\beta_0, \ldots, \beta_n \leftarrow 0$
Set γ	$\gamma \leftarrow \pi_1 = 11$
Define f	$f(0) = 8,\ f(1) = 4$
Sort B_{main}	$i_1 = 1,\ \ i_2 = 0$
β loop, $t = 1$	$\beta_1 \leftarrow \pi_4 - \gamma$ $= 59 - 11$ $= 48$
β loop, $t = 2$	$\beta_0 \leftarrow \pi_8 - \gamma - \beta_1$ $= 95 - 11 - 48$ $= 36$
α loop, $i = 5$	$\alpha_5 \leftarrow \pi_5 - \gamma - \beta_1$ $= 90 - 11 - 48$ $= 31$
α loop, $i = 3$	$\alpha_3 \leftarrow \pi_3 - \gamma - \beta_1$ $= 71 - 11 - 48$ $= 12$

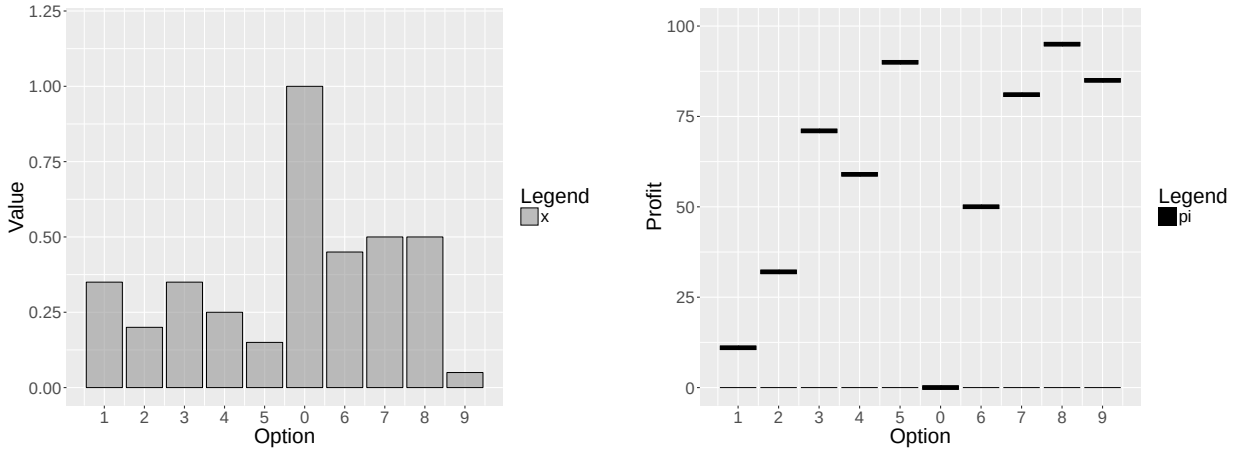

Figure 1 Primal (left) and dual (right) solutions before the execution of Algorithms 1 and 2 for Example 1.

tion, we also prove that these algorithms produce the optimal solution to their respective problems at termination (also in Sections EC.2.1 and EC.2.2) and terminate in finitely many iterations (Section EC.2.3 of the electronic companion); the latter result is shown by proving that Algorithms 1 and 2 produce extreme point solutions for their respective subproblems.

Second, we contrast our method with the solution method of Belloni et al. (2008). The method of Belloni et al. (2008) differs from our solution approach in two major dimensions. First, this method involves solving formulation (3), which we have shown in Theorem 1 to be weaker than our formulation (2). Second, the heart of the algorithmic approach in Belloni et al. (2008) is Lagrangean

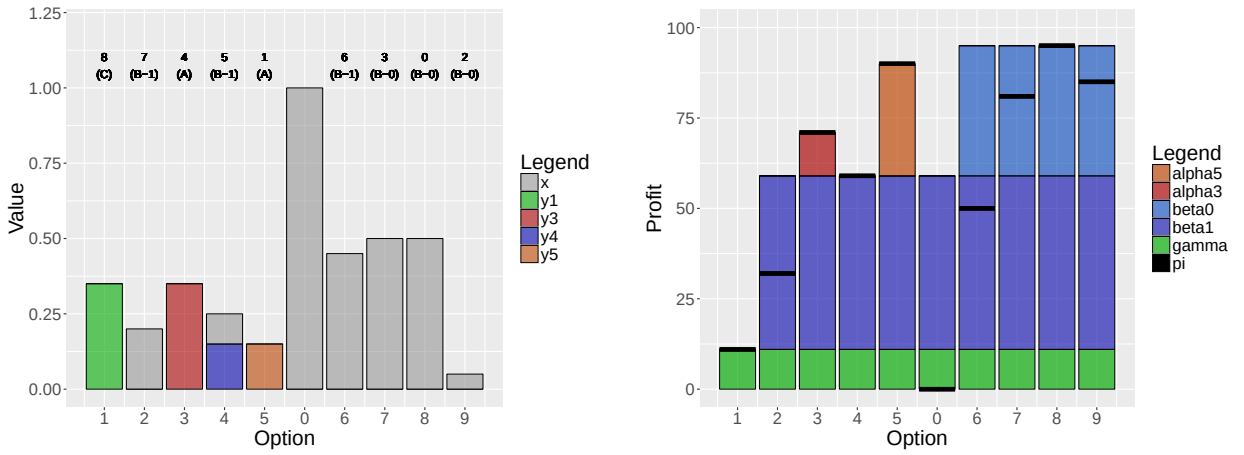

Figure 2 Primal (left) and dual (right) solutions after the execution of Algorithms 1 and 2 for Example 1.

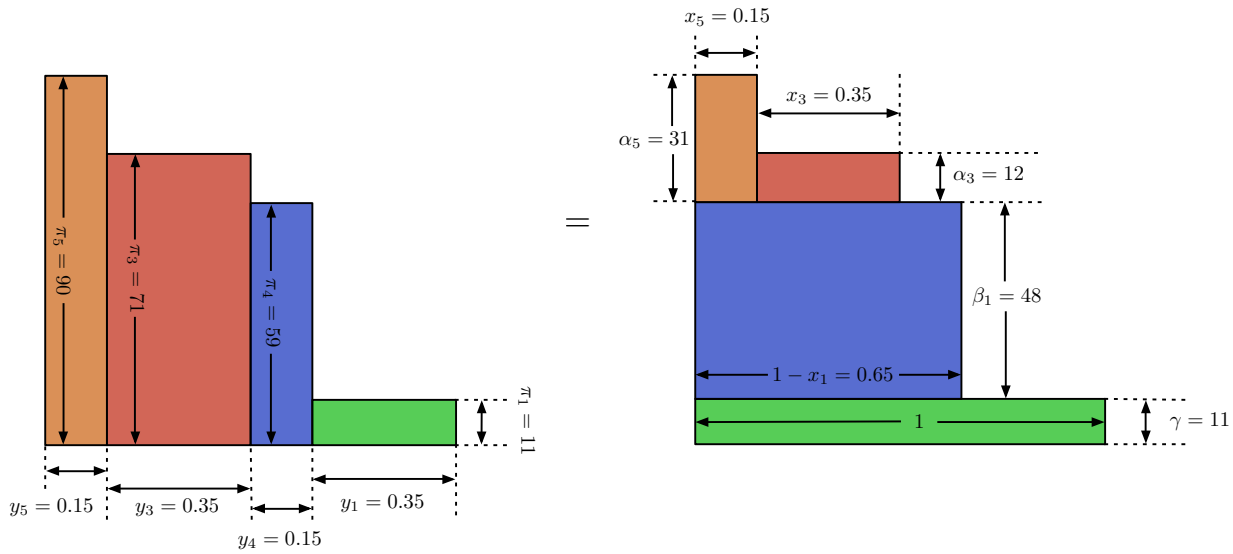

Figure 3 Representation of optimal objective value for Example 1 in terms of the primal solution (left) and dual solution (right). The colors of the left- and right-hand plots correspond to the left- and right-hand plots of Figure 2, respectively.

relaxation. The idea of the Lagrangean relaxation in Belloni et al. (2008) is to relax certain constraints in the formulation and to instead penalize violations of these constraints in the objective through Lagrange multipliers. Maximizing over the original $\mathbf{x}$ and $\mathbf{y}$ variables in this penalized version of the problem leads to an upper bound on the objective value of problem (3); by then minimizing over the Lagrange multipliers, one can obtain the tightest such bound. The solution method then essentially involves using this bound within branch-and-bound so as to more efficiently search the solution space. From a conceptual standpoint, this method is a completely different approach to large-scale optimization compared to our Benders method. The implementation requirements are also different. The first phase of our Benders method requires a simple constraint generation loop, while the second phase requires a lazy constraint callback, both of which are straightforward to implement using commercial solvers such as Gurobi. In contrast, the method of Belloni et al. (2008) requires the user to implement several rather sophisticated procedures, including a subgradient/bundle method for optimizing over the Lagrange multipliers in the Lagrangean bound

calculation, as well as a custom branch-and-bound implementation that can use the Lagrangean bound. Although it is subjective, we believe these components require a higher level of expertise and effort to develop, and we expect our method to be easier for practitioners to implement.

5. Computational results

In this section, we report on the results of our numerical experiments. Section 5.1 reports on the performance of our formulation, and Section 5.2 compares it to the other formulations of Section 3.3 using synthetic data. Section 5.3 focuses on our Benders decomposition approach from Section 4 and showcases the value of this approach in solving a product line design problem derived from a real conjoint data set.

All experiments were conducted on a late 2013 Apple MacBook Pro Retina laptop with a quad core 2.6GHz Intel i7 processor and 16GB of memory. All experiments were implemented in the Julia programming language, version 0.6.0 (Bezanson et al. 2017) using the `JuMP` package (Julia for Mathematical Programming; see Lubin and Dunning 2015, Dunning et al. 2017). All linear and mixed-integer optimization problems were solved using Gurobi 7.5 (Gurobi Optimization, Inc. 2015).

All of the code to perform these experiments, including the Benders decomposition algorithms, is freely available at `http://github.com/vvmisic/optimalPLD/`.

5.1. Experiments on synthetic data for formulation (2)

To test the tractability of the PLD formulation (2), we consider the following experiment. For fixed values of the number of products n and the number of rankings K, we randomly generate 20 instances, where we uniformly at random generate the set of rankings $\sigma^1, \ldots, \sigma^K$ from the set of all possible rankings, the profit π_i of each product i from the set $\{1, \ldots, 100\}$, and the probability distribution $\boldsymbol{\lambda}$ from the $(K-1)$-dimensional unit simplex. For each of these 20 instances, we solve problem (2), and record the time to solve the problem to full optimality. We solve the unconstrained problem (constraint (2f) is omitted) and the constrained problem, where we impose the constraint $\sum_{i=1}^n x_i \leq b$ for $b \in \{5, 10\}$. Table 3 shows the results of this experiment, for n up to 500 and K up to 1000; the times reported are the averages over the 20 instances. From this table, we can see that problem (2) is very tractable; even in the most challenging case ($n = 500, K = 500, b = 5$), it can be solved to full optimality in about half an hour on average, while for 200 or fewer products the problem can be solved within 15 minutes on average. In general, the constrained instances required more time than the unconstrained instances.

In addition to the instances shown in Table 3, we also consider twenty instances generated in the same way with $n = 500$ products and $K = 1000$ rankings. For each instance, we consider the unconstrained problem (constraint (2f) is omitted) and the constrained problem with the constraint $\sum_{i=1}^n x_i \leq 5$. Due to the difficulty of these instances, we solve these instances with a time limit of one hour. Table 4 shows the solution time and the final optimality gap averaged over the 20 instances, as well as how many of the instances are solved fully within the one hour time limit. We can see that with $b = 5$, solving problem (2) leads to good solutions, with an average optimality gap of 3.33%; as the problem becomes less constrained, the optimality gap improves (for example, 0.11% when unconstrained). In addition, the number of instances that can be solved to full optimality within the time limit improves as the problem becomes less constrained.

5.2. Experiments on synthetic data comparing formulations (2), (3), (4) and (5)

We now consider another experiment to compare the quality of the LO bounds from problem (2) (Z^*_{BM}) to those obtained from problems (3), (4) and (5). We solve the LO relaxation of each of these formulations restricted to $n \in \{20, 50, 100\}$. For the utility-based formulations (4) and (5), we set the utilities u_i^k as

$$u_i^k = n - \sigma^k(i). \tag{15}$$

Table 3 Solution times for problem (2).

n	K	b	Solution time (s)
20	100	5	0.10
20	100	10	0.09
20	100	–	0.08
20	200	5	0.24
20	200	10	0.25
20	200	–	0.23
20	500	5	1.51
20	500	10	1.03
20	500	–	0.78
20	1000	5	2.62
20	1000	10	1.62
20	1000	–	1.41
50	100	5	1.14
50	100	10	0.73
50	100	–	0.54
50	200	5	2.72
50	200	10	1.82
50	200	–	1.44
50	500	5	9.98
50	500	10	6.58
50	500	–	5.91
50	1000	5	24.00
50	1000	10	15.72
50	1000	–	13.98
100	100	5	4.05
100	100	10	3.22
100	100	–	2.55
100	200	5	12.17
100	200	10	9.92
100	200	–	7.18
100	500	5	36.15
100	500	10	28.21
100	500	–	27.13
100	1000	5	115.62
100	1000	10	94.39
100	1000	–	64.74
200	100	5	16.84
200	100	10	12.79
200	100	–	11.36
200	200	5	40.67
200	200	10	36.53
200	200	–	26.37
200	500	5	199.52
200	500	10	151.84
200	500	–	96.09
200	1000	5	827.85
200	1000	10	738.48
200	1000	–	410.10
500	100	5	70.80
500	100	10	63.71
500	100	–	50.37
500	200	5	191.24
500	200	10	184.09
500	200	–	149.43
500	500	5	1510.80
500	500	10	1497.67
500	500	–	741.34

Table 4 Results for synthetic instances with $n = 500$, $K = 1000$.

n	K	b	Solution time (s)	Gap (%)	Num. Solved
500	1000	5	3584.59	3.33	1
500	1000	10	3582.72	1.34	1
500	1000	–	2538.78	0.11	14

For each formulation F, we record the integrality gap. The integrality gap G_F is defined as the difference between the optimal value of the relaxation of F, Z_F^*, and the integer optimal value, Z^*, relative to Z^*:

$$G_F = 100\% \times \frac{Z_F^* - Z^*}{Z^*}.$$

Table 5 reports the average values of G_{BM}, G_{BFSS}, $G_{Utility}$ and G_{MZ}, where the four values correspond to the relaxations of problems (2), (3), (4) and (5), respectively. From this table, we can see that the average integrality gap of problem (2) (our formulation) is the lowest of the four – as predicted by Theorem 1. The gaps from our formulation are low: on average, they are below 2%. For the other formulations, the integrality gaps are higher; problem (3) has the second lowest, followed by problem (4) and finally problem (5). The gaps of these alternative formulations can

Table 5 Comparison of LO bounds.

n	K	b	G_{BM}	G_{BFSS}	$G_{Utility}$	G_{MZ}	n	K	b	G_{BM}	G_{BFSS}	$G_{Utility}$	G_{MZ}
20	100	5	0.46	3.10	15.27	28.32	50	500	5	1.73	4.12	12.08	22.41
20	100	10	0.36	4.12	15.82	27.98	50	500	10	0.52	5.35	11.22	17.87
20	100	–	0.36	4.12	15.82	27.98	50	500	–	0.48	6.35	11.74	17.78
20	200	5	0.65	3.85	17.25	35.91	50	1000	5	1.91	4.29	12.39	23.98
20	200	10	0.63	5.53	19.14	35.69	50	1000	10	0.86	5.84	11.84	19.48
20	200	–	0.63	5.53	19.16	35.69	50	1000	–	0.82	7.28	12.93	19.44
20	500	5	1.21	4.57	15.49	30.41	100	100	5	1.16	2.99	8.60	14.12
20	500	10	0.69	6.13	16.74	29.38	100	100	10	0.58	3.74	7.00	10.70
20	500	–	0.69	6.13	16.76	29.38	100	100	–	0.38	3.85	6.79	10.40
20	1000	5	0.53	3.78	14.50	30.82	100	200	5	2.04	3.89	10.15	16.97
20	1000	10	0.25	5.27	16.28	29.67	100	200	10	0.58	4.10	7.93	11.75
20	1000	–	0.25	5.27	16.35	29.67	100	200	–	0.44	4.95	8.06	11.40
50	100	5	1.29	3.90	10.84	18.77	100	500	5	2.28	4.04	11.01	19.45
50	100	10	0.55	4.52	9.49	15.62	100	500	10	1.08	4.49	8.70	13.56
50	100	–	0.52	4.62	9.45	15.57	100	500	–	0.76	5.89	9.13	12.98
50	200	5	1.19	3.75	11.59	21.30	100	1000	5	2.67	4.16	11.06	20.03
50	200	10	0.48	4.93	10.85	17.73	100	1000	10	1.23	4.49	8.70	13.75
50	200	–	0.44	5.44	10.95	17.64	100	1000	–	0.65	6.24	9.30	12.81

also be quite substantial; for $n = 100$, G_{BFSS} can be as high as 6%, whereas $G_{Utility}$ and G_{MZ} can be as high as 11% and 20%, respectively. These results show the value of the formulation of problem (2) in terms of providing a tighter upper bound on the true integer optimal value. Interestingly, despite the fact that the feasible regions of problems (3) and (4) are in general not contained in each other (Proposition 3), in all of the instances that we tested problem (3) yielded a tighter relaxation bound than problem (4).

We note that a potential limitation of the experiment presented in Table 5 is that the utilities used in problems (4) and (5) may not be representative of real utilities obtained through conjoint analysis. We thus conduct an additional set of experiments using the data from Toubia et al. (2003) (which will be described shortly in Section 5.3). These results are qualitatively similar; for completeness, we report them in Section EC.3.1 of the electronic companion.

To further see the value of problem (2), let us also compare the time required to solve the full integer problems (2), (3), (4) and (5). Table 6 shows the time required to solve these formulations for the same (n, K, b) combinations as in Table 5, averaged over the 20 instances for each (n, K, b) combination. Due to the prohibitively large computation time required by the alternative formulations, a time limit of one hour was imposed on formulations (3), (4) and (5). The columns labeled T indicate the computation time required to solve each formulation (subject to the aforementioned time limit). The columns labeled NU indicate how many instances out of the 20 for each (n, K) pair were not solved to full optimality within the one hour time limit.

From this table, we can see that the time required to solve problem (2) is significantly smaller than all of the other formulations. For example, for $n = 100$, $K = 500$ with no constraints, problem (2) takes under 30 seconds to solve on average, compared to about 7 minutes for (3), 20 minutes for (5) and 50 minutes for (4). Even with constraints, problem (2) provides an edge: for $n = 100$, $K = 500$ and $b = 5$, problem (2) is about ten times faster than problem (3), 100 times faster than problem (4) and 60 times faster than problem (5). In addition, formulation (2) is solved to full optimality within the one hour time limit for all of the instances. In contrast, the other formulations are sometimes not solved to optimality within the one hour time limit. Formulation (3) is not solved in one instance, while formulation (5) is not solved in 52 instances. The most striking behavior here is seen

Table 6 Comparison of MIO solution times and number of instances not solved within one hour.

n	K	b	T_{BM}	T_{BFSS}	$T_{Utility}$	T_{MZ}	NU_{BM}	NU_{BFSS}	$NU_{Utility}$	NU_{MZ}
20	100	5	0.10	0.34	1.08	0.83	0	0	0	0
20	100	10	0.09	0.35	0.94	0.80	0	0	0	0
20	100	–	0.08	0.39	0.98	0.80	0	0	0	0
20	200	5	0.24	0.88	2.90	2.92	0	0	0	0
20	200	10	0.25	0.87	2.42	2.86	0	0	0	0
20	200	–	0.23	1.04	2.37	2.97	0	0	0	0
20	500	5	1.51	4.53	13.09	17.60	0	0	0	0
20	500	10	1.03	4.38	10.38	15.41	0	0	0	0
20	500	–	0.78	4.71	10.44	15.60	0	0	0	0
20	1000	5	2.62	8.55	45.35	26.73	0	0	0	0
20	1000	10	1.62	8.05	30.84	19.29	0	0	0	0
20	1000	–	1.41	9.88	30.69	19.70	0	0	0	0
50	100	5	1.14	3.01	5.48	12.42	0	0	0	0
50	100	10	0.73	3.69	5.81	11.22	0	0	0	0
50	100	–	0.54	4.88	6.04	11.49	0	0	0	0
50	200	5	2.72	6.35	18.78	46.29	0	0	0	0
50	200	10	1.82	8.71	20.15	35.47	0	0	0	0
50	200	–	1.44	10.22	18.26	34.62	0	0	0	0
50	500	5	9.98	21.83	124.08	225.51	0	0	0	0
50	500	10	6.58	36.33	137.19	79.67	0	0	0	0
50	500	–	5.91	42.87	128.87	78.28	0	0	0	0
50	1000	5	24.00	68.24	662.11	1101.21	0	0	0	0
50	1000	10	15.72	112.63	780.73	502.26	0	0	0	0
50	1000	–	13.98	138.79	783.44	521.74	0	0	0	1
100	100	5	4.05	11.57	18.87	86.10	0	0	0	0
100	100	10	3.22	17.26	35.36	77.93	0	0	0	0
100	100	–	2.55	22.13	34.21	70.14	0	0	0	0
100	200	5	12.17	29.25	105.61	310.23	0	0	0	0
100	200	10	9.92	43.61	239.36	146.51	0	0	0	0
100	200	–	7.18	73.69	411.92	105.12	0	0	0	0
100	500	5	36.15	144.53	1294.43	2390.92	0	0	1	9
100	500	10	28.21	336.14	2639.80	1828.61	0	0	9	4
100	500	–	26.88	447.29	2888.77	1359.21	0	0	12	4
100	1000	5	115.62	714.65	3584.89	3601.76	0	0	19	20
100	1000	10	94.39	1457.48	3600.27	3546.21	0	0	20	17
100	1000	–	64.74	1426.24	3600.28	2669.32	0	1	20	7

for formulation (4), which is not solved in almost all of the $n = 100, K = 1000$ instances. Another interesting observation here is that, although problem (4) provides a bound that is theoretically at least as tight as problem (5) (Proposition 1) and numerically is tighter (Table 5 above), the integer version of problem (4) is significantly less tractable than (5): the time required to solve problem (4) is considerably higher and many more instances are not solved to full optimality within one hour of computation.

Table 7 Results of large-scale Benders experiment.

	D&C phase		LO Benders phase		MIO Benders phase			Gap (%)	
Constraints	Obj.	Time (s)	Bound	Time (s)	Obj.	Bound	Time (s)	Initial	Final
$\sum x_i = 2$	59.22	7.12	59.22	153.22	59.22	59.22	20.23	0.0	0.0
$\sum x_i = 3$	66.29	13.67	66.57	275.34	66.29	66.29	40.75	0.42	0.0
$\sum x_i = 4$	70.24	24.81	71.21	311.33	70.24	70.24	236.74	1.36	0.0
$\sum x_i = 5$	72.82	26.45	73.85	324.20	72.82	72.82	255.58	1.39	0.0
$\sum x_i = 10$	77.33	87.69	79.36	584.40	77.41	77.42	13679.69	2.56	0.01
$\sum x_i = 20$	80.14	219.18	82.65	1019.75	80.14	81.56	21600.03	3.04	1.74
$\sum x_i = 50$	81.85	1266.09	85.01	2418.44	81.85	84.46	21600.05	3.71	3.09

5.3. Experiments with real data

In Section 5.1, we considered instances of the first-choice PLD problem that were of a small to medium scale. In this section, we showcase the scalability of our formulation to large-scale instances, using the Benders-based solution method outlined in Section 4.3.

To test the Benders solution scheme, we will deviate from our approach in Section 5.1 by considering instances derived from real data. The real data set that we will use is from the paper of Toubia et al. (2003). In this paper, the authors proposed a new method for estimating attribute-level utility functions from pairwise comparison data. To evaluate the method, the authors conducted a field experiment to measure consumer preferences for a new laptop bag product to be launched by a real firm, Timbuk2 (Timbuk2 Designs Inc., San Francisco, CA, USA). The authors collected pairwise comparison data from 330 respondents, and used these to estimate respondent-level utility functions. A later paper, Belloni et al. (2008), used these utility functions, along with incremental revenue and cost data, for the purpose of selecting a product line. They formulated an optimization problem (problem (3)) for selecting a product line from a collection of 3584 candidate products so as to maximize the profit when each customer chooses the available option with the highest utility (either one of the products in the product line, or the no-purchase option).

We consider formulation (2) with the $n = 3584$ candidate products and $K = 330$ rankings of Belloni et al. (2008) with different constraints on the product line. We consider instances with different product line widths, i.e., each instance has a constraint of the form $\sum x_i = t$, where the product line width t ranges in $\{2, 3, 4, 5, 10, 20, 50\}$. We solve each instance (problem (2) with a specific set of constraints) using the Benders-based solution method. We run the LO phase of the method without any time limit. We terminate the MIO phase of the method after six hours of computation time or an optimality gap of 0.01% or lower. For each instance, we run the divide and conquer (D&C) heuristic (see Green and Krieger 1993) from ten random starting solutions, and retain the best solution; we provide the best solution to the MIO solver to warm-start the MIO phase of the Benders method.

Table 7 shows the results of this experiment, with each row corresponding to a different instance. The first column indicates the constraints that were imposed on the instance. The next two columns indicate the objective of the D&C solution and the time required to attain this solution. The next two columns indicate the LO relaxation bound and the time required to compute this bound (i.e., the time for the LO phase of the Benders method). The next three columns indicate, for the integer phase of the Benders method, the objective value of the best integer solution, the best bound for this solution, and the time required for the integer phase. Finally, the last two columns indicate the initial gap when starting the MIO phase (this is the gap of the D&C solution relative to the bound from the LO phase of the Benders method) and the final gap (this is the gap of the best MIO solution relative to the best MIO bound). All times are reported in seconds.

From this table, we see that for most of the instances, the LO Benders phase completes quickly. For example, when the width of the product line is 20, the LO Benders phase requires just over 15 minutes to complete. In the largest case ($\sum x_i = 50$), the LO Benders phase takes approximately 40

minutes to run. Note also that, even before the MIO phase, the initial solution provided by D&C is a very good solution; the initial optimality gap is under 4% for all of the instances. We have empirically observed that warm-starting the MIO phase using D&C is beneficial; for those instances with a smaller number of feasible solutions ($\sum x_i = t$ for $t \leq 5$), starting from the D&C solution did reduce the time to prove optimality, while for the larger instances, starting from the D&C solution led to a better gap after six hours of execution. For completeness, we report the results of our Benders method without warm-starting in Section EC.3.2 of the electronic companion.

With regard to the MIO phase, for the more constrained instances (size of product line is ten or lower), the MIO phase is able to prove optimality (final gap of 0.01% or lower) within the six hour time limit. For the less constrained instances (size of product line is greater than ten), the six hour time limit is reached, and the MIO phase is able to attain some reduction in the gap; this reduction becomes more modest as the bound on the size of the product line increases.

Aside from these general insights, we highlight two other important aspects of this experiment. The first is that we were not able to solve any of these instances by directly solving problem (2) with Gurobi, due to the extremely large memory requirement of problem (2) at this scale. Thus, the Benders approach really is necessary for solving problem (2). This highlights the value of the Benders approach, in that it allows us to solve instances that are simply too large for direct solution by solvers like Gurobi.

The second is to do with the fourth instance, which corresponds to $\sum_i x_i = 5$. This instance is exactly the problem that was solved in Belloni et al. (2008). Their paper solved the problem using an entirely different approach from us; first, they used formulation (3), and second, they solved the problem using a combination of ideas such as Lagrangean relaxation, subgradient descent and valid inequalities, but without the use of Benders decomposition. Even with all of these techniques, this earlier approach required *one week* of computation time to solve the problem to full optimality. In stark contrast, our combined LO and MIO Benders methods required a total of 606.22 seconds – *just over ten minutes* – to solve the problem to full optimality. This is a striking reduction. Admittedly, some of this reduction is likely due to hardware improvements and general improvements in LO/MIO solution software since the time that paper was written (2008) to now; however, we believe that these computational results are indicative of the potential of our Benders-based method to efficiently obtain tight upper bounds and solve practically-sized PLD problems.

6. Conclusions

In this paper, we presented a modern mixed-integer optimization approach for the problem of selecting a product line from a large set of candidate products under a first-choice model of customer behavior. We proposed a novel MIO formulation that we showed is theoretically stronger than three alternate formulations, thus providing a unified perspective on different formulations for this problem. We also developed a novel solution algorithm, based on Benders decomposition, by theoretically exploiting the structure of our MIO formulation. Through numerical experiments with synthetic data, we verified that our formulation yields tighter relaxation bounds and can be solved significantly faster than the alternate formulations on small to medium-sized instances. Through numerical experiments with a real conjoint data set, we show that our Benders decomposition approach advances the state-of-the-art in solving large-scale PLD instances, by showing that a PLD instance that required one week of computation time in 2008 can be solved to provable optimality by our approach in approximately ten minutes on a consumer laptop. We hope that these results will encourage the further exploration of linear and mixed-integer optimization modeling in the context of product design, and marketing science in general.

Acknowledgments

The authors thank the area editor Alexandre Belloni, the former area editor Harikesh Nair, the associate editor and the two referees for their helpful comments that greatly improved the paper. The authors thank the authors of Toubia et al. (2003) for graciously making their data set available, which was used in the numerical experiment in Section 5.3. The work of the second author was supported by a PGS-D award from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

A. Aouad, V. F. Farias, and R. Levi. Assortment optimization under consider-then-choose choice models. *Available at SSRN 2618823*, 2015.

A. Aouad, V. F. Farias, R. Levi, and D. Segev. The approximability of assortment optimization under ranking preferences. *Operations Research*, 2017. Forthcoming.

A. Belloni, R. Freund, M. Selove, and D. Simester. Optimizing product line designs: Efficient methods and comparisons. *Management Science*, 54(9):1544–1552, 2008.

D. Bertsimas and V. V. Mišić. Robust product line design. *Operations Research*, 65(1):19–37, 2017.

D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific, Belmont, MA, 1997.

D. Bertsimas and R. Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas Belmont, 2005.

J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

J. Blanchet, G. Gallego, and V. Goyal. A Markov chain approximation to choice modeling. *Operations Research*, 64(4):886–905, 2016.

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

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

I. Contreras, J.-F. Cordeau, and G. Laporte. Benders decomposition for large-scale uncapacitated hub location. *Operations research*, 59(6):1477–1490, 2011.

J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388, 2001.

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

G. Dobson and S. Kalish. Positioning and pricing a product line. *Marketing Science*, 7(2):107–125, 1988.

I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

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

J. B. Feldman and H. Topaloglu. Revenue Management Under the Markov Chain Choice Model. *Operations Research*, 65(5):1322–1342, 2017.

A. M. Geoffrion and G. W. Graves. Multicommodity distribution system design by benders decomposition. *Management Science*, 20(5):822–844, 1974.

P. E. Green and A. M. Krieger. Models and heuristics for product line selection. *Marketing Science*, 4(1): 1–19, 1985.

P. E. Green and A. M. Krieger. Conjoint analysis with product-positioning applications. In J. Eliashberg and G. L. Lilien, editors, *Handbooks in Operations Research and Management Science*, volume 5, pages 467–515. Elsevier, 1993.

Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual, 2015. URL `http://www.gurobi.com`.

D. Honhon, S. Jonnalagedda, and X. A. Pan. Optimal algorithms for assortment selection under ranking-based consumer choice models. *Manufacturing & Service Operations Management*, 14(2):279–289, 2012.

R. Kohli and K. Jedidi. Error theory for elimination by aspects. *Operations Research*, 63(3):512–526, 2015.

R. Kohli and R. Krishnamurti. A heuristic approach to product design. *Management Science*, pages 1523–1533, 1987.

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.

R. Kohli and R. Sukumar. Heuristics for product-line design using conjoint analysis. *Management Science*, 36(12):1464–1478, 1990.

R. Kohli, K. Boughanmi, and V. Kohli. Randomized algorithms for lexicographic inference. *Operations Research*, forthcoming, 2018.

U. G. Kraus and C. A. Yano. Product line selection and pricing under a share-of-surplus choice model. *European Journal of Operational Research*, 150(3):653–671, 2003.

G. Li, P. Rusmevichientong, and H. Topaloglu. The d-level nested logit model: Assortment and price optimization problems. *Operations Research*, 63(2):325–342, 2015.

M. Lubin and I. Dunning. Computing in Operations Research Using Julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

L. Luo. Product line design for consumer durables: an integrated marketing and engineering approach. *Journal of Marketing Research*, 48(1):128–139, 2011.

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.

V. V. Mišić. *Data, models and decisions for large-scale stochastic optimization problems*. PhD thesis, Massachusetts Institute of Technology, 2016.

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley Interscience, 1988.

R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

P. Rusmevichientong and H. Topaloglu. Robust assortment optimization in revenue management under the multinomial logit choice model. *Operations Research*, 60(4):865–882, 2012.

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.

C. Schön. On the optimal product line selection problem with price discrimination. *Management Science*, 56(5):896–902, 2010a.

C. Schön. On the product line selection problem under attraction choice models of consumer behavior. *European Journal of Operational Research*, 206(1):260–264, 2010b.

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

O. Toubia, D. I. Simester, J. R. Hauser, and E. Dahan. Fast polyhedral adaptive conjoint estimation. *Marketing Science*, 22(3):273–303, 2003.

O. Toubia, J. R. Hauser, and D. I. Simester. Polyhedral methods for adaptive choice-based conjoint analysis. *Journal of Marketing Research*, 41(1):116–131, 2004.

G. van Ryzin and G. Vulcano. A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Science*, 61(2):281–300, 2015.

M. Yee, E. Dahan, J. R. Hauser, and J. Orlin. Greedoid-based noncompensatory inference. *Marketing Science*, 26(4):532–549, 2007.

Electronic companion for "Exact first-choice product line optimization" by D. Bertsimas and V. V. Mišić

EC.1. Proofs

EC.1.1. Proof of Proposition 1

To prove that $\mathcal{F}_{Utility} \subseteq \mathcal{F}_{MZ}$, let $(\mathbf{x}, \mathbf{y})$ be an optimal solution to the relaxation of problem (4). Observe that constraints (5b), (5c) and (5g) are automatically satisfied by $(\mathbf{x}, \mathbf{y})$ as these constraints are also found in problem (4). Therefore, we only need to verify constraints (5d), (5e) and (5f).

To verify constraint (5d), let us start from constraint (4d) where the right-hand side product is j:

$$\sum_{j'=0}^{n} u_{j'}^k y_{j'}^k \geq (u_j^k - L^k)x_j + L^k.$$

Since the utilities are non-negative and $x_j \leq 1$, we have that

$$(u_j^k - L^k)x_j + L^k = u_j^k x_j + L^k(1 - x_j) \geq u_j^k x_j,$$

which allows us to assert that

$$\sum_{j'=0}^{n} u_{j'}^k y_{j'}^k \geq u_j^k x_j. \tag{EC.1}$$

The left-hand side can be upper-bounded as

$$u_i^k x_i + \sum_{\substack{j'=0 \\ j' \neq i}}^{n} U^k y_{j'}^k \geq \sum_{j'=0}^{n} u_{j'}^k y_{j'}^k,$$

since $y_j^k \leq x_j$ for all j, $U^k \geq u_j^k$ for all j, and all y_j^k are nonnegative. Combining this with (EC.1), we get that

$$u_i^k x_i + \sum_{\substack{j'=0 \\ j' \neq i}}^{n} U^k y_{j'}^k \geq u_j^k x_j.$$

We can re-arrange this to obtain

$$\begin{aligned} u_i^k x_i &\geq u_j^k x_j - \sum_{\substack{j'=0 \\ j' \neq i}}^{n} U^k y_{j'}^k \\ &= u_j^k x_j - U^k \cdot (1 - y_i^k), \end{aligned}$$

where the second step follows by the fact that $\sum_{j'=0}^{n} y_{j'}^k = 1$. This establishes that $(\mathbf{x}, \mathbf{y})$ satisfies the constraint

$$u_i^k x_i \geq u_j^k x_j - U^k \cdot (1 - y_i^k),$$

as required. Similar reasoning can be used to establish that constraints (5e) and (5f) also hold. $\square$

EC.1.2. Proof of Theorem 1

EC.1.2.1. Proof of Part (a) Let $(\mathbf{x}, \mathbf{y})$ be an optimal solution to the relaxation of problem (2). We need to establish that $(\mathbf{x}, \mathbf{y})$ is feasible for problem (3). We begin by observing that constraints (3b), (3c) and (3f) are automatically satisfied, since these constraints also exist in problem (2). Therefore, we only need to focus on constraints (3d) and (3e).

To establish constraint (3d), observe that $(\mathbf{x}, \mathbf{y})$ satisfies constraint (2d), so for each $i \in \{1, \ldots, n\}$, we have

$$\sum_{j:\sigma^k(j)>\sigma^k(i)} y_j^k \leq 1 - x_i.$$

Observe that for any j with $\sigma^k(j) > \sigma^k(i)$, we have that

$$y_j^k \leq \sum_{j':\sigma^k(j')>\sigma^k(i)} y_{j'}^k$$

which, combined with constraint (2d), lets us assert that

$$y_j^k \leq 1 - x_i,$$

for every $i \in \{1, \ldots, n\}$ and every j with $\sigma^k(j) > \sigma^k(i)$. This establishes that $(\mathbf{x}, \mathbf{y})$ satisfies constraint (3d); similar reasoning allows us to establish that constraint (3e) is also satisfied. $\square$

EC.1.2.2. Proof of Part (b) Let $(\mathbf{x}, \mathbf{y})$ be an optimal solution to the relaxation of problem (2). We need to establish that $(\mathbf{x}, \mathbf{y})$ is feasible for problem (4). We begin by observing that constraints (4b), (4c) and (4f) are automatically satisfied, since these constraints also exist in problem (2). Therefore, we only need to focus on constraints (4d) and (4e).

Before we begin, let us establish a useful identity for $(\mathbf{x}, \mathbf{y})$. Constraint (2d) is

$$\sum_{j:\sigma^k(j)>\sigma^k(i)} y_j^k \leq 1 - x_i,$$

which can be re-arranged to obtain

$$x_i \leq 1 - \sum_{j:\sigma^k(j)>\sigma^k(i)} y_j^k.$$

Combining this last inequality with constraint (2b), we obtain

$$x_i \leq \sum_{j:\sigma^k(j)\leq\sigma^k(i)} y_j^k. \tag{EC.2}$$

Now, to show that constraint (4d) holds, we have

$$\begin{aligned}
(u_i^k - L^k)x_i + L^k &\leq (u_i^k - L^k) \sum_{j:\sigma^k(j)\leq\sigma^k(i)} y_j^k + L^k \\
&= (u_i^k - L^k) \sum_{j:\sigma^k(j)\leq\sigma^k(i)} y_j^k + L^k \sum_{j=0}^{n} y_j^k \\
&= (u_i^k - L^k) \sum_{j:\sigma^k(j)\leq\sigma^k(i)} y_j^k + L^k \sum_{j:\sigma^k(j)\leq\sigma^k(i)} y_j^k + L^k \sum_{j:\sigma^k(j)>\sigma^k(i)} y_j^k \\
&= \sum_{j:\sigma^k(j)\leq\sigma^k(i)} u_i^k y_j^k + \sum_{j:\sigma^k(j)>\sigma^k(i)} L^k y_j^k
\end{aligned}$$

$$\begin{aligned}
&\leq \sum_{j:\sigma^k(j)\leq\sigma^k(i)} u_j^k y_j^k + \sum_{j:\sigma^k(j)>\sigma^k(i)} u_j^k y_j^k \\
&= \sum_{j=0}^{n} u_j^k y_j^k
\end{aligned}$$

where the first step follows by identity (EC.2) and the fact that $(u_i^k - L^k)$ is nonnegative (recall the definition of L^k as $L^k = \min_{0\leq i\leq n} u_i^k$); the second step follows by constraint (2b); the third step follows by splitting the second sum from the second step; the fourth step follows by putting together the first two sums in the third step, and then moving the coefficient in front of each sum to inside its respective sum; the fifth step follows by the fact that each y_j^k is nonnegative, each u_j^k is nonnegative, that

$$u_j^k \geq u_i^k$$

whenever $\sigma^k(j) \leq \sigma^k(i)$, and that $L^k \leq u_j^k$ for all j; and the final step by simple algebra. Similar reasoning can be used to establish that constraint (4e) holds. $\square$

EC.1.2.3. Proof of Part (c) Let $(\mathbf{x}, \mathbf{y})$ be an optimal solution to the relaxation of problem (2). We need to establish that $(\mathbf{x}, \mathbf{y})$ is feasible for problem (5). As in parts (a) and (b), constraints (5b), (5c) and (5g) are automatically satisfied as these constraints are also found in problem (2). We thus only need to establish that constraints (5d), (5e) and (5f) hold.

We begin by considering constraint (5d). We proceed in two cases.

Case 1: $u_j^k < u_i^k$. In this case, we have

$$\begin{aligned}
u_j^k x_j - U^k(1-y_i^k) &\leq u_j^k - U^k(1-y_i^k) \\
&\leq u_j^k - u_i^k(1-x_i) \\
&= u_j^k - u_i^k + u_i^k x_i \\
&\leq u_i^k x_i,
\end{aligned}$$

where the first step follows by the fact that u_j^k is nonnegative and x_j is upper bounded by 1; the second step follows by the fact that $U^k \geq u_i^k$ for all i and the fact that $x_i \leq y_i^k$, which can be re-arranged to assert that $(1-y_i^k) \geq (1-x_i)$; the third step by algebra; and the fourth by the assumption of this case.

Case 2: $u_j^k > u_i^k$. Recall that by constraint (2d), we have

$$y_j^k \leq \sum_{j':\sigma^k(j')>\sigma^k(i)} y_i^k \leq 1 - x_i$$

for any $\sigma^k(j) > \sigma^k(i)$; re-arranging, we get that whenever $\sigma^k(j) > \sigma^k(i)$, we have

$$x_i \leq 1 - y_j^k. \tag{EC.3}$$

Now, observe that, since $u_j^k > u_i^k$, it must be that $\sigma^k(j) < \sigma^k(i)$. We therefore have

$$\begin{aligned}
u_j^k x_j - U^k(1-y_i^k) &\leq u_j^k x_j - U^k x_j \\
&\leq 0 \\
&\leq u_i^k x_i,
\end{aligned}$$

where the first step follows by inequality (EC.3); the second step follows by the fact that $U^k \geq u_j^k$ for all j; and the final step by the fact that both u_i^k and x_i are nonnegative.

This establishes constraint (5d). Similar reasoning can be used to establish constraints (5e) and (5f). $\square$

EC.1.3. Proof of Proposition 2

The proof of Proposition 2 follows almost immediately from the proof of Part (c) of Theorem 1. As in part (c) of Theorem 1, the only constraints that need to be established are (5d), (5e) and (5f); just as in part (c) of Theorem 1, we will only focus on (5d).

Constraint (5d) can be established in the same way: case 1 follows through without any modifications, and for case 2, the key inequality that is needed is

$$y_j^k \leq 1 - x_i$$

for any j such that $\sigma^k(j) > \sigma^k(i)$. For a feasible solution of our problem (2), this was established as an implication of constraint (2d). For a feasible solution of problem (3), it is even more straightforward because this inequality is actually a constraint of problem (3) (specifically constraint (3d)) and so is automatically available to us.

Thus, the same steps used in the proof of part (c) of Theorem 1 are applicable here, which establishes the result. $\square$

EC.1.4. Proof of Proposition 3

We first prove that $\mathcal{F}_{BFSS} \nsubseteq \mathcal{F}_{Utility}$. We will prove this through a counterexample. Consider an instance with $n = 5$ and $K = 1$, and the ranking σ defined by $\sigma(i) = i - 1$ for $i \in \{1, \ldots, 5\}$ and $\sigma(0) = 5$ (we drop the k superscript for convenience). Set $\mathbf{C} = [\mathbf{0}^T], \mathbf{d} = [0]$ (i.e., the constraint $\mathbf{Cx} \leq \mathbf{d}$ is vacuous). Let the utility function be given by $u_i = 6 - i$ for $i \in \{1, \ldots, 5\}$ and $u_0 = 0$, so that $L = 0$.

We can see that the following is a feasible solution of $\mathcal{F}_{BFSS}$:

$$\begin{aligned} x_1 &= 0.5, \quad y_1 = 0, \\ x_2 &= 0.5, \quad y_2 = 0, \\ x_3 &= 0.5, \quad y_3 = 0, \\ x_4 &= 0.5, \quad y_4 = 0.5, \\ x_5 &= 0.5, \quad y_5 = 0.5, \\ & \qquad y_0 = 0. \end{aligned}$$

However, this solution does not belong to $\mathcal{F}_{Utility}$. To see this, observe that $\sum_{j=0}^{n} u_j \cdot y_j = 1.5$. However, for $i = 1$, we have

$$\begin{aligned} (u_i - L)x_i + L &= (5 - 0) \times 0.5 + 0 \\ &= 2.5 \\ &\nleq \sum_{j=0}^{n} u_j y_j = 1.5, \end{aligned}$$

i.e., constraint (4d) of problem (4) is violated. Thus, the candidate solution $(\mathbf{x}, \mathbf{y})$ is not in $\mathcal{F}_{Utility}$. This shows that in general, $\mathcal{F}_{BFSS}$ is not contained in $\mathcal{F}_{Utility}$.

We now prove that $\mathcal{F}_{Utility} \nsubseteq \mathcal{F}_{BFSS}$. Consider the same instance as above. The following solution is a feasible solution of $\mathcal{F}_{Utility}$:

$$\begin{aligned} x_1 &= 0.8, \quad y_1 = 0.5, \\ x_2 &= 0.5, \quad y_2 = 0.5, \\ x_3 &= 0, \quad\ \ y_3 = 0, \\ x_4 &= 0, \quad\ \ y_4 = 0, \\ x_5 &= 0, \quad\ \ y_5 = 0, \\ & \qquad y_0 = 0. \end{aligned}$$

The solution is not, however, a feasible solution of $\mathcal{F}_{BFSS}$. To see this, observe that we should have $y_2 \leq 1 - x_1$, by virtue of constraint (3d), but in the above solution, $y_2 = 0.5 \nleq 1 - x_1 = 1 - 0.8 = 0.2$. Therefore, in general, $\mathcal{F}_{Utility} \nsubseteq \mathcal{F}_{BFSS}$. $\square$

EC.1.5. Proof of Proposition 4

When $K = 1$ and the constraint $\mathbf{Cx} \leq \mathbf{d}$ is removed from the formulation, the feasible region $\mathcal{F}$ of the LO relaxation of problem (2) is the set of $(\mathbf{x}, \mathbf{y})$ that satisfy the following set of constraints:

$$x_i + \sum_{j:\sigma(j)>\sigma(i)} y_j \leq 1, \quad \forall i \in \{1, \ldots, n\}, \tag{EC.4}$$

$$\sum_{j:\sigma(j)>\sigma(0)} y_j \leq 0, \tag{EC.5}$$

$$-x_i + y_i \leq 0, \quad \forall i \in \{1, \ldots, n\}, \tag{EC.6}$$

$$x_i \leq 1, \quad \forall i \in \{1, \ldots, n\}, \tag{EC.7}$$

$$\sum_{j=0}^{n} y_j \leq 1, \tag{EC.8}$$

$$-\sum_{j=0}^{n} y_j \leq -1, \tag{EC.9}$$

$$\mathbf{x} \geq \mathbf{0}, \tag{EC.10}$$

$$\mathbf{y} \geq \mathbf{0}. \tag{EC.11}$$

Note that in the above system, we have re-arranged the inequalities so that all variables are on one side. Note also that constraint (2e) is re-expressed as an inequality and the unit sum constraint (2b) is expressed as two inequalities. In matrix form, the above system can be re-written as

$$\mathbf{A} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \leq \mathbf{b}, \tag{EC.12}$$

$$\mathbf{x}, \mathbf{y} \geq \mathbf{0}. \tag{EC.13}$$

To show that $\mathcal{F}$ is integral, we will first show that the matrix $\mathbf{A}$ is totally unimodular.

To prove that $\mathbf{A}$ is totally unimodular, we will use the following characterization of total unimodularity (see Bertsimas and Weismantel 2005):

PROPOSITION EC.1 **(Corollary 3.2 from Bertsimas and Weismantel 2005)**. *A matrix* $\mathbf{A}$ *is totally unimodular if and only if each collection* Q *of rows of* $\mathbf{A}$ *can be partitioned into two parts so that the sum of the rows in one part minus the sum of the rows in the other is a vector with entries only in* $\{0, +1, -1\}$.

For notational convenience, instead of working with rows, we will work in terms of algebraic expressions involving $\mathbf{x}$ and $\mathbf{y}$. There are four types of expressions which we denoted by A, B, C and D:

$$\mathrm{A}(i),\ i \in \{1, \ldots, n\}: \quad x_i + \sum_{j:\sigma(j)>\sigma(i)} y_j \tag{EC.14}$$

$$\mathrm{A}(0): \quad \sum_{j:\sigma(j)>\sigma(0)} y_j \tag{EC.15}$$

$$\mathrm{B}(i), i \in \{1, \ldots, n\}: \quad -x_i + y_i \tag{EC.16}$$

$$\mathrm{C}(i), i \in \{1, \ldots, n\}: \quad x_i \tag{EC.17}$$

$$\mathrm{D}(1): \qquad \sum_{j=0}^{n} y_j \tag{EC.18}$$

$$\mathrm{D}(2): \qquad -\sum_{j=0}^{n} y_j \tag{EC.19}$$

Instead of working with a collection of rows Q, we will assume that of each type of expression (A, B, C and D) we are given a collection of expressions:

$$\begin{aligned} &S_{\mathrm{A}} \subseteq \{0,1,\ldots,n\},\\ &S_{\mathrm{B}} \subseteq \{1,\ldots,n\},\\ &S_{\mathrm{C}} \subseteq \{1,\ldots,n\},\\ &S_{\mathrm{D}} \subseteq \{1,2\}. \end{aligned}$$

To establish the equivalent condition in Proposition EC.1, we will show that given $S_{\mathrm{A}}, S_{\mathrm{B}}, S_{\mathrm{C}}, S_{\mathrm{D}}$, we can partition the expressions into two groups R_+, R_- such that the difference of the sums of the expressions in each group will yield an expression

$$\sum_{e \in R_+} e - \sum_{e \in R_-} e = \sum_{i=1}^{n} q_i x_i + \sum_{j=0}^{n} w_j y_j,$$

where each q_i and each w_j is in $\{0, +1, -1\}$. This will establish that $\mathbf{A}$ is totally unimodular.

We provide a constructive procedure for generating a valid partition. This procedure proceeds in four steps.

Step 1. Sort the indices $i \in S_{\mathrm{A}}$ according to σ. Specifically, obtain the ordering $i_1, i_2, \ldots, i_{|S_{\mathrm{A}}|}$, such that

$$\sigma(i_1) < \sigma(i_2) < \cdots < \sigma(i_{|S_{\mathrm{A}}|}).$$

Now, for each $i_j \in S_{\mathrm{A}}$:

- If j is odd, put expression $\mathrm{A}(i_j)$ in R_+. If additionally $i_j \in S_{\mathrm{B}}$, put $\mathrm{B}(i_j)$ in R_+.
- If j is even, put expression $\mathrm{A}(i_j)$ in R_-. If additionally $i_j \in S_{\mathrm{B}}$, put $\mathrm{B}(i_j)$ in R_-.

If we evaluate $\sum_{e \in R_+} e - \sum_{e \in R_-} e$, we will obtain an expression of the following form:

$$\sum_{i \in I_+} (+1) x_i + \sum_{i \in I_-} (-1) x_i + \sum_{j \in J} (+1) y_j,$$

where $J \subseteq \{0,1,\ldots,n\}$, and where $I_+, I_- \subseteq \{1,\ldots,n\}$. Note that by the above procedure, the resulting coefficient of each x_i is either 0 (if the corresponding i was in both S_{A} and S_{B}, or not in S_{A}), +1 (if the corresponding $i = i_j$ was in S_{A} and not in S_{B}, and j was odd) or -1 (if the corresponding $i = i_j$ was in S_{A} and not in S_{B}, and j was even). Note that I_+ and I_- do not intersect – each $i \in I_+$ corresponds to an i_j for an odd j, and each $i \in I_-$ corresponds to an i_j for an even j.

Lastly, note that the coefficients on the y_j's so far are either 0 or +1. This is assured because we have sorted the i's from most to least preferred in terms of σ, and so the y_j's that participate in $\mathrm{A}(i_1), \mathrm{A}(i_2), \ldots, \mathrm{A}(i_{|S_{\mathrm{A}}|})$ form a nested sequence. In particular, for odd $|S_{\mathrm{A}}|$, we have

$$\begin{aligned} &\mathrm{A}(i_1) - \mathrm{A}(i_2) + \mathrm{A}(i_3) - \mathrm{A}(i_4) + \cdots + \mathrm{A}(|S_{\mathrm{A}}|)\\ &= x_{i_1} - x_{i_2} + x_{i_3} - x_{i_4} + \cdots + x_{i_{\mathrm{A}}} +\\ &\quad + \sum_{j:\sigma(j)>\sigma(i_1)} y_j - \sum_{j:\sigma(j)>\sigma(i_2)} y_j + \sum_{j:\sigma(j)>\sigma(i_3)} y_j - \sum_{j:\sigma(j)>\sigma(i_4)} + \cdots + \sum_{j:\sigma(j)>\sigma(i_{|S_{\mathrm{A}}|})} y_j\\ &= x_{i_1} - x_{i_2} + x_{i_3} - x_{i_4} + \cdots + x_{i_{\mathrm{A}}}\\ &\quad + \sum_{j:\sigma(i_2)\geq\sigma(j)>\sigma(i_1)} y_j + \sum_{j:\sigma(i_4)\geq\sigma(j)>\sigma(i_3)} y_j + \cdots + \sum_{j:\sigma(j)>\sigma(i_{|S_{\mathrm{A}}|})} y_j. \end{aligned}$$

and for even $|S_{\mathrm{A}}|$ we end up with (intermediate steps omitted)

$$\begin{aligned}
&\mathrm{A}(i_1) - \mathrm{A}(i_2) + \mathrm{A}(i_3) - \mathrm{A}(i_4) + \cdots + \mathrm{A}(|S_{\mathrm{A}}| - 1) - \mathrm{A}(|S_{\mathrm{A}}|) \\
&= x_{i_1} - x_{i_2} + x_{i_3} - x_{i_4} + \cdots + x_{i_{|S_{\mathrm{A}}|-1}} - x_{i_{|S_{\mathrm{A}}|}} \\
&\quad + \sum_{j:\sigma(i_2) \geq \sigma(j) > \sigma(i_1)} y_j + \sum_{j:\sigma(i_4) \geq \sigma(j) > \sigma(i_3)} y_j + \cdots + \sum_{j:\sigma(i_{|S_{\mathrm{A}}|}) \geq \sigma(j) > \sigma(i_{|S_{\mathrm{A}}|-1})} y_j.
\end{aligned}$$

By the definition of $i_1, i_2, \ldots, i_{|S_{\mathrm{A}}|}$, all of the intervals of the form $\{i' : \sigma(i_{j+1}) \geq \sigma(i') > \sigma(i_j)\}$ are disjoint and do not intersect, so all y_j's have coefficients of 0 or +1. Note that once we add the $\mathrm{B}(i)$ expressions for those $i \in S_{\mathrm{A}} \cap S_{\mathrm{B}}$ as described above (we add $\mathrm{B}(i_j)$ to R_+ if j is odd, and to R_- if j is even), this will only change whether the inequalities in each interval $\{i' : \sigma(i_{j+1}) \geq \sigma(i') > \sigma(i_j)\}$ are strict or non-strict. In the end, after we perform Step 1, all y_j's have a coefficient of 0 or +1.

Step 2. For $i \in S_{\mathrm{B}} \setminus S_{\mathrm{A}}$:

- If $i \in J$ (coefficient of y_j after Step 1 is +1), then add $\mathrm{B}(i)$ to R_-; otherwise,
- If $i \notin J$ (coefficient of y_j after Step 1 is 0), then add $\mathrm{B}(i)$ to R_+.

Observe that by definition of this step, each y_j still has a coefficient of either 0 or +1. Observe also that by taking this step, the coefficients of the x_i's remain in $\{0, +1, -1\}$. This is because the x_i's whose coefficients change in this step are disjoint from the x_i's whose coefficients changed in Step 1 – more specifically, $I_+, I_- \subseteq S_{\mathrm{A}}$, while the i's which are being set here are for those i's in $S_{\mathrm{B}} \setminus S_{\mathrm{A}}$.

If we evaluate $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ after this step, we will thus obtain an expression of the following form:

$$\begin{aligned}
&\sum_{i \in I_+} (+1) x_i + \sum_{i \in I_-} (-1) x_i + \sum_{j \in J \setminus (S_{\mathrm{B}} \setminus S_{\mathrm{A}})} (+1) y_j + \sum_{j \in (S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \setminus J} (+1) y_j \\
&+ \sum_{i \in (S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cap J} (+1) x_i + \sum_{i \in (S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cap J^C} (-1) x_i.
\end{aligned}$$

Step 3. For $i \in S_{\mathrm{C}}$:

- If $i \in I_+$, add $\mathrm{C}(i)$ to R_-.
- If $i \in I_-$, add $\mathrm{C}(i)$ to R_+.
- If $i \in (S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cap J$, add $\mathrm{C}(i)$ to R_-.
- If $i \in (S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cap J^C$, add $\mathrm{C}(i)$ to R_+.
- If i is not in any of the above sets, add $\mathrm{C}(i)$ to R_+.

In this step, we are adding the $\mathrm{C}(i)$ expressions in accordance with the current sign of x_i in the expression after Step 2, so as to ensure that every x_i's coefficient remains in $\{0, +1, -1\}$. After this step, if we evaluate $\sum_{e \in R_+} e - \sum_{e \in R_-} e$, we obtain

$$\begin{aligned}
&\sum_{i \in I_+ \setminus S_{\mathrm{C}}} (+1) x_i + \sum_{i \in I_- \setminus S_{\mathrm{C}}} (-1) x_i + \sum_{j \in J \setminus (S_{\mathrm{B}} \setminus S_{\mathrm{A}})} (+1) y_j + \sum_{j \in (S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \setminus J} (+1) y_j \\
&+ \sum_{i \in [(S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cap J] \setminus S_{\mathrm{C}}} (+1) x_i + \sum_{i \in [(S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cap J^C] \setminus S_{\mathrm{C}}} (-1) x_i + \sum_{i \in S_{\mathrm{C}} \setminus [I_+ \cup I_- \cup (S_{\mathrm{B}} \setminus S_{\mathrm{A}})]} (+1) x_i. \qquad \text{(EC.20)}
\end{aligned}$$

Step 4. Finally, we are left with assigning the expressions in S_{D}. This step has four possible cases:

- If S_{D} is empty, then we are left with expression (EC.20).
- If $S_{\mathrm{D}} = \{1, 2\}$, then assign both $\mathrm{D}(1)$ and $\mathrm{D}(2)$ to R_+. Because $\mathrm{D}(1)$ and $\mathrm{D}(2)$ involve the same variables and have opposite sign, they cancel out, and again we are left with expression (EC.20).

- If $S_{\mathrm{D}} = \{1\}$, then assign D(1) to R_-. Intuitively, this assignment is safe because all y_j's up to this point have a coefficient of +1 or 0 and we are only subtracting 1 from the coefficient of every y_j. As a result, our expression becomes

$$\sum_{i \in I_+ \setminus S_{\mathrm{C}}} (+1)x_i + \sum_{i \in I_- \setminus S_{\mathrm{C}}} (-1)x_i + \sum_{j \in (S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cap J} (-1)y_j + \sum_{j \in [(S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cup J]^C} (-1)y_j$$
$$+ \sum_{i \in [(S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cap J] \setminus S_{\mathrm{C}}} (+1)x_i + \sum_{i \in [(S_{\mathrm{B}} \setminus S_{\mathrm{A}}) \cap J^C] \setminus S_{\mathrm{C}}} (-1)x_i + \sum_{i \in S_{\mathrm{C}} \setminus [I_+ \cup I_- \cup (S_{\mathrm{B}} \setminus S_{\mathrm{A}})]} (+1)x_i. \tag{EC.21}$$

- If $S_{\mathrm{D}} = \{2\}$, then assign D(2) to R_+. Again, this assignment is safe, because all y_j's have a coefficient of +1 or 0, and we are only adding -1 to every y_j's coefficient. We again end up with expression (EC.21).

After completing Step 4, we are left with expression (EC.20) or (EC.21). In both of these expressions, all variables have coefficients in $\{0, +1, -1\}$; moreover, our procedure assigns all expressions to either R_+ and R_- and leaves no expression unassigned.

As a result, invoking Proposition EC.1, we have that the matrix $\mathbf{A}$ is totally unimodular. We now use the following classical result in integer optimization:

PROPOSITION EC.2 **(Theorem 3.1(b) of Bertsimas and Weismantel 2005)**. *Let $\mathbf{A} \in \mathbb{Z}^{m \times n}$ be an integer matrix. The matrix $\mathbf{A}$ is totally unimodular if and only the polyhedron $P(\mathbf{b}) = \{\mathbf{x} \in \mathbb{R}^n_+ \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ is integral for all $\mathbf{b} \in \mathbb{Z}^m$ for which $P(\mathbf{b}) \neq \emptyset$.*

In our context, the vector $\mathbf{b}$ defining the system of inequalities (EC.12) is indeed integer. Also, the feasible region $\mathcal{F}$ is nonempty; this will be assured by a later result, Proposition 6, which asserts that for any $\mathbf{x} \in \{0,1\}^n$ and any given ranking σ^k, there exists $\mathbf{y}^k \in \mathbb{R}^{n+1}$ with $\mathbf{y}^k \geq \mathbf{0}$ such that $(\mathbf{x}, \mathbf{y}^k)$ satisfy constraints (2b), (2c), (2d) and (2e). Thus, we can apply Proposition EC.2 to assert that $\mathcal{F}$ is integral. This concludes the proof. □

EC.1.6. Proof of Proposition 5

We consider each model separately. In each case, we assume that the constraint $\mathbf{C}\mathbf{x} \leq \mathbf{d}$ is removed (or equivalently, we set $\mathbf{C} = [\mathbf{0}^T], \mathbf{d} = [0]$, so as to make the constraint $\mathbf{C}\mathbf{x} \leq \mathbf{d}$ vacuous).

Problem (3) **(Belloni et al. 2008)**: For this formulation, consider $n = 5$ and σ defined as

$$\sigma(2) = 0,$$
$$\sigma(1) = 1,$$
$$\sigma(0) = 2,$$
$$\sigma(3) = 3,$$
$$\sigma(5) = 4,$$
$$\sigma(4) = 5.$$

It can be shown that $\mathbf{x} = (1/2, 1/2, 0, 0, 0)$, $\mathbf{y} = (1/2, 1/2, 0, 0, 0, 0)$ (where $\mathbf{y}$ is indexed from 0 to n) is an extreme point of $\mathcal{F}_{BFSS}$. Since this solution is fractional, $\mathcal{F}_{BFSS}$ cannot be integral in general.

Problem (4): For this formulation, consider $n = 5$ and σ defined as

$$\sigma(3) = 0,$$
$$\sigma(5) = 1,$$
$$\sigma(0) = 2,$$
$$\sigma(2) = 3,$$
$$\sigma(1) = 4,$$
$$\sigma(4) = 5.$$

For this ranking, consider the following utilities for the products:

$$\begin{aligned} u_0 &= 3, \\ u_1 &= 1, \\ u_2 &= 2, \\ u_3 &= 5, \\ u_4 &= 0, \\ u_5 &= 4. \end{aligned}$$

It can be shown that $\mathbf{x} = (1, 1, 1/3, 0, 3/4)$, $\mathbf{y} = (0, 0, 2/3, 1/3, 0, 0)$ is a fractional extreme point of $\mathcal{F}_{Utility1}$.

Problem (5) (McBride and Zufryden 1988): For this formulation, since Proposition 1 shows that $\mathcal{F}_{MZ} \supseteq \mathcal{F}_{Utility1}$, the same choice of σ and utilities $u_0, \ldots, u_n$ as for problem (4) can be used to establish that $\mathcal{F}_{MZ}$ is not integral in general.

EC.1.7. Proof of Proposition 6

We will prove the result by showing that, in fact, the proposed $\mathbf{y}^k$ is the *only* solution of problem (7). First, letting $i^* = \arg\min_{j \in S \cup \{0\}} \sigma^k(j)$, observe that by constraints (7d) and (7e), we have that for $j \in \{0, 1, \ldots, n\}$ with $\sigma^k(j) > \sigma^k(i^*)$, it must be that $y_j^k = 0$. Observe also that for $j \in \{1, \ldots, n\}$ with $\sigma^k(j) < \sigma^k(i^*)$, it must be that $y_j^k = 0$, because it must be that $x_j = 0$ (if x_j were 1, then i^* could not be the index that minimizes $\sigma^k(j')$ for $j' \in S \cup \{0\}$).

These two observations imply that $y_i^k = 0$ for $i \in \{0, 1, \ldots, n\} \setminus \{i^*\}$, and thus by constraint (7b), it must be that $y_{i^*}^k = 1$. Since this completely specifies $\mathbf{y}^k$, it must be the only solution to the problem and hence the optimal solution of the problem. $\square$

EC.1.8. Proof of Proposition 7

We will proceed in two steps: first, we will show that the proposed solution is feasible, and second, we will show that the objective value of the proposed solution is exactly π^*, which is the optimal objective of the primal problem.

Feasibility: Observe that both $\boldsymbol{\alpha}^k$ and $\boldsymbol{\beta}^k$ are nonnegative by their definition. Constraint (9c) is automatically satisfied because by their definition, γ^k and $\boldsymbol{\beta}^k$ are both nonnegative. Thus, we need only verify constraint (9b).

For $j \in S \setminus \{i^*\}$, we have

$$\begin{aligned} \gamma^k + \sum_{i\,:\,\sigma^k(i) < \sigma^k(j)} \beta_i^k &= \gamma^k + \beta_{i^*}^k \\ &= \pi^* + \max_{i' \in S} \pi_{i'} - \pi^* \\ &= \max_{i' \in S} \pi_{i'} \\ &\geq \pi_j \end{aligned}$$

where the first equality follows by definition of $\boldsymbol{\beta}^k$, the second by definition of $\boldsymbol{\beta}_{i^*}^k$ and γ^k, the third by simple algebra and the final inequality by the definition of the maximum. Since $\boldsymbol{\alpha}^k$ is nonnegative, it follows that

$$\gamma^k + \alpha_j^k + \sum_{i\,:\,\sigma^k(i) < \sigma^k(j)} \beta_i^k \geq \pi_j.$$

For $j \in \{1,\ldots,n\} \setminus S$, we have

$$\begin{aligned}\gamma^k + \alpha_j^k + \sum_{i\,:\,\sigma^k(i)<\sigma^k(j)} \beta_i^k &\geq \gamma^k + \left(\pi_j - \gamma - \sum_{i\,:\,\sigma^k(i)<\sigma^k(j)} \beta_i^k\right) + \sum_{i\,:\,\sigma^k(i)<\sigma^k(j)} \beta_i^k \\ &= \pi_j\end{aligned}$$

where the inequality follows by the definition of α_j^k via the maximum function. Thus, constraint (9b) holds for $j \in \{1,\ldots,n\} \setminus S$.

The above two cases fully verify constraint (9b) when $i^* = 0$. In the case that $i^* \neq 0$, we must check constraint (9b) for i^*. If $i^* \neq 0$, we have for i^* that

$$\begin{aligned}\gamma^k + \sum_{i\,:\,\sigma^k(i)<\sigma^k(i^*)} \beta_i^k &= \gamma^k \\ &= \pi^* \\ &= \pi_{i^*},\end{aligned}$$

where the first equality follows by definition of $\boldsymbol{\beta}$, the second by definition of γ^k and the third by definition of π^*. Since $\boldsymbol{\alpha} \geq \mathbf{0}$ by definition, constraint (9b) must hold for $j = i^*$. This concludes our proof of the feasibility of $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma)$.

Objective value: For $j \in S \setminus \{i^*\}$, we have that

$$\begin{aligned}\alpha_j^k &= \max\left\{\pi_j - \gamma^k - \sum_{i\,:\,\sigma^k(i)<\sigma^k(j)} \beta_i^k, 0\right\} \\ &= \max\{\pi_j - \max_{i' \in S} \pi_{i'}, 0\} \\ &= 0\end{aligned}$$

and if $i^* \neq 0$, then for i^* we have that

$$\begin{aligned}\alpha_{i^*}^k &= \max\left\{\pi_{i^*} - \gamma^k - \sum_{i\,:\,\sigma^k(i)<\sigma^k(j)} \beta_i^k, 0\right\} \\ &= \max\{\pi_{i^*} - \pi_{i^*} - 0, 0\} \\ &= 0.\end{aligned}$$

Thus, we have $\alpha_j^k = 0$ for any $j \in S$, or equivalently, any j with $x_j = 1$. Similarly, for $j \in \{1,\ldots,n\} \setminus S$ or equivalently, for any $j \in \{1,\ldots,n\}$ with $x_j = 0$, we have that $\beta_j^k = 0$. Thus, when we consider the objective value of the solution, we get

$$\begin{aligned}\gamma^k + \sum_{j=1}^n \alpha_j^k x_j + \sum_{i=1}^n \beta_i^k (1 - x_i) &= \pi^* + 0 + 0 \\ &= \pi^*\end{aligned}$$

which is exactly the optimal value of the primal problem. Thus, it follows that $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ is a dual optimal solution. $\square$

EC.1.9. Proof of Theorem 2

We proceed in three stages. First, we show that $\mathbf{y}$ is primal feasible. Second, we show that $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ is dual feasible. Finally, we show that the two solutions satisfy complementary slackness, which establishes that they are optimal.

Primal feasibility. It is clear from the structure of the algorithm that constraints (13d) and (13c) are never violated in the algorithm; in addition, at each stage of the algorithm, the solution $\mathbf{y}$ satisfies the inequality $\sum_{j=0}^{n} y_j \leq 1$. Note also that since these constraints are never violated, the slacks associated with these inequality constraints are never negative. Therefore, whenever $y_{\tau(s)}$ is set to q^*, it is never set to a negative value; y_j must therefore always be nonnegative, for all j.

We only need to verify that $\sum_{j=0}^{n} y_j = 1$, i.e., the sum of the y_j variables is exactly one upon termination. To see this, we proceed in two cases:

1. **Case 1**: $B_{main} \neq \emptyset$. If B_{main} is not empty, then let $i^{**} = \arg\min_{i \in B_{main}} \sigma(i)$, i.e., it is the option in B_{main} that has the lowest rank. After the algorithm terminates, we know that the preference constraint (13d) for i^{**} is satisfied at equality, that is:

$$\sum_{j:\sigma(j)>\sigma(i^{**})} y_j = 1 - x_{i^{**}}.$$

Given this, we now ask: what happens when the algorithm checks i^{**}? By our assumption on i^{**}, it cannot be that there is a B event when we check i^{**}. Therefore, either a C event happens, in which case we are done because we will set $y_{i^{**}}$ so as to reach the unit sum; or it is neither a C nor a B event, in which case it must be an A event. This latter case cannot happen (if i^{**} is checked before $f(i^{**})$, then when $f(i^{**})$ is checked there should be a C event; if i^{**} is checked after $f(i^{**})$, then there should also be a C event). Therefore, it must be the case $\sum_{j=0}^{n} y_j = 1$ upon termination.

2. **Case 2**: $B_{main} = \emptyset$. If B_{main} is empty, then at each stage of the algorithm there is either an A event or a C event. If there is a C event, we are done. Note that if a C event does not occur by stage $s = n$, then it must occur at stage $s = n$, because $\tau(n) = 0$ is the no-purchase option, $x_{\tau(n)}$ is 1, and $1 - \sum_{j=0}^{n} y_j$ at stage $s = n$ is at most 1.

Dual feasibility. Before we show that $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ is dual feasible, let us present two useful results regarding the dual phase of the algorithm:

Observation 1. In Algorithm 2, observe that when we sort $B_{main} = \{i_1, i_2, \ldots, i_{|B_{main}|}\}$ in increasing order of σ (i.e., $\sigma(i_1) < \sigma(i_2) < \cdots < \sigma(i_{|B_{main}|})$), then we also have that $\pi_{f(i_1)} \leq \pi_{f(i_2)} \leq \cdots \leq \pi_{f(i_{|B_{main}|})}$. The reason for this is that, by definition of the arg min in the clause for B events in Algorithm 1, we always take the one with the lowest value of σ. Therefore, as we progress through the algorithm, the options i^* that we add to B_{main} are such that their values of σ are decreasing. (They cannot be increasing, because of the way we have defined the arg min.) Since Algorithm 1 checks the options in decreasing order of π_i, it must be that $\pi_{f(i_1)} \leq \pi_{f(i_2)} \leq \cdots \leq \pi_{f(i_{|B_{main}|})}$.

Observation 2. By the definition of Algorithm 2, we have that

$$\gamma + \sum_{t'=1}^{t} \beta_{i_{t'}} = \pi_{f(i_t)}.$$

Having defined the two observations, let us now check dual feasibility. The dual constraints are:

$$\gamma + \alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j \geq \pi_i, \quad \forall i \in \{0, 1, \ldots, n\} \tag{EC.22}$$

$$\alpha_i \geq 0, \quad \forall i \in \{0, 1, \ldots, n\}, \tag{EC.23}$$

$$\beta_j \geq 0, \quad \forall j \in \{0, 1, \ldots, n\}. \tag{EC.24}$$

Let us begin by checking (EC.24). The set of i's breaks into two cases:

1. **Case 1**: $i \in B_{main}$. If $i \in B_{main}$, then $i = i_t$ for some $t \in \{1, \ldots, |B_{main}|\}$. For $t = 1$, we have

$$\beta_{i_1} = \pi_{f(i_1)} - \gamma,$$

which is nonnegative because $\gamma = \pi_C$, and option $f(i_1)$ must have been checked before option C. For $t > 1$, we have:

$$\begin{aligned}
\beta_{i_t} &= \pi_{f(i_t)} - \gamma - \sum_{t'=1}^{t-2} \beta_{i_{t'}} - \beta_{i_{t-1}} \\
&= \pi_{f(i_t)} - \gamma - \sum_{t'=1}^{t-2} \beta_{i_{t'}} - \left[\pi_{f(i_{t-1})} - \gamma - \sum_{t'=1}^{t-2} \beta_{i_{t'}} \right] \\
&= \pi_{f(i_t)} - \pi_{f(i_{t-1})},
\end{aligned}$$

which is nonnegative because option $f(i_t)$ is checked before option $f(i_{t-1})$, and so its profit must be at least that of $f(i_{t-1})$ (see Observation 1). We therefore have $\beta_i \geq 0$ for $i \in B_{main}$.

2. **Case 2**: $i \notin B_{main}$. By the way that β is initialized, $\beta_i = 0$ for these i's.

Having checked (EC.24), let us check (EC.23). For $i \notin A$, α_i is initialized to zero, so the condition is satisfied. For $i \in A$, we have that

$$\alpha_i = \pi_i - \gamma - \sum_{j:\sigma(j)<\sigma(i)} \beta_j,$$

and by the structure of the β_j's, we know that the last sum can be written as

$$\alpha_i = \pi_i - \gamma - \sum_{t'=1}^{t} \beta_{i_{t'}},$$

for some $t \in \{1, \ldots, |B_{main}|\}$. Note that by Observation 2, we have that

$$\gamma + \sum_{t'=1}^{t} \beta_{i_{t'}} = \pi_{f(i_t)}.$$

Therefore, α_i simplifies to

$$\alpha_i = \pi_i - \pi_{f(i_t)}.$$

So now the question is whether $f(i_t)$ was checked after i or not (i.e., is $\pi_i \geq \pi_{f(i_t)}$). If $f(i_t)$ was checked before i, then we know that the constraint $\sum_{j:\sigma(j)>\sigma(i_t)} y_j \leq 1 - x_{i_t}$ became tight. But recall that it is also the case that $\sigma(i_t) < \sigma(i)$. Therefore, if $f(i_t)$ were checked before i, the aforementioned constraint becoming tight would mean that when checking i, q_2 would have been equal to q^* (a B event would have occurred) and Algorithm 1 would *not* have added i to A. Therefore, it must be that $\pi_i - \pi_{f(i_t)} \geq 0$, so that $\alpha_i \geq 0$.

Having checked (EC.23), let us now check the last dual constraint (EC.22). We can break the set of i's into four mutually exclusive and collectively exhaustive cases.

1. **Case 1**: i's for which $\pi_i \leq \pi_C$. In this case, we have

$$\begin{aligned}
\gamma + \alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j &\geq \gamma + 0 + 0 \\
&= \pi_C \\
&\geq \pi_i,
\end{aligned}$$

where the first inequality holds by the non-negativity of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ verified above. Thus, the constraint holds for this case.

2. **Case 2**: The set of i's for which a B event occurs ($q^* = q_2$) and $i = f(i_t)$ for some $i_t \in B_{main}$. In this case, we can use Observation 2 to assert that

$$\begin{aligned}\gamma + \alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j &= \gamma + 0 + \sum_{t'=1}^{t} \beta_{i_{t'}} \\ &= \pi_{f(i_t)},\end{aligned}$$

which clearly satisfies the constraint.

3. **Case 3**: The set of i's for which a B event occurs ($q^* = q_2$), but $i \neq f(i_t)$ for all $i_t \in B_{main}$. Let $i_t \in B_{main}$ be the minimizer in the B event clause in Algorithm 1 when i is checked. Specifically, we have:

$$i_t = \arg \min_{p:\sigma(p)<\sigma(i)} \left\{ 1 - x_p - \sum_{j:\sigma(j)>\sigma(p)} y_j \right\}.$$

Since $i \neq f(i_t)$ for any $i_t \in B_{main}$, this means that i_t was already added to B_{main} in an earlier iteration of the Algorithm 1. Therefore, $\pi_{f(i_t)} \geq \pi_i$. Also, observe that $\sigma(i_1) < \cdots < \sigma(i_t) < \sigma(i)$, by the definition of i_t above. Therefore, we have

$$\begin{aligned}\gamma + \alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j &\geq \gamma + \sum_{t'=1}^{t} \beta_{i_{t'}} \\ &= \pi_{f(i_t)} \\ &\geq \pi_i,\end{aligned}$$

where the first inequality follows because $\sigma(i_t) < \sigma(i)$; $i_1, \ldots, i_t$ are a subset of the sum condition on the left-hand expression. This establishes the constraint.

4. **Case 4**: $i \in A$. By the definition of α_i, we have

$$\begin{aligned}\gamma + \alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j &= \gamma + (\pi_i - \gamma - \sum_{j:\sigma(j)<\sigma(i)} \beta_j) + \sum_{j:\sigma(j)<\sigma(i)} \beta_j \\ &= \pi_i,\end{aligned}$$

which verifies the constraint.

Complementary slackness. The complementary slackness conditions are:

$$\gamma \cdot (1 - \sum_{j=0}^{n} y_j) = 0 \tag{EC.25}$$

$$\beta_i \cdot (1 - x_i - \sum_{j:\sigma(j)>\sigma(i)} y_j) = 0 \tag{EC.26}$$

$$\alpha_i \cdot (x_i - y_i) = 0 \tag{EC.27}$$

$$y_i \cdot (\gamma + \alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j - \pi_i) = 0. \tag{EC.28}$$

Equation (EC.25) is automatically satisfied, because the y solution produced by our algorithm is feasible and satisfies the unit sum condition.

To see that equation (EC.26) holds, observe that for $i \notin B_{main}$ produced by the algorithm, $\beta_i = 0$ and the condition holds. Therefore, we only need to check $i \in B_{main}$. For $i \in B_{main}$, observe that i is only added to B_{main} by the algorithm whenever the inequality $\sum_{j:\sigma(j)>\sigma(i)} y_j \leq 1 - x_i$ becomes

tight the very first time. Therefore, for $i \in B_{main}$, the y produced by our algorithm will satisfy $(1 - x_i - \sum_{j:\sigma(j)>\sigma(i)} y_j) = 0$, and the condition holds.

To see that equation (EC.27) holds, observe that for $i \notin A$, α_i by default is set to zero and the condition holds. Therefore, we only need to check $i \in A$. For $i \in A$, observe that i is only added to A when the inequality $y_i \leq x_i$ becomes tight. Therefore, for $i \in A$, the y produced by our algorithm will satisfy $x_i - y_i = 0$, and thus the condition will hold.

Finally, to see that equation (EC.28) holds, we proceed carefully in several steps. First, observe that in Algorithm 1, if a C event occurs, then $C = \tau(s)$, and the loop is terminated. For i with $\pi_i \leq \pi_C$, observe that $y_i = 0$ (since the loop was terminated at option C, and all y_j's for j's with lower profit than C were initialized to zero).

This leaves i's for which $\pi_i > \pi_C$. This remaining set of i's can be partitioned into three mutually exclusive and collectively exhaustive cases, which we now treat.

1. **Case 1**: A B event occurred for i ($q^* = q_2$) and $i = f(i_t)$ for some $i_t \in B_{main}$. This is the case when we first encounter the option i_t as the arg min of expression q_2. In this case, by the way that we have specified the dual solution, we have that

$$\begin{aligned}\gamma + \alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j - \pi_i &= \gamma + \sum_{t'=1}^{t} \beta_{i_{t'}} - \pi_i \\ &= \pi_i - \pi_i \\ &= 0,\end{aligned}$$

and thus equation (EC.28) must hold.

2. **Case 2**: A B event occurred for i ($q^* = q_2$) and $i \neq f(i_t)$ for all $i_t \in B_{main}$. This is the case when $q^* = q_2$, but the i_t which is the arg min that leads to q_2 has already been encountered. Since i_t has already been encountered, it is the case that $\pi_{f(i_t)} \geq \pi_i$. This is the case because Algorithm 1 scans through the options in decreasing order of profit.

Since i_t was already added to B_{main} by Algorithm 1 when option $f(i^*)$ was tested, the constraint $\sum_{j:\sigma(j)>\sigma(i_t)} y_j \leq 1 - x_{i_t}$ must have become binding after option $f(i^*)$. Therefore y_i must have been set to zero. Since y_i is zero, equation (EC.28) must hold.

3. **Case 3**: $i \in A$. By the definition of α_i, we have that:

$$\begin{aligned}\gamma + \alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j - \pi_i &= \gamma + \left[\pi_i - \gamma - \sum_{j:\sigma(j)<\sigma(i)} \beta_j\right] + \sum_{j:\sigma(j)<\sigma(i)} \beta_j - \pi_i \\ &= 0,\end{aligned}$$

so that again, equation (EC.28) holds.

Since we have established that the two solutions are feasible for their respective problems and satisfy complementary slackness, this concludes the proof. □

EC.2. Additional theoretical results for Benders decomposition

EC.2.1. Solving the integer Benders problem using classical constraint generation

In this section, we present a constraint generation procedure for solving the Benders problem (10). For the purpose of defining our constraint generation procedure, let us suppose that for each customer type k, we have a set of dual solutions $\bar{A}_k \subseteq A_k$ that have been generated so far. We define the restricted master problem as

$$\underset{\mathbf{x},\mathbf{t}}{\text{maximize}} \quad \sum_{k=1}^{K} \lambda^k t_k \tag{EC.29a}$$

$$\text{subject to} \quad t_k \leq \gamma^k + \sum_{i=1}^{n} \alpha_i^k \cdot x_i + \sum_{i=1}^{n} \beta_i^k \cdot (1 - x_i), \quad \forall\ k \in \{1, \ldots, K\},\ (\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k) \in \bar{A}_k, \tag{EC.29b}$$

$$\mathbf{C}\mathbf{x} \leq \mathbf{d}, \tag{EC.29c}$$

$$x_i \in \{0, 1\}, \quad \forall\ i \in \{1, \ldots, n\}. \tag{EC.29d}$$

We define our classical constraint generation procedure as Algorithm 3.

Algorithm 3 Classical constraint generation algorithm for solving integer Benders formulation

Set $\bar{A}_k = \emptyset$ for all customer types $k \in \{1, \ldots, K\}$.
Solve the restricted master problem (EC.29) to obtain a solution $(\mathbf{x}, \mathbf{t})$.
For each $k \in \{1, \ldots, K\}$:
  Determine primal subproblem solution $\mathbf{y}^k$ using equation (8) and dual subproblem solution $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ using equations (11a) – (11c).
while $\max_{k=1,\ldots,K}(t_k - \boldsymbol{\pi}^T \mathbf{y}^k) > 0$ **do**
  For each $k \in \{1, \ldots, K\}$ with $\boldsymbol{\pi}^T \mathbf{y}^k < t_k$:
    Set $\bar{A}_k \leftarrow \bar{A}_k \cup \{(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)\}$.
  Solve restricted master problem (EC.29) with $\bar{A}_1, \ldots, \bar{A}_K$ to obtain $(\mathbf{x}, \mathbf{t})$.
  For each $k \in \{1, \ldots, K\}$:
    Determine primal subproblem solution $\mathbf{y}^k$ using equation (8) and dual subproblem solution $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ using equations (11a) – (11c).
end while
return Optimal solution $(\mathbf{x}, \mathbf{t})$ of problem (10).

A standard result for constraint generation algorithms is that the solution returned by the algorithm is optimal. For completeness, we prove below that the solution produced by Algorithm 3 is in fact optimal.

Proposition EC.3. *Let* $(\mathbf{x}, \mathbf{t})$ *be the solution obtained upon termination of Algorithm 3. Then* $(\mathbf{x}, \mathbf{t})$ *is an optimal solution of problem* (10).

Proof of Proposition EC.3: To prove this, let $(\mathbf{x}^*, \mathbf{t}^*)$ be an optimal solution of problem (10) and let $Z_1 = \sum_{k=1}^{K} \lambda^k t_k^*$ be the optimal objective value of the master solution. Let $Z_2 = \sum_{k=1}^{K} \lambda^k t_k$ be the objective value of the solution generated by Algorithm 3.

Upon termination of Algorithm 3, $(\mathbf{x}, \mathbf{t})$ is the optimal solution of the restricted master problem with constraints (EC.29b) enforced at the sets $\bar{A}_1, \ldots, \bar{A}_K$ of dual subproblem solutions that were generated over the execution of Algorithm 3. Since $\bar{A}_k \subseteq A_k$ (the set of generated dual solutions for a given customer type k is a subset of *all* dual feasible solutions for a customer type k), then $(\mathbf{x}^*, \mathbf{t}^*)$ must be a feasible solution for problem (EC.29). Since problems (10) and (EC.29) share the same objective function in terms of $\mathbf{x}$ and $\mathbf{t}$, and since $(\mathbf{x}, \mathbf{t})$ is an optimal solution of (EC.29) it follows that $Z_1 \leq Z_2$.

Now, upon termination of the algorithm, we have that

$$\max_{k=1,\ldots,K}(t_k - \boldsymbol{\pi}^T \mathbf{y}^k) \leq 0,$$

which is equivalent to

$$t_k \leq \boldsymbol{\pi}^T \mathbf{y}^k, \quad \forall\ k \in \{1, \ldots, K\},$$

where $\mathbf{y}^k$ is the solution obtained from equation (8). By Proposition 6, each $\mathbf{y}^k$ is guaranteed to be an optimal solution of the primal subproblem for customer type k at the current solution $\mathbf{x}$.

Similarly, by Proposition 7, each $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ is guaranteed to be an optimal solution of the dual subproblem for customer type k at $\mathbf{x}$. Therefore, by strong duality, the primal subproblem objective value $\boldsymbol{\pi}^T\mathbf{y}^k$ is equal to the dual subproblem objective value $\gamma^k + \sum_{i=1}^n \alpha_i^k x_i + \sum_{j=0}^n \beta_j^k(1-x_j)$. We thus have:

$$t_k \leq \gamma^k + \sum_{i=1}^n \alpha_i^k x_i + \sum_{j=1}^n \beta_j^k (1-x_j), \quad \forall\ k \in \{1,\ldots,K\},$$

Since $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ is the optimal dual subproblem solution at $\mathbf{x}$, the above is equivalent to

$$t_k \leq \min\left\{ \bar{\gamma}^k + \sum_{i=1}^n \bar{\alpha}_i^k x_i + \sum_{j=1}^n \bar{\beta}_j (1-x_j) \;\middle|\; (\bar{\boldsymbol{\alpha}}^k, \bar{\boldsymbol{\beta}}^k, \bar{\gamma}^k) \in A_k \right\}, \quad \forall\ k \in \{1,\ldots,K\}.$$

This last inequality is exactly equivalent to

$$t_k \leq \bar{\gamma}^k + \sum_{i=1}^n \bar{\alpha}_i^k x_i + \sum_{j=1}^n \bar{\beta}_j^k (1-x_j), \quad \forall\ (\bar{\boldsymbol{\alpha}}^k, \bar{\boldsymbol{\beta}}^k, \bar{\gamma}^k) \in A_k,\ k \in \{1,\ldots,K\},$$

which is exactly constraint (10b). Thus, $(\mathbf{x}, \mathbf{t})$ must be a feasible solution of the master problem (10). Since problems (10) and (EC.29) share the same objective function, we must have that $Z_2 = \sum_{k=1}^K \lambda^k t_k \leq Z_1$.

Since we have established that $Z_1 \leq Z_2$ and $Z_1 \geq Z_2$, it must be that $Z_1 = Z_2$, and that $(\mathbf{x}, \mathbf{t})$ attains the optimal objective value in problem (10). Since $(\mathbf{x}, \mathbf{t})$ is a feasible solution of problem (10), it thus follows that it is optimal. $\square$

EC.2.2. Solving the LO relaxation of the Benders problem using classical constraint generation

Analogously to the integer problem, we can solve the LO relaxation of the Benders formulation using a classical constraint generation approach. We define the restricted master problem as:

$$\underset{\mathbf{x},\mathbf{t}}{\text{maximize}} \quad \sum_{k=1}^K \lambda^k t_k \tag{EC.30a}$$

$$\text{subject to} \quad t_k \leq \gamma^k + \sum_{i=1}^n \alpha_i^k \cdot x_i + \sum_{i=1}^n \beta_i^k \cdot (1-x_i), \quad \forall\ k \in \{1,\ldots,K\},\ (\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k) \in \bar{A}_k, \tag{EC.30b}$$

$$\mathbf{C}\mathbf{x} \leq \mathbf{d}, \tag{EC.30c}$$

$$0 \leq x_i \leq 1, \quad \forall\ i \in \{1,\ldots,n\}. \tag{EC.30d}$$

Algorithm 4 provides the classical constraint generation procedure for solving the LO relaxation of the Benders problem. This algorithm is almost exactly the same as Algorithm 3; the only difference is that instead of problem (EC.29), we solve problem (EC.30), and instead of using equations (11a) – (11c) to find the primal and dual subproblem solutions, we instead apply Algorithms 1 and 2.

Like Algorithm 3, Algorithm 4 is guaranteed to obtain the optimal solution of problem (12). We formalize this as the proposition below; we omit the proof, since it is almost identical to that of Proposition EC.3.

PROPOSITION EC.4. *Let* $(\mathbf{x}, \mathbf{t})$ *be the solution obtained upon termination of Algorithm 4. Then* $(\mathbf{x}, \mathbf{t})$ *is an optimal solution of problem* (12).

Algorithm 4 Classical constraint generation algorithm for solving relaxation of Benders formulation

Set $\bar{A}_k = \emptyset$ for all customer types $k \in \{1, \ldots, K\}$.
Solve the restricted master problem (EC.30) to obtain a solution $(\mathbf{x}, \mathbf{t})$.
For each $k \in \{1, \ldots, K\}$:
  Run Algorithms 1 and 2 with $\mathbf{x}$ to obtain a primal solution $\mathbf{y}^k$ and a dual solution $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$.
while $\max_{k=1,\ldots,K}(t_k - \boldsymbol{\pi}^T\mathbf{y}^k) > 0$ **do**
  For each $k \in \{1, \ldots, K\}$ with $\boldsymbol{\pi}^T\mathbf{y}^k < t_k$:
    Set $\bar{A}_k \leftarrow \bar{A}_k \cup \{(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)\}$.
  Solve restricted master problem (EC.30) with $\bar{A}_1, \ldots, \bar{A}_K$ to obtain $(\mathbf{x}, \mathbf{t})$.
  For each $k \in \{1, \ldots, K\}$:
    Run Algorithms 1 and 2 with $\mathbf{x}$ to obtain a primal solution $\mathbf{y}^k$ and a dual solution $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$.
end while
return Optimal solution $(\mathbf{x}, \mathbf{t})$ of problem (10).

EC.2.3. Finite convergence of constraint generation

Algorithms 3 and 4 are guaranteed to provide optimal solutions upon termination to the integer problem and the LO relaxation problem, respectively. In this section, we establish that both algorithms are guaranteed to terminate in finitely many iterations. The first result that we establish is that the primal and dual subproblem solutions produced by Algorithms 1 and 2 are guaranteed to be extreme points of their respective subproblems. As with our definitions of Algorithms 1 and 2, we develop the result in terms of a ranking σ and drop the index k to lighten notation.

THEOREM EC.1. *Let $\sigma : \{0, 1, \ldots, n\} \to \{0, 1, \ldots, n\}$ and $\mathbf{x} \in [0, 1]^n$. Let $\mathbf{y}$ be the solution produced by Algorithm 1, and $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ be the solution produced by Algorithm 2. Then $\mathbf{y}$ is an extreme point of the primal problem* (13) *and $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ is an extreme point of the dual problem* (14).

Proof of Theorem EC.1: We prove the result separately for the primal and dual solutions.

Primal solution is an extreme point: To prove this, we will proceed directly from the definition of an extreme point. A point $\mathbf{z}$ in a polyhedron P is said to be an extreme point if there do *not* exist two points, $\mathbf{z}^1, \mathbf{z}^2 \neq \mathbf{z}$ and a real number $\theta \in (0, 1)$ such that $\mathbf{z} = \theta\mathbf{z}^1 + (1 - \theta)\mathbf{z}^2$.

We will prove this by contradiction. Let us suppose that $\mathbf{y}$ is not an extreme point. Then there exist $\mathbf{y}^1, \mathbf{y}^2 \neq \mathbf{y}$ and a scalar $\theta \in (0, 1)$ such that $\mathbf{y} = \theta\mathbf{y}^1 + (1 - \theta)\mathbf{y}^2$. Let τ be the same ordering used in Algorithm 1. Define the index s^* as

$$s^* = \min\{s \in \{0, 1, \ldots, n\} \mid y^1_{\tau(s^*)} \neq y_{\tau(s^*)} \text{ or } y^2_{\tau(s^*)} \neq y_{\tau(s^*)}\},$$

i.e., it is the first stage of Algorithm 1 at which a coordinate of $\mathbf{y}$ differs from $\mathbf{y}^1$ or $\mathbf{y}^2$. Note that the set defining the minimum cannot be empty, because otherwise both $\mathbf{y}^1$ and $\mathbf{y}^2$ would be equal to $\mathbf{y}$. At s^*, we must have that both $y^1_{\tau(s^*)} \neq y_{\tau(s^*)}$ and $y^2_{\tau(s^*)} \neq y_{\tau(s^*)}$ (both must be different from $y_{\tau(s^*)}$, because if only one is distinct, then their convex combination could not be equal to $y_{\tau(s^*)}$). Note also that s^* cannot happen after the C event in Algorithm 1 (if s^* is after the C event, then $y_{\tau(s^*)} = 0$, and since $\mathbf{y}^1$ and $\mathbf{y}^2$ must be nonnegative, this would again imply that both $\mathbf{y}^1$ and $\mathbf{y}^2$ would have to be equal to $\mathbf{y}$).

Without loss of generality, let us assume that $y^1_{\tau(s^*)} < y_{\tau(s^*)} < y^2_{\tau(s^*)}$. We now argue that $\mathbf{y}^2$ cannot be feasible. Observe that, as we proceed from $s = 0$ to $s = s^* - 1$, the coordinates of $\mathbf{y}$ and $\mathbf{y}^2$ are set the same way. Algorithm 1 always sets each coordinate to the largest possible value it can be set that maintains the feasibility of all constraints. Thus, at $s = s^*$, $y_{\tau(s^*)}$ is set to the largest feasible value based on the current values of all the variables; if $y^2_{\tau(s^*)} > y_{\tau(s^*)}$, then that implies that $\mathbf{y}^2$ cannot be feasible:

- If an A event occurred, then this would mean that $y^2_{\tau(s^*)} > x_{\tau(s^*)}$;
- If a C event occurred, then this would mean that $\sum_{s=0}^{s^*} y^2_{\tau(s^*)} > \sum_{s=0}^{s^*} y_{\tau(s^*)} = 1$, which implies that $\sum_{j=0}^{n} y_j^2 > 1$; and
- If a B event occurred, with i^* as the corresponding preference inequality that became tight, then this would mean that

$$1 - x_{i^*} = \sum_{\substack{s=0: \\ \sigma(\tau(s))<\sigma(i^*)}}^{s^*} y_{\tau(s)} < \sum_{\substack{s=0: \\ \sigma(\tau(s))<\sigma(i^*)}}^{s^*} y^2_{\tau(s)},$$

which implies that $\sum_{j:\sigma(j)<\sigma(i^*)} y_j^2 > 1 - x_{i^*}$.

We thus have a contradiction, and it must be that $\mathbf{y}$ is an extreme point of the primal problem.

Dual solution is an extreme point: To prove this, we will use the equivalence between extreme points and basic feasible solutions (see Theorem 2.3 of Bertsimas and Tsitsiklis 1997), and show that $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ is a basic feasible solution. A feasible solution $\mathbf{z}$ of a polyhedron $P = \{\mathbf{z} \in \mathbb{R}^m \mid \mathbf{A}\mathbf{z} \leq \mathbf{b}\}$ is a basic feasible solution if there are m linearly independent active constraints at $\mathbf{z}$.

Consider the following system of equations:

$$\gamma + \alpha_C + \sum_{j:\sigma(j)<\sigma(C)} \beta_j = \pi_C, \tag{EC.31}$$

$$\gamma + \alpha_{f(i_t)} + \sum_{j:\sigma(j)<\sigma(f(i_t))} \beta_j = \pi_{f(i_t)}, \quad \forall\ t \in \{1, \ldots, |B_{main}|\}, \tag{EC.32}$$

$$\gamma + \alpha_i + \sum_{j:\sigma(j)<\sigma(i)} \beta_j = \pi_i, \quad \forall\ i \in A, \tag{EC.33}$$

$$\alpha_i = 0, \quad \forall i\ \in \{0, 1, \ldots, n\} \setminus A, \tag{EC.34}$$

$$\beta_j = 0, \quad \forall j\ \in \{0, 1, \ldots, n\} \setminus \{i_1, \ldots, i_{|B_{main}|}\}. \tag{EC.35}$$

We note that there are $1 + |B_{main}| + |A| + (n + 1 - |A|) + (n + 1 - |B_{main}|) = 2n + 3$ equations, which is exactly the number of variables in the dual problem. These equations are constraints from problem (14) that are made to hold at equality. We now show that this system of equations implies a unique solution, establishing that the system of equations is linearly independent, and we show that the solution produced by our algorithm coincides with this solution.

First, observe that equations (EC.34) imply that $\alpha_i = 0$ for all $i \notin A$. This is also true at the end of Algorithm 2, because all α_i's are initially set to zero, and the only ones that are potentially changed from zero are those with $i \in A$. Similarly, equations (EC.34) – (EC.35) imply that $\beta_j = 0$ for any $j \notin B_{main}$, which is also true at the end of Algorithm 2.

Second, let us establish that γ must be equal to π_C. To do so, we observe the following:

- In Algorithm 1, suppose a C event occurs at stage s. Then the corresponding q^* must satisfy $q^* > 0$. If this is not the case, then $q^* = 0$, which would mean that the equality $\sum_{j=0}^{n} y_j = 1$ became true before the current stage s. However, a C event would then have to have been triggered before stage s, which cannot be true.
- Using the above observation, we now show that for any $i_t \in B_{main}$, we must have $\sigma(i_t) \geq \sigma(C)$. To see this, suppose that there is an $i_t \in B_{main}$ such that $\sigma(i_t) < \sigma(C)$. This would imply that the preference inequality for option i_t became tight before the C event was triggered. However, since the preference inequality became tight and C is less preferred than i_t, this would mean that q^* would have to be zero in the iteration in which the C event is triggered, which is not possible.

These observations, together with the fact that $\beta_j = 0$ for any $j \notin \{i_1, \ldots, i_{|B_{main}|}\}$ (this is a consequence of equations (EC.35)) and the fact that $\alpha_C = 0$ (this follows from (EC.34) and the

fact that $C \notin A$, since at most one type of event can occur at each stage), equation (EC.31) implies that:

$$\begin{aligned}\gamma + \alpha_C + \sum_{j:\sigma(j)<\sigma(C)} \beta_j &= \gamma + 0 + 0 \\ &= \gamma \\ &= \pi_C.\end{aligned}$$

At the end of Algorithm 2, γ is also set to π_C.

Third, we handle $\beta_{i_1}, \ldots, \beta_{i_{|B_{main}|}}$. Observe that by using equations (EC.34) and (EC.35), together with the fact that $f(i_1), f(i_2), \ldots, f(i_{|B_{main}|}) \notin A$ (this is true because at most one type of event can occur at each stage), we can simplify equations (EC.32) to

$$\begin{aligned}\gamma + \beta_{i_1} &= \pi_{f(i_1)} \\ \gamma + \beta_{i_1} + \beta_{i_2} &= \pi_{f(i_2)} \\ \gamma + \beta_{i_1} + \beta_{i_2} + \beta_{i_3} &= \pi_{f(i_3)} \\ &\vdots \\ \gamma + \beta_{i_1} + \beta_{i_2} + \cdots + \beta_{i_{|B_{main}|}} &= \pi_{f(i_{|B_{main}|})}\end{aligned}$$

Notice that the unique solution to this system of equations is exactly given by the first loop of Algorithm 2. Since our algorithm also sets γ to π_C, it follows that any solution to (EC.31) – (EC.35) must match the solution created by our algorithm for $\beta_{i_1}, \ldots, \beta_{i_{|B_{main}|}}$.

Finally, we handle α_i for $i \in A$. The corresponding equations are given by (EC.33), which uniquely determine α_i for each $i \in A$ to be

$$\alpha_i = \pi_i - \gamma - \sum_{j:\sigma(j)<\sigma(i)} \beta_j.$$

These are exactly the same values set by the second loop of Algorithm 2.

We have thus established that the unique solution to (EC.31) – (EC.35) must exactly coincide with the solution produced by our algorithm. This establishes that the solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ of Algorithm 2 is a basic feasible solution or equivalently, an extreme point of the dual problem. □

A corollary of this theorem is that the primal and dual subproblem solutions for the integer problem, which are defined in equations (8) and equations (11a) – (11c) respectively, are also extreme points.

COROLLARY EC.1. *Let* $\mathbf{x} \in \{0,1\}^n$ *and* $k \in \{1, \ldots, K\}$. *Let* $\mathbf{y}$ *be the primal subproblem solution specified by equation* (8) *and* $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ *be the dual subproblem solution specified by equations* (11a) – (11c). *Then* $\mathbf{y}$ *is an extreme point of the primal problem* (13) *and* $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ *is an extreme point of the dual problem* (14).

Proof of Corollary EC.1: It can be shown that the primal solution specified equations (8) and the dual solution specified by equations (11a) – (11c) coincide with the primal and dual solutions produced by Algorithms 1 and 2, respectively, when $\mathbf{x}$ is binary. The result then follows from Theorem EC.1. □

With these results, we are now ready to prove that both constraint generation procedures terminate in finitely many iterations.

PROPOSITION EC.5. *Algorithms 3 and 4 terminate in finitely many iterations.*

Proof of Proposition EC.5: Let us consider Algorithm 4 for the LO relaxation; the proof for Algorithm 3 follows in the same way. By Algorithm 4, we generate at most one dual subproblem solution $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ corresponding to a violated constraint for each customer type k. For a given k, once the subproblem solution $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ is added to $\bar{A}_k$, any solution of the restricted master problem in subsequent iterations must satisfy constraint (10b) at $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$; thus, the solution $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$ will not be generated again in subsequent iterations. This means that for a given k, each iteration of Algorithm 4 generates a distinct solution $(\boldsymbol{\alpha}^k, \boldsymbol{\beta}^k, \gamma^k)$. By Theorem EC.1, this implies that for a given k, each iteration generates a distinct extreme point of the polyhedron A_k. By standard results in linear optimization theory (see Corollary 2.1 of Bertsimas and Tsitsiklis 1997), the polyhedron A_k of feasible dual subproblem solutions for customer type k has finitely many extreme points. Since there are finitely many customer types k, Algorithm 4 must terminate in finitely many iterations.

The proof for Algorithm 3 proceeds in the same way, with the difference that Corollary EC.1 guarantees that each dual subproblem solution is an extreme point. $\square$

EC.3. Additional results

EC.3.1. Formulation comparisons with Toubia et al. (2003) data

In this section, we report on additional experiments to compare the solvability and strength of the different formulations, using instances derived from the Toubia et al. (2003) data. For these experiments, we tested $n \in \{20, 50, 100\}$ and $K = 100$. For each (n, K), we created 20 instances as follows. We sampled n products randomly without replacement from the full set of 3584 products (as described in Section 5.3), and sampled K customers without replacement from the full set of 330 customers. The utilities for this random subset of products and the no-purchase option for this random subset of customers were calculated in exactly the same way as in Section 5.3. The marginal profit of each product was also computed the same way. For each instance, we tested the constrained formulation with the constraint $\sum_i x_i \leq b$ for $b \in \{5, 10\}$, as well as the unconstrained formulation.

Table EC.1 compares the average integrality gap of the relaxations of the four MIO formulations of the first-choice PLD problem, while Table EC.2 compares the average solution time (for full optimality) for the four MIO formulations. From these tables, we observe the same behavior as with the synthetic instances, namely that our formulation (problem (2)) produces the tightest LO relaxation bounds and requires the least amount of time to solve to full optimality.

Table EC.1 Comparison of LO bounds for instances derived from Toubia et al. (2003) data.

n	K	b	G_{BM}	G_{BFSS}	$G_{Utility}$	G_{MZ}
20	100	5	2.71	7.15	15.83	23.23
20	100	10	2.10	7.36	14.68	21.81
20	100	–	2.10	7.36	14.68	21.81
50	100	5	3.44	7.51	21.63	40.94
50	100	10	3.01	8.96	18.75	33.72
50	100	–	2.43	9.29	17.90	32.32
100	100	5	3.69	6.94	20.22	35.47
100	100	10	3.28	7.99	16.24	27.36
100	100	–	2.69	8.51	14.76	24.95

Table EC.2 Comparison of MIO solution times for instances derived from Toubia et al. (2003) data.

n	K	b	T_{BM}	T_{BFSS}	$T_{Utility}$	T_{MZ}
20	100	5	0.40	0.70	1.72	1.40
20	100	10	0.37	0.70	1.22	1.22
20	100	–	0.37	0.76	1.14	1.21
50	100	5	1.44	2.33	11.38	7.95
50	100	10	1.37	3.66	13.91	7.56
50	100	–	1.04	3.36	8.88	6.13
100	100	5	5.12	9.39	88.01	119.27
100	100	10	5.51	18.58	147.75	146.73
100	100	–	3.79	39.65	215.83	126.19

EC.3.2. Additional Benders results without warm-starting

In this set of additional experiments, we test out our Benders method from Section 4 (the LO relaxation phase and the integer phase) on the same real PLD instance from Section 5.3. The method was tested in exactly the same way as in Section 5.3, except that we do not run the divide and conquer algorithm before the Benders method, and so we do not warm-start the Benders method with an initial integer solution. Table EC.3 shows the results without this warm-starting. From this table, we can see that for the smaller instances, the time required to prove optimality is in general larger – for example, for $\sum x_i = 5$, the warm-started approach from Section 5.3 could solve the problem in around 10 minutes, whereas without warm-starting, it takes about 17 minutes. Similarly, for the larger instances, the final optimality gap is larger without warm-starting than it is with warm-starting. For example, for $\sum x_i = 50$, we obtain a final optimality gap of 3.09% with warm-starting, whereas without it, we obtain a gap of 5.73%. The main takeaway from this experiment is that providing a high-quality integer solution to the Benders method can be very beneficial, allowing for the problem to be solved more quickly (in smaller instances) or allowing a better final optimality gap (in larger instances).

Table EC.3 Results of large-scale Benders experiment without warm-starting with D&C solution.

	LO Benders phase		MIO Benders phase			Final
Constraints	Bound	Time (s)	Obj.	Bound	Time (s)	Gap (%)
$\sum x_i = 2$	59.22	175.59	59.22	59.22	54.26	0.00
$\sum x_i = 3$	66.57	268.96	66.29	66.29	109.02	0.00
$\sum x_i = 4$	71.21	325.45	70.24	70.24	580.42	0.00
$\sum x_i = 5$	73.85	392.79	72.82	72.82	650.01	0.00
$\sum x_i = 10$	79.36	553.34	77.41	77.41	14934.88	0.01
$\sum x_i = 20$	82.65	914.75	77.18	81.81	21617.32	5.67
$\sum x_i = 50$	85.01	2572.22	79.64	84.48	21600.09	5.73