Assortment Optimization under the Decision Forest Model

Yi-Chun Chen
UCLA Anderson School of Management, University of California, Los Angeles, California 90095, United States,
yi-chun.chen.phd@anderson.ucla.edu

Velibor V. Mišić
UCLA Anderson School of Management, University of California, Los Angeles, California 90095, United States,
velibor.misic@anderson.ucla.edu

The decision forest model is a recently proposed nonparametric choice model that is capable of representing any discrete choice model and in particular, can be used to represent non-rational customer behavior. In this paper, we study the problem of finding the assortment that maximizes expected revenue under the decision forest model. This problem is of practical importance because it allows a firm to tailor its product offerings to profitably exploit deviations from rational customer behavior, but at the same time is challenging due to the extremely general nature of the decision forest model. We approach this problem from a mixed-integer optimization perspective and propose three different formulations. We theoretically compare these formulations in strength, and analyze when these formulations are integral in the special case of a single tree. We propose a methodology for solving these problems at a large-scale based on Benders decomposition, and show that the Benders subproblem can be solved efficiently by primal-dual greedy algorithms when the master solution is fractional for two of our formulations, and in closed form when the master solution is binary for all of our formulations. Using synthetically generated instances, we demonstrate the practical tractability of our formulations and our Benders decomposition approach, and their edge over heuristic approaches.

Key words: decision trees; choice modeling; integer optimization; Benders decomposition

1. Introduction

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

While assortment optimization and the related problem of product line design have been studied extensively under a wide range of choice models, the majority of research in this area focuses on rational choice models, specifically those that follow the random utility maximization (RUM) assumption. A significant body of research in the behavioral sciences shows that customers behave in ways that deviate significantly from predictions made by RUM models. In addition, there is a substantial number of empirical examples of firms that make assortment decisions in ways that directly exploit customer irrationality. For example, the paper of Kivetz et al. (2004) provides an example of an assortment of document preparation systems from Xerox that are structured around the decoy effect, and an example of an assortment of servers from IBM that are structured around the compromise effect.

In the authors' recent paper Chen and Mišić (2019), we proposed a new choice model called the decision forest (DF) model for capturing customer irrationalities. This model involves representing the customer population as a probability distribution over binary trees, with each tree representing the decision process of one customer type. In a key result of our paper, we showed that this model is universal: *every discrete choice model is representable as a DF model.* While the paper of Chen and Mišić (2019) alludes to the downstream assortment optimization problem, it is entirely focused on model representation and prediction: it does not provide any answer to how one can select an optimal assortment with respect to a DF model.

In the present paper, we present a methodology for assortment optimization under the decision forest model, based on mixed-integer optimization. Our approach allows the firm to obtain assortments that are either provably optimal or within a desired optimality gap for a given DF model. At the same time, the approach easily allows the firm to incorporate business rules as linear constraints in the MIO formulation. Most importantly, given the universality property of the decision forest model, our optimization approach allows a firm to optimally tailor its assortment to any kind of predictable irrationality in the firm's customer population.

We make the following specific contributions:

1. We propose three different integer optimization models – LeafMIO, SplitMIO and ProductMIO– for the problem of assortment optimization under the DF model. We show that these three formulations are ordered by strength, with LeafMIO being the weakest and ProductMIO being the strongest. In the special case of a single purchase decision tree, we show that LeafMIO is in general not integral, SplitMIO is integral in the special case that a product appears at most once in the splits of a purchase decision tree, and ProductMIO is always integral regardless of the structure of the tree.
2. We propose a Benders decomposition approach for solving the three formulations at a large scale. We show that Benders cuts for the linear optimization relaxations of LeafMIO and SplitMIO can be obtained via a greedy algorithm that solves both the primal and dual of the subproblem. We also provide a simple example to show that the same type of greedy algorithm fails to solve the primal subproblem of ProductMIO. We also show how to obtain Benders cuts for the integer solutions of LeafMIO, SplitMIO and ProductMIO in closed form.
3. We present numerical experiments using synthetic data to compare our different formulations. For small instances (100 products, up to 500 trees and up to 64 leaves per tree), we show that all of our formulations can obtain solutions with low optimality gaps within a one hour time limit. We also show that there is a substantial difference in the tightness of the LO bounds of the different formulations, and that the solutions obtained by our approach often significantly outperform solutions obtained by simple heuristic approaches. For large-scale instances (up to 3000 products, 500 trees and 512 leaves per tree), we show that our Benders decomposition approach for the SplitMIO formulation is able to solve constrained instances of the assortment optimization problem to a low optimality gap within a two hour time limit.

The rest of the paper is organized as follows. In Section 2, we review the related literature in choice modeling and assortment optimization. In Section 3, we define the problem, and define our three MIO formulations. In Section 4, we consider a Benders decomposition approach to our three formulations, and analyze the subproblem for each of the three formulations for fractional and binary solutions of the master problem. In Section 5, we present the results of our numerical experiments. Finally, in Section 6, we conclude and provide some directions for future research.

2. Literature review

The problem of assortment optimization has been extensively studied in the operations management community; we refer readers to Gallego and Topaloglu (2019) for a recent review of the literature. The literature on assortment optimization has focused on developing approaches for finding the optimal assortment under many different rational choice models, such as the MNL model (Talluri and Van Ryzin 2004, Sumida et al. 2020), the latent class MNL model (Bront et al. 2009, Méndez-Díaz et al. 2014, Şen et al. 2018), the nested logit model (Davis et al. 2014, Alfandari et al. 2021) the Markov chain choice model (Feldman and Topaloglu 2017, Désir et al. 2020) and the ranking-based model (Aouad et al. 2020, 2018, Feldman et al. 2019).

In addition to the assortment optimization literature, our paper is also related to the literature on product line design found in the marketing community. While assortment optimization is more often focused on the tactical decision of selecting which existing products to offer, where the products are ones that have been sold in the past and the choice model comes from transaction data involving those products, the product line design problem involves selecting which new products to offer, where the products are candidate products (i.e., they have not been offered before) and the choice model comes from conjoint survey data, where customers are asked to rate or choose between hypothetical products. Research in this area has considered different approaches to solve the problem under the ranking-based/first-choice model (McBride and Zufryden 1988, Belloni et al. 2008, Bertsimas and Mišić 2019) and the multinomial logit model (Chen and Hausman 2000, Schön 2010); for more details, we refer the reader to the literature review of Bertsimas and Mišić (2019).

Our paper is most closely related to Belloni et al. (2008) and Bertsimas and Mišić (2019), both of which present integer optimization formulations of the product line design problem when the choice model is a ranking-based model. As we will see later, our formulations LeafMIO and SplitMIO can be viewed as generalizations of the formulations of Belloni et al. (2008) and Bertsimas and Mišić (2019), respectively, to the decision forest model. In addition, the paper of Bertsimas and Mišić (2019) develops a specialized Benders decomposition approach for its formulation, which uses the fact that one can solve the subproblem associated with each customer type by applying a greedy algorithm. We will show in Section 4 that this same property generalizes to two of our formulations, LeafMIO and SplitMIO, leading to tailored Benders decomposition algorithms for solving these problems at scale.

Beyond these specific connections, the majority of the literature on assortment optimization and product line design considers rational choice models, whereas our paper contributes a methodology for non-rational assortment optimization. Fewer papers have focused on choice modeling for irrational customer behavior; besides the decision forest model, other models include the generalized attraction model (GAM; Gallego et al. 2015), the generalized stochastic preference model (Berbeglia 2018) and the generalized Luce model (Echenique and Saito 2019). An even smaller set of papers has considered assortment optimization under non-rational choice models, which we now review. The paper of Flores et al. (2017) considers assortment optimization under the two-stage Luce model, and develops a polynomial time algorithm for solving the unconstrained assortment optimization problem. The paper of Rooderkerk et al. (2011) considers a context-dependent utility model where the utility of a product can depend on other products that are offered and that can capture compromise, attraction and similarity effects; the paper empirically demonstrates how incorporating context effects leads to a predicted increase of 5.4% in expected profit.

Relative to these papers, our paper differs in that it considers the decision forest model. As noted earlier, the decision forest model can represent any type of choice behavior, and as such, an assortment optimization methodology based on such a model is attractive in terms of allowing a firm to take the next step from a high-fidelity model to a decision. In addition, our methodology is built on mixed-integer optimization. This is advantageous because it allows a firm to leverage continuing improvements in solution software for integer optimization (examples include commercial solvers like Gurobi and CPLEX), as well as continuing improvements in computer hardware. At the same time, integer optimization allows firms to accommodate business requirements using linear constraints, which further enhances the practical applicability of the approach. Lastly, integer optimization also allows one to take advantage of well-studied large-scale solution methods for integer optimization problems. One such method that we focus on in this paper is Benders decomposition, which has seen an impressive resurgence in recent years for delivering state-of-the-art performance on large-scale problems such as hub location (Contreras et al. 2011), facility location (Fischetti et al. 2017) and set covering (Cordeau et al. 2019); see also Rahmaniani et al. (2017) for a review of the recent literature. Stated more concisely, the main contribution of our paper is a general-purpose methodology for assortment optimization under a general-purpose choice model.

In addition to the assortment optimization and product line design literatures, our formulations also have connections with others that have been proposed in the broader optimization literature. The formulation SplitMIO that we will present later can be viewed as a special case of the

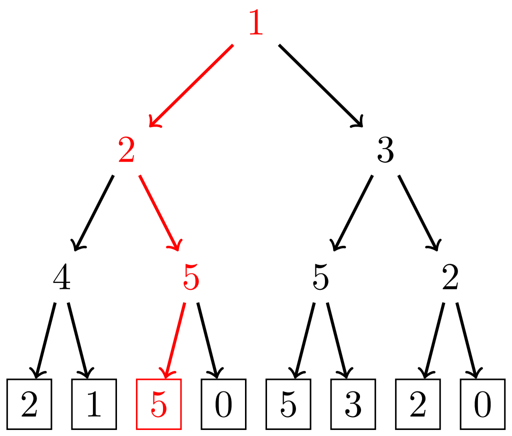


Figure 1 **Example of a purchase decision tree for $n = 5$ products. Leaf nodes are enclosed in squares, while split nodes are not enclosed. The number on each node corresponds either to $v(t,s)$ for splits, or $c(t,\ell)$ for leaves. The path highlighted in red indicates how a customer following this tree maps the assortment $S = \{1,3,4,5\}$ to a leaf. For this assortment, the customer's decision is to purchase product 5.**

mixed-integer optimization formulation of Mišić (2020) for optimizing the predicted value of a tree ensemble model, such as a random forest or a boosted tree model; we discuss this connection in more detail in Section 3.3. The formulation ProductMIO, which is our strongest formulation, also has a connection to the literature in the integer optimization community on formulating disjunctive constraints through independent branching schemes (Vielma et al. 2010, Vielma and Nemhauser 2011, Huchette and Vielma 2019); we also discuss this connection in more detail in Section 3.4.

3. Optimization model

In this section, we define the decision forest assortment optimization problem (Section 3.1) and subsequently develop our three formulations, LeafMIO (Section 3.2), SplitMIO (Section 3.2) and ProductMIO (Section 3.4).

3.1. Problem definition

In this section, we briefly review the decision forest model of Chen and Mišić (2019), and then formally state the assortment optimization problem. We assume that there are n products, indexed from 1 to n, and let $\mathcal{N} = \{1,\ldots,n\}$ denote the set of all products. An assortment S corresponds to a subset of $\mathcal{N}$. When offered S, a customer may choose to purchase one of the products in S, or to not purchase anything at all; we use the index 0 to denote the latter possibility, which we will also refer to as the no-purchase option.

The basic building block of the decision forest model is a purchase decision tree. A purchase decision tree is a directed binary tree, with each leaf node corresponding to an option in $\mathcal{N} \cup \{0\}$, and each non-leaf (or *split*) node corresponding to a product in $\mathcal{N}$. We use $\textbf{splits}(t)$ to denote the set of split nodes of tree t, and $\textbf{leaves}(t)$ to denote the set of leaf nodes. We use $c(t,\ell)$ to denote the purchase decision of leaf ℓ of tree t, i.e., the option chosen by tree t if the assortment is mapped to leaf ℓ. We use $v(t,s)$ to denote the product that is checked at split node s in tree t.

Each tree represents the purchasing behavior of one type of customer. Specifically, for an assortment S, the customer behaves as follows: the customer starts at the root of the tree. The customer checks whether the product corresponding to the root node is contained in S; if it is, he proceeds to the left child, and if not, he proceeds to the right child. He then checks again with the product at the new node, and the process repeats, until the customer reaches a leaf; the option that is at the leaf represents the choice of that customer. Figure 1 shows an example of a purchase decision tree being used to map an assortment to a purchase decision.

We make the following assumption about our purchase decision trees.

Assumption 1. *Let t be a purchase decision tree. For any two split nodes s and s' of t such that s' is a descendant of s, $v(t,s) \neq v(t,s)$.*

This assumption states that once a product appears on a split s, it cannot appear on any subsequent split s' that is reached by proceeding to the left or right child of s; in other words, each product in $\mathcal{N}$ appears at most once along the path from the root node to a leaf node, for every leaf node. As discussed in Chen and Mišić (2019), this assumption is not restrictive, as any tree for which this assumption is violated has a set of splits and leaves that are redundant and unreachable, and the tree can be modified to obtain an equivalent tree that satisfies the assumption.

The decision forest model assumes that the customer population is represented by a collection or *forest* F. Each tree $t \in F$ corresponds to a different customer type. We use λ_t to denote the probability associated with customer type/tree t, and $\boldsymbol{\lambda} = (\lambda_t)_{t \in F}$ to denote the probability distribution over the forest F. For each tree t, we use $\hat{A}(t,S)$ to denote the choice that a customer type following tree t will make when given the assortment S. For a given assortment $S \subseteq \mathcal{N}$ and a given choice $j \in S \cup \{0\}$, we use $\mathbf{P}^{(F,\boldsymbol{\lambda})}(j \mid S)$ to denote the choice probability, i.e., the probability of a random customer customer choosing j when offered the assortment S. It is defined as

$$\mathbf{P}^{(F,\boldsymbol{\lambda})}(j \mid S) = \sum_{t \in F} \lambda_t \cdot \mathbb{I}\{\hat{A}(t,S) = j\}. \tag{1}$$

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

$$\underset{S \subseteq \mathcal{N}}{\text{maximize}} \sum_{i \in S} \bar{r}_i \cdot \mathbf{P}^{(F,\boldsymbol{\lambda})}(i \mid S). \tag{2}$$

This is a challenging problem because of the general nature of the choice model $\mathbf{P}^{(F,\boldsymbol{\lambda})}(\cdot \mid \cdot)$. It turns out that problem (2) is theoretically intractable.

Proposition 1. *The decision forest assortment optimization problem* (2) *is NP-Hard.*

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

3.2. Formulation 1: LeafMIO

We now present our first formulation of the assortment optimization problem (2) as a mixed-integer optimization (MIO) problem. To formulate the problem, we introduce some additional notation. For notational convenience we let $r_{t,\ell} = \bar{r}_{c(t,\ell)}$ be the revenue of the purchase option of leaf ℓ of tree t. We let $\mathbf{left}(s)$ denote the set of leaf nodes that are to the left of split s (i.e., can only be reached by taking the left branch at split s), and similarly, we let $\mathbf{right}(s)$ denote the leaf nodes that are to the right of s.

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

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

$$\text{LeafMIO}: \quad \underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{t \in F} \lambda_t \cdot \left[\sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} y_{t,\ell} \right] \tag{3a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, \quad \forall\, t \in F, \tag{3b}$$

$$y_{t,\ell} \leq x_{v(t,s)}, \quad \forall\, t \in F,\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{left}(s), \tag{3c}$$
$$y_{t,\ell} \leq 1 - x_{v(t,s)}, \quad \forall\, t \in F,\ s \in \mathbf{splits}(t),\ \ell \in \mathbf{right}(s), \tag{3d}$$
$$x_i \in \{0,1\}, \quad \forall\, i \in \mathcal{N}, \tag{3e}$$
$$y_{t,\ell} \geq 0, \quad \forall\, t \in F,\ \ell \in \mathbf{leaves}(t). \tag{3f}$$

In order of appearance, the constraints in this formulation have the following meaning. Constraint (3b) requires that for each customer type t, the assortment encoded by $\mathbf{x}$ is mapped to exactly one leaf. Constraint (3c) requires that for any split s and any leaf ℓ that is to the left of split s, the assortment can be mapped to leaf ℓ only if the assortment includes the product $v(t,s)$ (i.e., if product $v(t,s)$ is not included in the assortment, then $y_{t,\ell}$ is forced to zero). Similarly, constraint (3d) requires the same for each split s and each leaf ℓ that is to the right of split s. The last two constraints require that $\mathbf{x}$ is binary and $\mathbf{y}$ is nonnegative. Note that it is not necessary to require $\mathbf{y}$ to be binary, as the constraints ensure that each $y_{t,\ell}$ automatically takes the correct value whenever $\mathbf{x}$ is binary. Finally, the objective function corresponds to the expected per-customer revenue of the assortment.

Formulation LeafMIO is related to two other formulations, arising in the assortment optimization and product line design literature. In the literature on first-choice product line design, Belloni et al. (2008) proposed a similar formulation for selecting a product line out of a collection of candidate products when the choice model is given by a collection of rankings. The formulation of that paper is actually a special case of LeafMIO when the decision forest corresponds to a ranking-based model. In the assortment optimization literature, Feldman et al. (2019) studied a modified version of Belloni et al. (2008), wherein one omits the unit sum constraint (3b). The paper shows that the resulting formulation possesses a half-integrality property, in that every basic feasible solution $(\mathbf{x},\mathbf{y})$ of the relaxation is such that $x_i \in \{0, 0.5, 1\}$ for all i. The paper then uses this property to design an approximation algorithm for the ranking-based assortment optimization problem. Formulation LeafMIO can potentially be modified in the same way and used to develop an approximation algorithm for the decision forest assortment optimization problem; we leave the pursuit of this question to future research.

To motivate our main result, let $\mathcal{F}_{\text{LeafMIO}}$ denote the feasible region of the linear optimization relaxation of problem (3). Our main result is that, even in the simple case when the forest F consists of a single tree, $\mathcal{F}_{\text{LeafMIO}}$ may fail to be integral.

Proposition 2. *There exists a decision forest model* $(F, \boldsymbol{\lambda})$ *with* $|F| = 1$ *such that* $\mathcal{F}_{\text{LeafMIO}}$ *is not integral, that is, there exists an extreme point* $(\mathbf{x},\mathbf{y}) \in \mathcal{F}_{\text{LeafMIO}}$ *such that* $\mathbf{x} \notin \{0,1\}^n$.

To prove the proposition, we directly propose an instance of formulation LeafMIO with $|F| = 1$ and a non-integral extreme point. We comment on two interesting aspects of this result. First, we note that the paper of Bertsimas and Mišić (2019) develops a similar result (Proposition 5 of that paper) for the Belloni et al. (2008) formulation. Since that formulation is a special case of LeafMIO, Proposition 2 is implied by this existing result. However, the instance that we construct for Proposition 2 is special in that it does not correspond to a ranking. Thus, even when the decision forest consists of one tree that is not a ranking, formulation LeafMIO can be non-integral.

Second, the instance that we use to prove Proposition 2 is constructed so that each split corresponds to a distinct product, i.e., each product appears at most once in the splits of the tree. The dichotomy between decision forests where a product appears at most once in the splits of a given tree, and decision forests where a product may appear in two or more splits of a given tree, is important: the next formulation that we will consider, SplitMIO, is guaranteed to be integral in the former case when $|F| = 1$, but can be non-integral in the latter case.

3.3. Formulation 2: SplitMIO

While problem LeafMIO is one formulation of problem (2), it is not the strongest possible formulation. In particular, for a fixed split s, the constraints (3c) and (3d) can be aggregated over all leaves in $\mathbf{left}(s)$ and $\mathbf{right}(s)$, respectively, for a fixed split s. This leads to our second formulation, SplitMIO, which is defined below.

$$\begin{aligned}
\textsc{SplitMIO}: \quad & \underset{\mathbf{x},\mathbf{y}}{\text{maximize}} && \sum_{t\in F} \lambda_t \cdot \left[\sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} y_{t,\ell} \right] && \text{(4a)}\\
& \text{subject to} && \sum_{\ell\in\mathbf{leaves}(t)} y_{t,\ell} = 1, \quad \forall\, t \in F, && \text{(4b)}\\
& && \sum_{\ell\in\mathbf{left}(s)} y_{t,\ell} \leq x_{v(t,s)}, \quad \forall\, t\in F,\ s \in \mathbf{splits}(t), && \text{(4c)}\\
& && \sum_{\ell\in\mathbf{right}(s)} y_{t,\ell} \leq 1 - x_{v(t,s)}, \quad \forall\, t\in F,\ s \in \mathbf{splits}(t), && \text{(4d)}\\
& && x_i \in \{0,1\}, \quad \forall\, i \in \mathcal{N}, && \text{(4e)}\\
& && y_{t,\ell} \geq 0, \quad \forall\, t\in F,\ \ell \in \mathbf{leaves}(t). && \text{(4f)}
\end{aligned}$$

Constraints (4b), (4e) and (4f) and the objective function are the same as in formulation LeafMIO. Constraint (4c) is an aggregated version of constraint (3c): for a split s in tree t, if product $v(t,s)$ is not in the assortment, then the assortment cannot be mapped to any of the leaves that are to the left of split s in tree t. Similarly, constraint (4d) is an aggregated version of constraint (3d), requiring that if $v(t,s)$ is included in the assortment, then the assortment cannot be mapped to any leaf to the right of split s in tree t. As in LeafMIO, $\mathbf{y}$ is modeled as nonnegative without affecting the validity of the formulation.

The above formulation we present here is related to two existing MIO formulations in the literature. The formulation here can be viewed as a specialized case of the MIO formulation in Mišić (2020). In that paper, the author develops a formulation for tree ensemble optimization, i.e., the problem of setting the independent variables in a tree ensemble model (e.g., a random forest or a gradient boosted tree model) to maximize the value predicted by that ensemble. Since the decision forest model is a type of tree ensemble model, where the "independent variables" are binary (i.e., product i is in the assortment or not), the formulation in Mišić (2020) naturally applies here, leading to problem (4).

In addition to Mišić (2020), problem (4) also relates to another MIO formulation, specifically that of Bertsimas and Mišić (2019). In that paper, the authors develop a formulation for the product line design problem under the ranking-based model. Chen and Mišić (2019) showed that the ranking-based model can be regarded as a special case of the decision forest model. In the special case that each tree in the forest corresponds to a ranking and the decision forest corresponds to a ranking-based choice model, it can be verified that the formulation (4) actually coincides with the MIO formulation for product line design under ranking-based models presented in Bertsimas and Mišić (2019).

Before continuing to our other formulations, we establish a couple of important properties of problem (4). Let $\mathcal{F}_{\textsc{SplitMIO}}$ denote the feasible region of the linear optimization relaxation of problem (4). Our first result, alluded to above, is that SplitMIO is at least as strong as LeafMIO.

Proposition 3. *For any decision forest model* $(F,\boldsymbol{\lambda})$, $\mathcal{F}_{\textsc{SplitMIO}} \subseteq \mathcal{F}_{\textsc{LeafMIO}}$.

This result follows straightforwardly from the definition of SplitMIO; we thus omit the proof.

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

PROPOSITION 4. *Let $(F, \boldsymbol{\lambda})$ be a decision forest model consisting of a single tree, i.e., $|F| = 1$. In addition, assume that for every $i \in \mathcal{N}$, $v(s) = i$ for at most one $s \in$ **splits**. Then $\mathcal{F}_{\text{SPLITMIO}}$ is integral, i.e., every extreme point $(\mathbf{x}, \mathbf{y})$ of the polyhedron $\mathcal{F}_{\text{SPLITMIO}}$ satisfies $\mathbf{x} \in \{0,1\}^N$.*

The proof of this result (see Section EC.2.3 of the ecompanion) follows by showing that the constraint matrix defining $\mathcal{F}_{\text{SPLITMIO}}$ is totally unimodular. This result is a generalization of a similar result for the ranking-based assortment optimization formulation of Bertsimas and Mišić (2019) (see Proposition 4 in that paper); the same procedure for establishing the total unimodularity of the constraint matrix also applies to decision forest models. Note also that Proposition 4 in Bertsimas and Mišić (2019) for the ranking-based formulation of that paper is implied by Proposition 4, due to the aforementioned equivalence between that formulation and SPLITMIO when the trees correspond to rankings, and the fact that when one represents a ranking as a tree, a product cannot appear more than once in the splits.

In addition to Proposition 5, we also have the following proposition that sheds light on when SPLITMIO is not integral.

PROPOSITION 5. *There exists a decision forest model $(F, \boldsymbol{\lambda})$ with $|F| = 1$ and for which $v(s_1) = v(s_2) = i$ for at least two $s_1, s_2 \in$ **splits**, $s_1 \neq s_2$ and at least one $i \in \mathcal{N}$, such that $\mathcal{F}_{\text{SPLITMIO}}$ is not integral.*

The proof of this result is given in Section EC.2.4 of the ecompanion. Proposition 5 is significant because it implies that for $|F| = 1$, the distinction between trees where each product appears at most once in any split and trees where a product may appear two or more times as a split is sharp. This insight provides the motivation for our third formulation, PRODUCTMIO, which we present next.

3.4. Formulation 3: ProductMIO

The third formulation of problem (2) that we will present is motivated by the behavior of SPLITMIO when a product participates in two or more splits. In particular, observe that in a given purchase decision tree, a product i may participate in two different splits s_1 and s_2 in the same tree. In this case, constraint (4c) in problem (4) will result in two constraints:

$$\sum_{\ell \in \mathbf{left}(s_1)} y_{t,\ell} \leq x_i, \tag{5}$$

$$\sum_{\ell \in \mathbf{left}(s_2)} y_{t,\ell} \leq x_i. \tag{6}$$

In the above two constraints, observe that $\mathbf{left}(s_1)$ and $\mathbf{left}(s_2)$ are disjoint (this is a straightforward consequence of Assumption 1). Given this and constraint (4b) that requires the $y_{t,\ell}$ variables to sum to 1, we can come up with a constraint that strengthens constraints (5) and (6) by combining them:

$$\sum_{\ell \in \mathbf{left}(s_1)} y_{t,\ell} + \sum_{\ell \in \mathbf{left}(s_2)} y_{t,\ell} \leq x_i. \tag{7}$$

In general, one can aggregate all the $y_{t,\ell}$ variables that are to the left of all splits involving a product i to produce a single left split constraint for product i. The same can also be done for the right split constraints. Generalizing this principle leads to the following alternate formulation, which we refer to as PRODUCTMIO:

$$\text{PRODUCTMIO}: \quad \underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{t \in F} \lambda_t \cdot \left[\sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} y_{t,\ell} \right] \tag{8a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, \quad \forall\, t \in F, \tag{8b}$$

$$\sum_{\substack{s \in \mathbf{splits}(t):\\ v(t,s)=i}} \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq x_i, \quad \forall\, t \in F,\ i \in \mathcal{N}, \tag{8c}$$

$$\sum_{\substack{s \in \mathbf{splits}(t):\\ v(t,s)=i}} \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - x_i, \quad \forall\, t \in F,\ i \in \mathcal{N}, \tag{8d}$$

$$x_i \in \{0,1\}, \quad \forall\, i \in \mathcal{N}, \tag{8e}$$

$$y_{t,\ell} \geq 0, \quad \forall\, t \in F,\ \ell \in \mathbf{leaves}(t). \tag{8f}$$

Relative to SplitMIO, ProductMIO differs in several ways. First, note that while both formulations have the same number of variables, formulation ProductMIO has a smaller number of constraints. In particular, problem SplitMIO has one left and one right split constraints for each split in each tree, whereas ProductMIO has one left and one right split constraint for each product. When the trees involve a large number of splits, this can lead to a sizable reduction in the number of constraints. Note also that when a product does not appear in any splits of a tree, we can also safely omit constraints (8c) and (8d) for that product.

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

Proposition 6. *For any decision forest model $(F, \boldsymbol{\lambda})$, $\mathcal{F}_{\text{ProductMIO}} \subseteq \mathcal{F}_{\text{SplitMIO}}$.*

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

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

Proposition 7. *For any decision forest model $(F, \boldsymbol{\lambda})$ with $|F| = 1$, $\mathcal{F}_{\text{ProductMIO}}$ is integral.*

The proof of this proposition, given in Section EC.2.5 of the electronic companion, follows by recognizing the connection between ProductMIO and another type of formulation in the literature. In particular, a stream of papers in the mixed-integer optimization community (Vielma et al. 2010, Vielma and Nemhauser 2011, Huchette and Vielma 2019) has considered a general approach for deriving small and strong formulations of disjunctive constraints using independent branching schemes; we briefly review the most general such approach from Huchette and Vielma (2019). In this approach, one has a finite ground set J, and is interested in optimizing over a particular subset of the $(|J| - 1)-$dimensional unit simplex over J, $\Delta^J = \{\boldsymbol{\lambda} \in \mathbb{R}^J \mid \sum_{j \in J} \lambda_j = 1; \boldsymbol{\lambda} \geq \mathbf{0}\}$. The specific subset that we are interested in is called a *combinatorial disjunctive constraint* (CDC), and is given by

$$\text{CDC}(\mathcal{S}) = \bigcup_{S \in \mathcal{S}} Q(S), \tag{9}$$

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

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

$$\text{CDC}(\mathcal{S}) = \bigcap_{m=1}^{t} \bigcup_{i=1}^{k} Q(L_i^m), \tag{10}$$

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

$$\sum_{j \in L_m} \lambda_j \leq z_m, \quad \forall\, m \in \{1, \ldots, k\}, \tag{11a}$$

$$\sum_{j \in R_m} \lambda_j \leq 1 - z_m, \quad \forall\, m \in \{1, \ldots, k\}, \tag{11b}$$

$$z_m \in \{0, 1\}, \quad \forall\, m \in \{1, \ldots, t\}, \tag{11c}$$

$$\sum_{j \in J} \lambda_j = 1, \tag{11d}$$

$$\lambda_j \geq 0, \quad \forall\, j \in J. \tag{11e}$$

This particular special case is important because it is always integral (see Theorem 1 of Vielma et al. 2010). Moreover, we can see that ProductMIO bears a strong resemblance to formulation (11). Constraints (11a) and (11a) correspond to constraints (8c) and (8d), respectively. In terms of variables, the λ_j and z_m variables in formulation (11) correspond to the $y_{t,\ell}$ and x_i variables in ProductMIO, respectively.

One notable difference is that in practice, one would use formulation (11) in a modular way; specifically, one would be faced with a problem where the feasible region can be written as $\text{CDC}(\mathcal{S}_1) \cap \text{CDC}(\mathcal{S}_2) \cap \cdots \cap \text{CDC}(\mathcal{S}_G)$, where each $\mathcal{S}_g$ is a collection of subsets of J. To model this feasible region, one would introduce a set of $z_{g,m}$ variables for the gth CDC, enforce constraints (11a) - (11c) for the gth CDC, and use only one set of λ_j variables for the whole formulation. Thus, the λ_j variables are the "global" variables, while the $z_{g,m}$ variables would be "local" and specific to each CDC. In contrast, in ProductMIO, the x_i variables (the analogues of z_m) are the "global" variables, while the $y_{t,\ell}$ variables (the analogues of λ_j) are the "local" variables.

4. Solution methodology based on Benders decomposition

While the formulations in Section 3 bring the assortment optimization problem under the decision forest choice model closer to being solvable in practice, the effectiveness of these formulations can be limited in large-scale problems. In particular, consider the case where there is a large number of trees in the decision forest model and each tree consists of a large number of splits and leaves. In this setting, all three formulations – LeafMIO, SplitMIO and ProductMIO– will have a large number of $y_{t,\ell}$ variables and a large number of constraints to link those variables with the x_i variables, and may require significant computation time.

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

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

The main message of this section of the paper is that, in most cases, the primal and the dual forms of the second-stage problem can be solved either in closed form (when $\mathbf{x}$ is binary) or via a greedy algorithm (when $\mathbf{x}$ is fractional), thus allowing one to identify violated constraints for either the relaxation or the integer problem in a computationally efficient manner. In the remaining sections, we carefully analyze the second-stage problem for each of the three formulations. For LeafMIO, we show that the second-stage problem can be solved by a greedy algorithm when $\mathbf{x}$ is fractional (Section 4.1). For SplitMIO, we similarly show that the second-stage problem can be solved by a slightly different greedy algorithm when $\mathbf{x}$ is fractional (Section 4.2). For ProductMIO, we show that the same greedy approach does not solve the second-stage problem in the fractional case (Section 4.3). For all three formulations, when $\mathbf{x}$ is binary, we characterize the primal and dual solutions in closed form; due to space considerations, we relegate these results to the electronic companion (LeafMIO in Section EC.1.1, SplitMIO in Section EC.1.2 and ProductMIO in Section EC.1.3). Lastly, in Section 4.4, we briefly describe our overall algorithmic approach to solving the assortment optimization problem, which involves solving the Benders reformulation of the relaxed problem, followed by the Benders reformulation of the integer problem.

4.1. Benders reformulation of the LeafMIO relaxation

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

$$\begin{aligned}
\underset{\mathbf{x},\boldsymbol{\theta}}{\text{maximize}} \quad & \sum_{t\in F} \lambda_t \theta_t && \text{(12a)}\\
\text{subject to} \quad & \theta_t \leq G_t(\mathbf{x}), \quad \forall\, t \in F, && \text{(12b)}\\
& \mathbf{x} \in [0,1]^n, && \text{(12c)}
\end{aligned}$$

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

$$\begin{aligned}
G_t(\mathbf{x}) \;=\; \underset{\mathbf{y}_t}{\text{maximize}} \quad & \sum_{\ell\in\mathbf{leaves}(t)} r_{t,\ell}\cdot y_{t,\ell} && \text{(13a)}\\
\text{subject to} \quad & \sum_{\ell\in\mathbf{leaves}(t)} y_{t,\ell} = 1, && \text{(13b)}\\
& y_{t,\ell} \leq x_{v(t,s)}, \quad \forall\, s\in\mathbf{splits}(t),\ \ell\in\mathbf{left}(s), && \text{(13c)}\\
& y_{t,\ell} \leq 1 - x_{v(t,s)}, \quad \forall\, s\in\mathbf{splits}(t),\ \ell\in\mathbf{right}(s), && \text{(13d)}\\
& y_{t,\ell} \geq 0, \quad \forall\, \ell\in\mathbf{leaves}(t). && \text{(13e)}
\end{aligned}$$

We now present a greedy algorithm for solving problem (13), which is presented below as Algorithm 1. The algorithm requires a bijection $\tau : \{1,\ldots,|\mathbf{leaves}(t)|\} \to \mathbf{leaves}(t)$ such that $r_{t,\tau(1)} \geq r_{t,\tau(2)} \geq \cdots \geq r_{t,\tau(|\mathbf{leaves}(t)|)}$, i.e., an ordering of leaves in nondecreasing revenue. In addition, in the definition of Algorithm 1, we use $\mathbf{LS}(\ell)$ and $\mathbf{RS}(\ell)$ to denote the set of left and right splits, respectively, of ℓ, which are defined as

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

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

Intuitively, this algorithm progresses through the leaves in order of their revenue $r_{t,\ell}$, and sets the $y_{t,\ell}$ variable of each leaf ℓ to the highest it can be set to without violating constraints (13c) and (13d), while also ensuring that $\sum_\ell y_{t,\ell} \leq 1$. At each stage of the algorithm, the algorithm keeps track of which constraints become tight through the event set $\mathcal{E}$. If the constraint (13c) becomes tight for a particular split-leaf pair (s,ℓ), we say that an $A_{s,\ell}$ event has occurred, and we add $A_{s,\ell}$ to $\mathcal{E}$. Similarly, if constraint (13d) becomes tight for (s,ℓ), we say that a $B_{s,\ell}$ event has occurred and add $B_{s,\ell}$ to $\mathcal{E}$. (In the case of a tie, that is, when there is more than one split s which attains the minimum on line 13 or 17, we choose the split arbitrarily.) If the constraint (13b) holds, then we say that a C event has occurred, and we terminate the algorithm, as all the remaining $y_{t,\ell}$ variables cannot be set to anything other than zero. In addition to the events in $\mathcal{E}$, we also keep track of which $y_{t,\ell}$ variable was being modified when each event in $\mathcal{E}$ occurred; this is done through the function f. We note that both $\mathcal{E}$ and f are not essential for the primal algorithm, but they become important for the dual algorithm (to be defined as Algorithm 2 below), in order to determine the corresponding dual solution.

Algorithm 1 Primal greedy algorithm for LeafMIO.

```
Require: Bijection $\tau : \{1,\dots,|\mathbf{leaves}(t)|\} \to \mathbf{leaves}(t)$ such that $r_{t,\tau(1)} \geq r_{t,\tau(2)} \geq \cdots \geq r_{t,\tau(|\mathbf{leaves}(t)|)}$
 1: Initialize $y_{t,\ell} \leftarrow 0$ for all $\ell \in \mathbf{leaves}(t)$
 2: for $i = 1,\dots,|\mathbf{leaves}(t)|$ do
 3:     Set $q_A \leftarrow \min\{x_{v(t,s)} \mid s \in \mathbf{LS}(\tau(i))\}$
 4:     Set $q_B \leftarrow \min\{1 - x_{v(t,s)} \mid s \in \mathbf{RS}(\tau(i))\}$
 5:     Set $q_C \leftarrow 1 - \sum_{j=1}^{i-1} y_{t,\tau(j)}$
 6:     Set $q^* \leftarrow \min\{q_A, q_B, q_C\}$
 7:     Set $y_{t,\tau(i)} \leftarrow q^*$
 8:     if $q^* = q_C$ then
 9:         Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{C\}$
10:         Set $f(C) = \tau(j)$
11:         break
12:     else if $q^* = q_A$ then
13:         Select $s^* \in \arg\min_{s \in \mathbf{LS}(\tau(i))} x_{v(t,s)}$
14:         Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{A_{s^*,\tau(i)}\}$
15:         Set $f(A_{s^*,\tau(i)}) = \tau(i)$
16:     else
17:         Select $s^* \in \arg\min_{s \in \mathbf{RS}(\tau(i))} [1 - x_{v(t,s)}]$
18:         Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{B_{s^*,\tau(i)}\}$
19:         Set $f(B_{s^*,\tau(i)}) = \tau(i)$
20:     end if
21: end for
```

It turns out that Algorithm 1 returns a feasible solution that is an extreme point of the polyhedron defined in problem (13), which we establish as Theorem 1 below.

Theorem 1. *Fix $t \in F$. Let $\mathbf{y}_t$ be a solution to problem* (13) *produced by Algorithm 1. Then:*
a) $\mathbf{y}_t$ is a feasible solution to problem (13).
b) $\mathbf{y}_t$ is an extreme point of the feasible region of problem (13).

By Theorem 1, problem (13) is feasible; since the feasible region is additionally bounded, it follows that problem (13) has a finite optimal value. Therefore, by strong duality, the optimal objective value of problem (13) is equal to the optimal value of its dual. The dual of problem (13) can be written as:

$$\underset{\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t}{\text{minimize}} \quad \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} x_{v(t,s)} + \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{right}(s)} \beta_{t,s,\ell}(1 - x_{v(t,s)}) + \gamma_t \tag{14a}$$

$$\text{subject to} \quad \sum_{s:\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} + \sum_{s:\ell \in \mathbf{right}(s)} \beta_{t,s,\ell} + \gamma_t \geq r_{t,\ell}, \quad \forall\, \ell \in \mathbf{leaves}(t), \tag{14b}$$

$$\alpha_{t,s,\ell} \geq 0, \quad \forall\, s \in \mathbf{splits}(t),\ \ell \in \mathbf{left}(s), \tag{14c}$$

$$\beta_{t,s,\ell} \geq 0, \quad \forall\, s \in \mathbf{splits}(t),\ \ell \in \mathbf{right}(s). \tag{14d}$$

Letting $\mathcal{D}_{t,\text{LeafMIO}}$ denote the set of feasible solutions $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ to the dual subproblem (14), we can re-write the master problem (12) as

$$\underset{\mathbf{x},\boldsymbol{\theta}}{\text{maximize}} \quad \sum_{t \in F} \lambda_t \theta_t \tag{15a}$$

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

$$\forall\, (\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t) \in \mathcal{D}_{t,\text{LeafMIO}}, \tag{15b}$$

$$\mathbf{x} \in [0,1]^n. \tag{15c}$$

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

The constraint generation approach to solving problem (15) involves starting the problem with no constraints and then, for each $t \in F$, checking whether constraint (15b) is violated. If constraint (15b) is not violated for any $t \in F$, then we conclude that the current solution $\mathbf{x}$ is optimal. Otherwise, for any $t \in F$ such that constraint (15b) is violated, we add the constraint corresponding to the $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ solution at which the violation occurred, and solve the problem again to obtain a new $\mathbf{x}$. The procedure then repeats at the new $\mathbf{x}$ solution until no more violated constraints have been found.

The critical step in the constraint generation approach is the separation procedure for constraint (15b): that is, for a fixed $t \in F$, either asserting that the current solution $\mathbf{x}$ satisfies constraint (15b) or identifying a $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ at which constraint (15b). This amounts to solving the dual subproblem (14) and comparing its objective value to θ_t.

Fortunately, it turns out that we can solve the dual subproblem (14) using a specialized algorithm, in the same way that we can solve the primal subproblem (13) using Algorithm 1. Using the event set $\mathcal{E}$ and the mapping f produced by Algorithm 1, we can now consider a separate algorithm for solving the dual problem (14), which we present below as Algorithm 2.

Algorithm 2 Dual greedy algorithm for LeafMIO.

```
1: Initialize $\alpha_{t,s,\ell} \leftarrow 0$, $\beta_{t,s,\ell} \leftarrow 0$ for all $s \in \mathbf{splits}(t)$, $\ell \in \mathbf{leaves}(t)$, $\gamma_t \leftarrow 0$.
2: Set $\gamma_t \leftarrow r_{t,f(C)}$
3: for $\ell \in \mathbf{leaves}(t)$ do
4:     Set $\alpha_{t,s,\ell} = r_{t,f(A_{s,\ell})} - \gamma_t$ for any $s$ such that $A_{s,\ell} \in \mathcal{E}$
5:     Set $\beta_{t,s,\ell} = r_{t,f(B_{s,\ell})} - \gamma_t$ for any $s$ such that $B_{s,\ell} \in \mathcal{E}$
6: end for
```

As with Algorithm 1, we can show that the dual solution produced by Algorithm 2 is a feasible extreme point solution of problem (14).

Theorem 2. *Fix $t \in F$. Let $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ be a solution to problem* (14) *produced by Algorithm 2. Then:*

a) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to problem (14).

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *is an extreme point of the feasible region of problem* (14).

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

THEOREM 3. *Fix* $t \in F$. *Let* $\mathbf{y}_t$ *be a solution to problem* (13) *produced by Algorithm 1 and let* $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *be a solution to problem* (14) *produced by Algorithm 2. Then:*
a) $\mathbf{y}_t$ *is an optimal solution to problem* (13).
b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *is an optimal solution to problem* (14).

Before continuing, we pause to make two important remarks on Theorem 3 and our results in this section. First, the value of Theorem 3 is that it allows us to use Algorithms 1 and 2 to solve the primal and dual subproblems (13) and (14). Thus, rather than invoking a linear optimization solver, such as Gurobi, to solve problem (14), we can simply run Algorithms 1 and 2.

Second, we note that the existence of a greedy algorithm is perhaps not too surprising, because of the connection between problem (13) and the 0-1 knapsack problem. In particular, consider the following problem:

$$\underset{\tilde{\mathbf{y}}_t}{\text{maximize}} \quad \sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} \cdot w_{t,\ell} \cdot \tilde{y}_{t,\ell} \tag{16a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}(t)} w_{t,\ell} \cdot \tilde{y}_{t,\ell} \leq 1, \tag{16b}$$

$$0 \leq \tilde{y}_{t,\ell} \leq 1, \quad \forall\, \ell \in \mathbf{leaves}(t). \tag{16c}$$

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

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

and $\tilde{y}_{t,\ell}$ is a new decision variable defined for each $\ell \in \mathbf{leaves}(t)$. Note that this problem is equivalent to problem (13) with the constraint $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1$ relaxed to $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} \leq 1$. The coefficient $w_{t,\ell}$ has the interpretation of the tightest upper bound on $y_{t,\ell}$ in problem (13). The variable $\tilde{y}_{t,\ell}$ can therefore be viewed as a re-scaling of $y_{t,\ell}$ relative to this bound; in other words, we can recover $y_{t,\ell}$ from a solution by setting it as $y_{t,\ell} = w_{t,\ell} \cdot \tilde{y}_{t,\ell}$. Problem (16) is special because it is exactly the linear optimization relaxation of a 0-1 knapsack problem: each leaf ℓ correspond to an item; each $w_{t,\ell}$ value corresponds to item ℓ's weight; and the coefficient $r_{t,\ell} \cdot w_{t,\ell}$ corresponds to the profit of item ℓ. It is well-known that the optimal solution to the relaxation of a 0-1 knapsack problem can be obtained via a greedy heuristic that sets the fractional amount of each item to the highest it can be, in order of decreasing profit-to-weight ratio (Martello and Toth 1987). For problem (16) above, the profit-to-weight ratio is exactly $r_{t,\ell} \cdot w_{t,\ell} / w_{t,\ell} = r_{t,\ell}$, so the greedy algorithm coincides with our greedy algorithm (Algorithm 1).

4.2. Benders reformulation of the SplitMIO relaxation

We now turn our attention to the SPLITMIO formulation. We can reformulate the relaxation of SPLITMIO in the same way as LEAFMIO; in particular, we have the same master problem (12), where the function $G_t(\mathbf{x})$ is now defined as the optimal value of the tree t subproblem in SPLITMIO:

$$G_t(\mathbf{x}) = \underset{\mathbf{y}_t}{\text{maximize}} \quad \sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} \cdot y_{t,\ell} \tag{17a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, \tag{17b}$$

$$\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq x_{v(t,s)}, \quad \forall\, s \in \mathbf{splits}(t), \tag{17c}$$

$$\sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - x_{v(t,s)}, \quad \forall\, s \in \mathbf{splits}(t), \tag{17d}$$

$$y_{t,\ell} \geq 0, \quad \forall\, \ell \in \mathbf{leaves}(t). \tag{17e}$$

As with LeafMIO, it turns out that the primal subproblem (17) can be solved using a greedy algorithm, which we present below as Algorithm 3. As with Algorithm 1, this algorithm requires as input an ordering τ of the leaves in nondecreasing revenue. Like Algorithm 1, this algorithm also progresses through the leaves from highest to lowest revenue, and sets the $y_{t,\ell}$ variable of each leaf ℓ to the highest value it can be set to without violating the left and right split constraints (17c) and (17d) and without violating the constraint $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} \leq 1$. At each iteration, the algorithm additionally keeps track of which constraint became tight through the event set $\mathcal{E}$. An A_s event indicates that the left split constraint (17c) for split s became tight; a B_s event indicates that the right split constraint (17d) for split s became tight; and a C event indicates that the constraint $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} \leq 1$ became tight. When a C event is not triggered, Algorithm 3 looks for the split which has the least remaining capacity (line 17). In the case that the arg min is not unique and there are two or more splits that are tied, we break ties by choosing the split s with the lowest depth $d(s)$ (i.e., the split closest to the root node of the tree).

The function f keeps track of which leaf ℓ was being checked when an A_s / B_s / C event occurred. As with LeafMIO, $\mathcal{E}$ and f are not needed to find the primal solution, but they are essential to determining the dual solution in the dual procedure (Algorithm 4, which we will define shortly).

The following result establishes that Algorithm 3 produces a feasible, extreme point solution of problem (17).

Theorem 4. *Fix $t \in F$. Let $\mathbf{y}_t$ be a solution to problem* (17) *produced by Algorithm 3. Then:*
a) $\mathbf{y}_t$ is a feasible solution to problem (17)*; and*
b) $\mathbf{y}_t$ is an extreme point of the feasible region of problem (17)*.*

As in our analysis of LeafMIO, a consequence of Theorem 4 is that problem (17) is feasible, and since the problem is bounded, it has a finite optimal value. By strong duality, the optimal value of problem (18) is equal to the optimal value of its dual:

$$\begin{aligned}
\underset{\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t}{\text{minimize}} \quad & \sum_{s \in \mathbf{splits}(t)} x_{v(t,s)} \cdot \alpha_{t,s} + \sum_{s \in \mathbf{splits}(t)} (1 - x_{v(t,s)})\beta_{t,s} + \gamma_t & \text{(18a)}\\
\text{subject to} \quad & \sum_{s:\ell \in \mathbf{left}(s)} \alpha_{t,s} + \sum_{s:\ell \in \mathbf{right}(s)} \beta_{t,s} + \gamma_t \geq r_{t,\ell}, \quad \forall\, \ell \in \mathbf{leaves}(t), & \text{(18b)}\\
& \alpha_{t,s} \geq 0, \quad \forall\, s \in \mathbf{splits}(t), & \text{(18c)}\\
& \beta_{t,s} \geq 0, \quad \forall\, s \in \mathbf{splits}(t). & \text{(18d)}
\end{aligned}$$

Letting $\mathcal{D}_{t,\text{SplitMIO}}$ denote the set of feasible solutions to the dual subproblem (18), we can formulate the master problem (12) as

$$\begin{aligned}
\underset{\mathbf{x}, \boldsymbol{\theta}}{\text{maximize}} \quad & \sum_{t \in F} \lambda_t \theta_t & \text{(19a)}\\
\text{subject to} \quad & \theta_t \leq \sum_{s \in \mathbf{splits}(t)} x_{v(t,s)} \cdot \alpha_{t,s} + \sum_{s \in \mathbf{splits}(t)} (1 - x_{v(t,s)})\beta_{t,s} + \gamma_t, & \\
& \quad \forall\, (\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t) \in \mathcal{D}_{t,\text{SplitMIO}}, & \text{(19b)}\\
& \mathbf{x} \in [0,1]^n. & \text{(19c)}
\end{aligned}$$

Algorithm 3 Primal greedy algorithm for SplitMIO.

```
Require: Bijection $\tau : \{1,\dots,|\mathbf{leaves}(t)|\} \to \mathbf{leaves}(t)$ such that $r_{t,\tau(1)} \geq r_{t,\tau(2)} \geq \cdots \geq r_{t,\tau(|\mathbf{leaves}(t)|)}$
 1: Initialize $y_{t,\ell} \leftarrow 0$ for each $\ell \in \mathbf{leaves}(t)$.
 2: for $i = 1,\dots,|\mathbf{leaves}(t)|$ do
 3:     Set $q_C \leftarrow 1 - \sum_{j=1}^{i-1} y_{t,\tau(j)}$.
 4:     for $s \in \mathbf{LS}(\tau(i))$ do
 5:         Set $q_s \leftarrow x_{v(t,s)} - \sum^{i-1}_{\substack{j=1:\\ \tau(j)\in\mathbf{left}(s)}} y_{t,\tau(j)}$
 6:     end for
 7:     for $s \in \mathbf{RS}(\tau(i))$ do
 8:         Set $q_s \leftarrow 1 - x_{v(t,s)} - \sum^{i-1}_{\substack{j=1:\\ \tau(j)\in\mathbf{right}(s)}} y_{t,\tau(j)}$
 9:     end for
10:     Set $q_{A,B} \leftarrow \min_{s\in\mathbf{LS}(\tau(i))\cup\mathbf{RS}(\tau(i))} q_s$
11:     Set $q^* \leftarrow \min\{q_C, q_{A,B}\}$
12:     Set $y_{t,\tau(i)} \leftarrow q^*$
13:     if $q^* = q_C$ then
14:         Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{C\}$.
15:         Set $f(C) = \tau(i)$.
16:     else
17:         Set $s^* \leftarrow \arg\min_{s\in\mathbf{LS}(\tau(i))\cup\mathbf{RS}(\tau(i))} q_s$
18:         if $s^* \in \mathbf{LS}(\tau(i))$ then
19:             Set $e = A_s$
20:         else
21:             Set $e = B_s$
22:         end if
23:         if $e \notin \mathcal{E}$ then
24:             Set $\mathcal{E} \leftarrow \mathcal{E} \cup \{e\}$.
25:             Set $f(e) = \tau(i)$.
26:         end if
27:     end if
28: end for
```

As with the Benders approach to LeafMIO, the crucial step to solving this problem is being able to solve the dual subproblem (18). Similarly to problem (17), we can also obtain a solution to the dual problem (18) via an algorithm that is formalized as Algorithm 4 below. Algorithm 4 uses auxiliary information obtained during the execution of Algorithm 3. In the definition of Algorithm 4, we use $d(s)$ to denote the depth of an arbitrary split, where the root split corresponds to a depth of 1, and $d_{\max} = \max_{s\in\mathbf{splits}(t)} d(s)$ is the depth of the deepest split in the tree. In addition, we use $\mathbf{splits}(t,d) = \{s \in \mathbf{splits}(t) \mid d(s) = d\}$ to denote the set of all splits at a particular depth d.

We provide a worked example of the execution of both Algorithms 3 and 4 in Section EC.3.

Our next result, Theorem 5, establishes that Algorithm 4 returns a feasible, extreme point solution of the dual subproblem (18).

Theorem 5. *Fix $t \in F$. Let $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ be a solution to problem* (18) *produced by Algorithm 4. Then:*

a) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to problem (18)*; and*

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is an extreme point of the feasible region of problem (18).

Lastly, and most importantly, we show that the solutions produced by Algorithms 3 and 4 are optimal for their respective problems. Thus, Algorithm 4 is a valid procedure for identifying values of $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ at which constraint (19b) is violated.

Algorithm 4 Dual greedy algorithm for SplitMIO.

```
1: Initialize $\alpha_{t,s} \leftarrow 0, \beta_{t,s} \leftarrow 0$ for all $s \in \mathbf{splits}(t)$, $\gamma_t \leftarrow 0$
2: Set $\gamma \leftarrow r_{f(C)}$
3: for $d = 1, \dots, d_{\max}$ do
4:     for $s \in \mathbf{splits}(t,d)$ do
5:         if $A_s \in \mathcal{E}$ then
6:             Set $\alpha_{t,s} \leftarrow r_{t,f(A_s)} - \gamma_t - \sum_{\substack{s' \in \mathbf{LS}(f(A_s)):\\ A_{s'} \in \mathcal{E},\\ d(s') < d}} \alpha_{t,s'} - \sum_{\substack{s' \in \mathbf{RS}(f(A_s)):\\ B_{s'} \in \mathcal{E},\\ d(s') < d}} \beta_{t,s'}$
7:         end if
8:         if $B_s \in \mathcal{E}$ then
9:             Set $\beta_{t,s} \leftarrow r_{t,f(B_s)} - \gamma_t - \sum_{\substack{s' \in \mathbf{LS}(f(A_s)):\\ A_{s'} \in \mathcal{E},\\ d(s') < d}} \alpha_{t,s'} - \sum_{\substack{s' \in \mathbf{RS}(f(A_s)):\\ B_{s'} \in \mathcal{E},\\ d(s') < d}} \beta_{t,s'}$
10:        end if
11:    end for
12: end for
```

Theorem 6. *Fix $t \in F$. Let $\mathbf{y}_t$ be a solution to problem* (17) *produced by Algorithm 3 and* $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *be a solution to problem* (18) *produced by Algorithm 4. Then:*
a) $\mathbf{y}_t$ is an optimal solution to problem (17)*; and*
b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is an optimal solution to problem (18).

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

Before continuing, we note that Algorithms 3 and 4 can be viewed as the generalization of the algorithms arising in the Benders decomposition approach to the ranking-based assortment optimization problem in Bertsimas and Mišić (2019) (see Section 4 of that paper). The results of that paper show that the primal subproblem of the MIO formulation in Bertsimas and Mišić (2019) can be solved via a greedy algorithm (analogous to Algorithm 3) and the dual subproblem can be solved via an algorithm that uses information from the primal algorithm (analogous to Algorithm 4). This generalization is not straightforward. The main challenge in this generalization is redesigning the sequence of updates in the greedy algorithm according to the tree topology. For the ranking-based assortment problem, one only needs to calculate the "capacities" (the q_s values in Algorithm 3) by subtracting the y values of the preceding products in the rank. In contrast, in Algorithm 3, one considers all left/right splits and the y values of their left/right leaves when constructing the lowest upper bound of y_ℓ for each leaf node ℓ. Also, as shown in Algorithm 4, the dual variables $\alpha_{t,s}$ and $\beta_{t,s}$ have to be updated according to the tree topology and the events $A_{s'}$ and $B_{s'}$ of the split s' with smaller depth. For these reasons, the primal and dual Benders subproblems for the decision forest assortment problem are more challenging than that of the ranking-based assortment problem.

4.3. Benders reformulation of the ProductMIO relaxation

Lastly, we can consider a Benders reformulation of the relaxation of ProductMIO. The Benders master problem is given by formulation (12) where the function $G_t(\mathbf{x})$ is defined as the optimal value of the ProductMIO subproblem for tree t. To aid in the definition of the subproblem, let $P(t)$ denote the set of products that appear in the splits of tree t:

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

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

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

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

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

$$\begin{aligned}
G_t(\mathbf{x}) = \quad & \underset{\mathbf{y}_t}{\text{maximize}} && \sum_{\ell \in \mathbf{leaves}(t)} r_{t,\ell} \cdot y_{t,\ell} && \text{(20a)}\\
& \text{subject to} && \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, && \text{(20b)}\\
& && \sum_{\ell \in \mathbf{left}(i)} y_{t,\ell} \leq x_i, \quad \forall\, i \in P(t), && \text{(20c)}\\
& && \sum_{\ell \in \mathbf{right}(i)} y_{t,\ell} \leq 1 - x_i, \quad \forall\, i \in P(t), && \text{(20d)}\\
& && y_{t,\ell} \geq 0, \quad \forall\, \ell \in \mathbf{leaves}(t). && \text{(20e)}
\end{aligned}$$

In the same way as LeafMIO and SplitMIO, one can consider solving problem (20) using a greedy approach, where one iterates through the leaves from highest to lowest revenue, and sets each leaf's $y_{t,\ell}$ variable to the highest possible value without violating any of the constraints. Unlike LeafMIO and SplitMIO, it unfortunately turns out that this greedy approach is not always optimal, which is formalized in the following proposition.

Proposition 8. *There exists an $\mathbf{x} \in [0,1]^n$, a tree t and revenues $\bar{r}_1, \dots, \bar{r}_n$ for which the greedy solution to problem* (20) *is not optimal.*

The proof of Proposition 8 involves an instance where a product appears in more than one split. (Recall that ProductMIO and SplitMIO are equivalent when a product appears at most once in each tree.)

4.4. Overall Benders algorithm

We conclude Section 4 by summarizing how the results are used. In our overall algorithmic approach below, we focus on LeafMIO and SplitMIO, as the subproblem can be solved for these two formulations when $\mathbf{x}$ is either fractional or binary (whereas for ProductMIO, the subproblem can only be solved when $\mathbf{x}$ is binary).

1. *Relaxation phase.* We first solve the relaxed problem (problem (15) for LeafMIO or problem (19) for SplitMIO) using ordinary constraint generation. Given a solution $\mathbf{x} \in [0,1]^n$, we generate Benders cuts by running the primal-dual procedure (either Algorithm 1 followed by Algorithm 2 for LeafMIO, or Algorithm 3 followed by Algorithm 4 for SplitMIO).

2. *Integer phase.* In the integer phase, we add all of the Benders cuts generated in the relaxation phase to the integer version of problem (15) (if solving LeafMIO) or problem (19) (if solving SplitMIO). We then solve the problem as an integer optimization problem, where we generate Benders cuts for integer solutions using the closed form expressions in Section EC.1 of the ecompanion (Theorem EC.1 in Section 4.1 if solving LeafMIO, or Theorem EC.2 in Section 4.2 if solving SplitMIO). In either case, we add these cuts using *lazy constraint generation.* That is, we solve the master problem using a single branch-and-bound tree, and we check whether the main constraint of the Benders formulation (either constraint (15b) for LeafMIO or constraint (19b) for SplitMIO) is violated at every integer solution generated in the branch-and-bound tree.

5. Numerical Experiments with Synthetic Data

In this section, we present the results from our numerical experiments involving synthetically-generated problem instances. Section 5.1 describes how the instances were generated. Section 5.2 presents results on the tightness of the LO relaxation of the three formulations. Section 5.3 presents results on the tractability of the integer version of each formulation. Finally, Section 5.4 compares the Benders approach for SplitMIO with the direct solution approach and with a simple local search heuristic on a collection of large-scale instances. Our experiments were implemented in the Julia programming language, version 0.6.2 (Bezanson et al. 2017) and executed on Amazon Elastic Compute Cloud (EC2) using a single instance of type `r4.4xlarge` (Intel Xeon E5-2686 v4 processor with 16 virtual CPUs and 122 GB memory). All mixed-integer optimization formulations were solved using Gurobi version 8.1 and modeled using the `JuMP` package (Lubin and Dunning 2015).

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

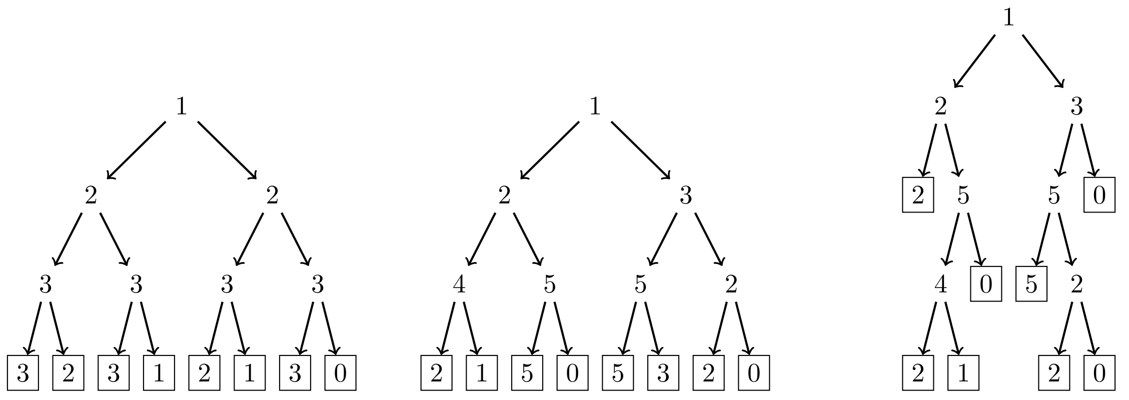
5.1. Background

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

1. **T1 forest**. A T1 forest consists of balanced trees of depth d (i.e., trees where all leaves are at depth $d+1$). For each tree, we sample d products $i_1, \dots, i_d$ uniformly without replacement from $\mathcal{N}$, the set of all products. Then, for every depth $d' \in \{1, \dots, d\}$, we set the split product $v(t,s)$ as $v(t,s) = i_{d'}$ for every split s that is at depth d'.
2. **T2 instances**. A T2 forest consists of balanced trees of depth d. For each tree, we set the split products at each split iteratively, starting at the root, in the following manner:
   (a) Initialize $d' = 1$.
   (b) For all splits s at depth d', set $v(s,t) = i_s$ where i_s is drawn uniformly at random from the set $\mathcal{N} \setminus \cup_{s' \in A(s)} \{v(t,s')\}$, where $A(s)$ is the set of ancestor splits to split s (i.e., all splits appearing on the path from the root node to split s).
   (c) Increment $d' \leftarrow d' + 1$.
   (d) If $d' > d$, stop; otherwise, return to Step (b).
3. **T3 instances**. A T3 forest consists of unbalanced trees with L leaves. Each tree is generated according to the following iterative procedure:
   (a) Initialize t to a tree consisting of a single leaf.
   (b) Select a leaf ℓ uniformly at random from $\mathbf{leaves}(t)$, and replace it with a split s and two child leaves ℓ_1, ℓ_2. For split s, set $v(s,t) = i_s$ where i_s is drawn uniformly at random from $\mathcal{N} \setminus \cup_{s' \in A(s)} \{v(t,s')\}$.
   (c) If $|\mathbf{leaves}(t)| = L$, terminate; otherwise, return to Step (b).

For all three types of forests, we generate the purchase decision $c(t,\ell)$ for each leaf ℓ in each tree t in the following way: for each leaf ℓ, we uniformly at random choose a product $i \in \cup_{s \in \mathbf{LS}(\ell)} \{v(t,s)\} \cup \{0\}$. In words, the purchase decision is chosen to be consistent with the products that are known to be in the assortment if leaf ℓ is reached. Figure 2 shows an example of each type of tree (T1, T2, and T3). Given a forest of any of the three types above, we generate the customer type probability vector $\boldsymbol{\lambda} = (\lambda_t)_{t \in F}$ by drawing it uniformly from the $(|F|-1)-$dimensional unit simplex.

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



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

Figure 2 Examples of T1, T2 and T3 trees.

5.2. Experiment #1: Formulation Strength

Our first experiment is to simply understand how the three formulations – LeafMIO, SplitMIO and ProductMIO– compare in terms of formulation strength. Recall from Propositions 3 and 6 that SplitMIO is at least as strong as LeafMIO, and ProductMIO is at least as strong as SplitMIO. For a given instance and a given formulation $\mathcal{M}$ (one of LeafMIO, SplitMIO and ProductMIO), we define the integrality gap G_F as

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

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

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

Table 1 Average integrality gap of LeafMIO, SplitMIO and ProductMIO for T1, T2 and T3 instances with $n = 100$, $|\mathbf{leaves}(t)| = 8$.

Table 1 displays the average integrality gap of each of the three formulations for each combination of n and $|F|$ and each instance type. From this table, we can see that in general there is an appreciable difference in the integrality gap between LeafMIO, SplitMIO and ProductMIO. In particular, the integrality gap of LeafMIO is in general about 2 to 44%; for SplitMIO, it ranges from 0.2 to 17%; and for ProductMIO, it ranges from 0 to 17%. Note that the difference between ProductMIO and SplitMIO is most pronounced for the T1 instances, as the decision forests in these instances exhibit the highest degree of repetition of products within the splits of a tree. In contrast, the difference is smaller for the T2 and T3 instances, where the trees are balanced but there is less repetition of products within the splits of the tree (as the trees are not forced to have the same product appear on all of the splits at a particular depth).

5.3. Experiment #2: Tractability

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

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

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

Table 2 displays the average computation time and average optimality gap of each formulation for each combination of n, $|F|$ and $|\mathbf{leaves}(t)|$. Due to space considerations, we focus on the T3 instances; results for the T1 and T2 instances are provided in Section EC.4.1 of the ecompanion. From this table, we can see that for the smaller instances, LeafMIO requires significantly more time to solve than SplitMIO, which itself requires more time to solve than ProductMIO. For larger instances, where the computation time limit is exhausted, the average gap obtained by ProductMIO tends to be lower than that of SplitMIO, which is lower than that of LeafMIO.

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

Table 2 Comparison of final optimality gaps and computation times for LeafMIO, SplitMIO and ProductMIO, for T3 instances.

5.4. Experiment #3: Benders Decomposition for Large-Scale Problems

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

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

We compare three different methods: the two-phase Benders method described in Section 4.4, using the SPLITMIO cut results (Section 4.2 and Section EC.1.2 of the ecompanion); the divide-and-conquer (D&C) heuristic; and the direct solution approach, where we attempt to directly solve the full SPLITMIO formulation using Gurobi. The D&C heuristic is a type of local search heuristic proposed in the product line design literature (see Green and Krieger 1993; see also Belloni et al. 2008). In this heuristic, one iterates through the b products currently in the assortment, and replaces a single product with the product outside of the assortment that leads to the highest improvement in the expected revenue; this process repeats until the assortment can no longer be improved. We choose the initial assortment uniformly at random from the collection of assortments of size b. For each instance, we repeat the D&C heuristic 10 times, and retain the best solution. We do not impose a time limit on the D&C heuristic. For the Benders approach, we do not impose a time limit on the LO phase, and impose a time limit of one hour on the integer phase. For the direct solution approach, we impose a time limit of two hours; this time limit was chosen as it exceeded the total solution time used by the Benders approach across all of the instances.

Table 3 reports the performance of the three methods – the Benders approach, the D&C heuristic and direct solution of SPLITMIO– across all combinations of n and ρ. In this table, $Z_{B,LO}$ indicates the objective value of the LO relaxation obtained after the Benders relaxation phase; $Z_{B,UB}$ and $Z_{B,LB}$ indicate the best upper and lower bounds obtained from Gurobi after the Benders integer phase; G_B indicates the final optimality gap of the Benders integer phase, defined as $G_B = (Z_{B,UB} - Z_{B,LB})/Z_{B,UB} \times 100\%$; $Z_{D\&C}$ indicates the objective value of the D&C heuristic; $RI_{D\&C}$ indicates the relative improvement of the final Benders solution over the D&C solution, defined as $RI_{D\&C} = (Z_{B,LB} - Z_{D\&C})/Z_{D\&C} \times 100\%$; Z_{Direct} indicates the best lower bound obtained from directly solving SPLITMIO; and RI_{Direct} indicates the relative improvement of the final Benders solution over the final solution obtained from directly solving SPLITMIO. The value reported of each metric is the average over the five replications corresponding to the particular (n, ρ) combination.

In addition to the comparison of the objective values obtained by the three methods, it is also useful to compare the methods by computation time. Table 4 displays the computation time required for all three methods. In this table, $T_{B,LO}$ indicates the time required by the LO relaxation phase of the Benders approach; $T_{B,IO}$ indicates the time required by the integer phase of the Benders approach; $T_{B,Total}$ indicates the total time of the Benders approach (i.e., $T_{B,LO} + T_{B,IO}$); $T_{D\&C}$ indicates the time required for the D&C heuristic; and T_{Direct} indicates the time required for the direct solution approach, i.e., solving SPLITMIO directly using Gurobi. For all of these metrics, we report the average over the five replications for each combination of n and ρ. In addition, the table also reports the metric NU_{Direct}, which is the number of instances for which the direct solution approach terminated without an upper bound (in other words, the LO relaxation of SPLITMIO was not solved within the two hour time limit).

Comparing the performance of the Benders approach with the D&C heuristic, we can see that in general, the Benders approach is able to find better solutions than the D&C heuristic. The performance gap, as indicated by the $RI_{D\&C}$ metric, can be substantial: with $n = 3000$ and $\rho = 0.06$, the Benders solution achieves an objective value that is on average more than 5% higher than that of the D&C heuristic's solution. In addition, from a computation time standpoint, the Benders

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

Table 3 Comparison of the Benders decomposition approach, the D&C heuristic and direct solution of SplitMIO in terms of solution quality.

approach compares quite favorably to the D&C heuristic. While the D&C heuristic is faster for small problems with low n and/or low ρ, it can require a significant amount of time for $n = 2000$ or $n = 3000$. In addition to this comparison against the D&C heuristic, in Section EC.4.2 of the electronic companion, we also provide a comparison of the MIO solutions for the smaller T1, T2 and T3 instances used in the previous two sections against heuristic solutions; in those instances, we similarly find that solutions obtained from our MIO formulations can be significantly better than heuristic solutions.

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

The second type of behavior corresponds to "hard" instances, which are the instances with $\rho \in \{0.06, 0.08\}$. For these instances, when one applies the direct approach, Gurobi is not able to solve the LO relaxation of SPLITMIO within the two hour time limit for any instance (see the NU_{Direct} column of Table 4). In those instances, the integer solution returned by Gurobi is obtained from applying heuristics before solving the root node of the branch-and-bound tree, which is often quite suboptimal. In contrast, the Benders approach delivers significantly better solutions. In particular, as indicated by the RI_{Direct} column, for $n \in \{1000, 2000, 3000\}$, the Benders solution

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

Table 4 Comparison of the Benders decomposition approach, the D&C heuristic and direct solution of SplitMIO in terms of solution time.

can achieve an objective value that is anywhere from 177% to 621% better, on average, than the solution obtained by Gurobi. It is also interesting to note that while Gurobi is not able to solve the LO relaxation within the two hour time limit, our Benders method solves it quickly; in the largest case, the solution time for the relaxation is no more than about 1500 seconds, or roughly 25 minutes. This highlights another benefit of our Benders approach, which is that it is capable of solving problems that are simply too large to be directly solved using a solver like Gurobi.

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

6. Conclusions

In this paper, we have developed a mixed-integer optimization methodology for solving the assortment optimization problem when the choice model is a decision forest model. This methodology allows a firm to find optimal or near optimal assortments given a decision forest model, which is valuable due to the ability of the decision forest model to capture non-rational customer behavior. We developed three different formulations of increasing strength, which generalize existing formulations in the literature on assortment optimization/product line design under ranking preferences, and also have interesting connections to other integer optimization models. We analyzed the solvability of the Benders decomposition subproblem for each, showing that for two of our formulations it is possible to solve the primal and dual subproblems using a greedy algorithm when the master solution is fractional, and that for all three formulations it is possible to solve primal and dual subproblems in closed form when the master solution is binary. Using synthetic data, we showed that our formulations can be solved directly to optimality or near optimality at a small to medium scale, and using our Benders approach, we show that it is possible to solve large instances with up to 3000 products to a low optimality gap within an operationally feasible timeframe.

There are at least two directions of future research that are interesting to pursue. First, there is a rich literature on approximation algorithms for assortment optimizations under a variety of choice models and in particular, ranking-based models (Aouad et al. 2018, Feldman et al. 2019); an interesting direction is to understand whether one can use the solutions of the relaxations of our formulations within a randomized rounding procedure to obtain assortments with an approximation guarantee. Second, there is a growing literature on robust assortment optimization (Rusmevichientong and Topaloglu 2012, Bertsimas and Mišić 2017, Désir et al. 2019, Wang et al. 2020); it would thus be interesting to consider robust versions of our formulations, where one optimizes against the worst-case revenue over an uncertainty set of decision forest models.

References

L. Alfandari, A. Hassanzadeh, and I. Ljubić. An exact method for assortment optimization under the nested logit model. *European Journal of Operational Research*, 291(3):830–845, 2021.

A. Aouad, V. Farias, R. Levi, and D. Segev. The approximability of assortment optimization under ranking preferences. *Operations Research*, 66(6):1661–1669, 2018.

A. Aouad, V. Farias, and R. Levi. Assortment optimization under consider-then-choose choice models. *Management Science*, 2020.

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

G. Berbeglia. The generalized stochastic preference choice model. *arXiv preprint arXiv:1803.04244*, 2018.

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

D. Bertsimas and V. V. Mišić. Exact first-choice product line optimization. *Operations Research*, 67(3): 651–670, 2019.

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. 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. Mathematical properties of the optimal product line selection problem using choice-based conjoint analysis. *Management Science*, 46(2):327–332, 2000.

Y.-C. Chen and V. V. Mišić. Decision forest: A nonparametric approach to modeling irrational choice. *arXiv preprint arXiv:1904.11532*, 2019.

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, F. Furini, and I. Ljubić. Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research*, 275(3):882–896, 2019.

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

A. Désir, V. Goyal, B. Jiang, T. Xie, and J. Zhang. A nonconvex min-max framework and its applications to robust assortment optimization. *Working paper, INSEAD, Fontainebleau*, 2019.

A. Désir, V. Goyal, D. Segev, and C. Ye. Constrained assortment optimization under the markov chain–based choice model. *Management Science*, 66(2):698–721, 2020.

F. Echenique and K. Saito. General luce model. *Economic Theory*, 68(4):811–826, 2019.

J. Feldman, A. Paul, and H. Topaloglu. Assortment optimization with small consideration sets. *Operations Research*, 67(5):1283–1299, 2019.

J. B. Feldman and H. Topaloglu. Revenue management under the markov chain choice model. *Operations Research*, 65(5):1322–1342, 2017.

M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning benders decomposition for large-scale facility location. *Management Science*, 63(7):2146–2162, 2017.

A. Flores, G. Berbeglia, and P. Van Hentenryck. Assortment and price optimization under the two-stage luce model. *arXiv preprint arXiv:1706.08599*, 2017.

G. Gallego and H. Topaloglu. *Assortment Optimization*, pages 129–160. Springer New York, New York, NY, 2019.

G. Gallego, R. Ratliff, and S. Shebalov. A general attraction model and sales-based linear program for network revenue management under customer choice. *Operations Research*, 63(1):212–232, 2015.

M. R. Garey and D. S. Johnson. *Computers and intractability*. W. H. Freeman New York, 1979.

P. E. Green and A. M. Krieger. Conjoint analysis with product-positioning applications. *Handbooks in operations research and management science*, 5:467–515, 1993.

J. Huchette and J. P. Vielma. A combinatorial approach for small and strong formulations of disjunctive constraints. *Mathematics of Operations Research*, 44(3):793–820, 2019.

R. Kivetz, O. Netzer, and V. Srinivasan. Extending compromise effect models to complex buying situations and other context effects. *Journal of Marketing Research*, 41(3):262–268, 2004.

M. Lubin and I. Dunning. Computing in operations research using julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

S. Martello and P. Toth. Algorithms for knapsack problems. In *North-Holland Mathematics Studies*, volume 132, pages 213–257. Elsevier, 1987.

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.

I. Méndez-Díaz, J. J. Miranda-Bront, G. Vulcano, and P. Zabala. A branch-and-cut algorithm for the latent-class logit assortment problem. *Discrete Applied Mathematics*, 164:246–263, 2014.

V. V. Mišić. Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624, 2020.

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.

R. P. Rooderkerk, H. J. Van Heerde, and T. H. A. Bijmolt. Incorporating context effects into a choice model. *Journal of Marketing Research*, 48(4):767–780, 2011.

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

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

A. Şen, A. Atamtürk, and P. Kaminsky. A conic integer optimization approach to the constrained assortment problem under the mixed multinomial logit model. *Operations Research*, 66(4):994–1003, 2018.

M. Sumida, G. Gallego, P. Rusmevichientong, H. Topaloglu, and J. Davis. Revenue-utility tradeoff in assortment optimization under the multinomial logit model with totally unimodular constraints. *Management Science*, 2020.

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

J. P. Vielma and G. L. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128(1-2):49–72, 2011.

J. P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, 58(2):303–315, 2010.

Z. Wang, H. Peura, and W. Wiesemann. Randomized assortment optimization. *Available at SSRN 3685695*, 2020.

EC.1. Benders cuts for integer master solutions

In this section, we provide closed form expressions for the structure of the optimal primal and dual Benders subproblem solutions for integer solutions $\mathbf{x}$, for LeafMIO (Section EC.1.1), SplitMIO (Section EC.1.2) and ProductMIO (Section EC.1.3).

EC.1.1. Benders cuts for integer solutions of LeafMIO

Our results in Section 4.1 for obtaining primal and dual solutions for the subproblem of LeafMIO apply for any $\mathbf{x} \in [0,1]^n$; in particular, they apply for fractional choices of $\mathbf{x}$, thus allowing us to solve the Benders reformulation of the relaxation of LeafMIO (presented as problem (15)).

In the special case that $\mathbf{x}$ is integer, the optimal solutions to the primal and dual subproblems can be obtained more directly than by applying Algorithms 1 and 2; more specifically, they can be obtained in closed form. We formalize this as the following theorem.

Theorem EC.1. *Fix $t \in F$, and let $\mathbf{x} \in \{0,1\}^n$. Define the primal subproblem solution $\mathbf{y}_t$ as*

$$y_{t,\ell} = \begin{cases} 1 & \text{if } \ell = \ell^*, \\ 0 & \text{if } \ell \neq \ell^*, \end{cases}$$

where ℓ^* denotes the leaf of tree t that the assortment encoded by $\mathbf{x}$ is mapped to. Define the dual subproblem solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ as

$$\alpha_{t,s,\ell} = \begin{cases} \max\{0, r_{t,\ell} - r_{t,\ell^*}\} & \text{if } s \in \mathbf{RS}(\ell^*),\ \ell \in \mathbf{left}(s), \\ 0 & \text{otherwise}, \end{cases}$$

$$\beta_{t,s,\ell} = \begin{cases} \max\{0, r_{t,\ell} - r_{t,\ell^*}\} & \text{if } s \in \mathbf{LS}(\ell^*),\ \ell \in \mathbf{right}(s), \\ 0 & \text{otherwise}, \end{cases}$$

$$\gamma_t = r_{t,\ell^*}.$$

Then:

a) $\mathbf{y}_t$ is a feasible solution to problem (13);

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to problem (14); and

c) $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for problems (13) and (14), respectively.

The significance of Theorem EC.1 is that it provides a simpler means to checking for violated constraints when $\mathbf{x}$ is binary than applying Algorithms 1 and 2. In particular, for the integer version of LeafMIO, a similar derivation as in Section 4.1 leads us to the following Benders reformulation of the integer problem for the LeafMIO formulation:

$$\underset{\mathbf{x},\boldsymbol{\theta}}{\text{maximize}} \quad \sum_{t \in F} \lambda_t \theta_t \tag{EC.1a}$$

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

$$\forall\ (\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t) \in \mathcal{D}_{t,\text{LeafMIO}}, \tag{EC.1b}$$

$$\mathbf{x} \in \{0,1\}^n. \tag{EC.1c}$$

To check whether constraint (EC.1b) is violated for a particular $\mathbf{x}$ and a tree t, we can simply use Theorem EC.1 to determine the optimal value of the subproblem, and compare it against θ_t; if the constraint corresponding to the dual solution of Theorem EC.1 is violated, we add that constraint to the problem. In our implementation of Benders decomposition, we embed the constraint generation process for the integer problem (EC.1) within the branch-and-bound tree, using a technique referred to as *lazy constraint generation*; we discuss this more in Section 4.4.

EC.1.2. Benders cuts for integer solutions of SplitMIO

We next turn our attention to SplitMIO. In the special case that $\mathbf{x}$ is a candidate integer solution of SplitMIO, we can find optimal solutions to the primal and dual subproblems of SplitMIO in closed form, analogously to Theorem EC.1 for the primal and dual subproblems of LeafMIO.

Theorem EC.2. *Fix $t \in F$, and let $\mathbf{x} \in \{0,1\}^n$. Define the primal subproblem solution $\mathbf{y}_t$ as*

$$y_{t,\ell} = \begin{cases} 1 \text{ if } \ell = \ell^*, \\ 0 \text{ if } \ell \neq \ell^*, \end{cases}$$

where ℓ^* denotes the leaf that the assortment encoded by $\mathbf{x}$ is mapped to. Define the dual subproblem solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ as

$$\alpha_{t,s} = \begin{cases} \max\{0, \max_{\ell \in \mathbf{left}(s)} r_{t,\ell} - r_{t,\ell^*}\} & \text{if } s \in \mathbf{RS}(\ell^*), \\ 0 & \text{otherwise}, \end{cases}$$
$$\beta_{t,s} = \begin{cases} \max\{0, \max_{\ell \in \mathbf{right}(s)} r_{t,\ell} - r_{t,\ell^*}\} & \text{if } s \in \mathbf{LS}(\ell^*), \\ 0 & \text{otherwise}, \end{cases}$$
$$\gamma_t = r_{t,\ell^*}.$$

Then:

a) $\mathbf{y}_t$ is a feasible solution to problem (17);
b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to problem (18); and
c) $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for problems (17) and (18), respectively.

As with Theorem EC.1, the significance of this theorem is that it provides an even simpler approach than Algorithms 3 and 4 for identifying violated constraints when dealing with integer solutions $\mathbf{x}$.

EC.1.3. Benders cuts for integer solutions of ProductMIO

We finally consider ProductMIO. We begin by writing down the dual of the subproblem, for which we need to define several additional sets. We let $\mathbf{LP}(\ell)$ denote the set of "left products" of leaf ℓ (those products that must be included in the assortment for leaf ℓ to be reached), and let $\mathbf{RP}(\ell)$ denote the set of "right products" of leaf ℓ (those products that must be excluded from the assortment for leaf ℓ to be reached). Note that $\ell \in \mathbf{left}(i)$ if and only if $i \in \mathbf{LP}(\ell)$, and similarly $\ell \in \mathbf{right}(i)$ if and only if $i \in \mathbf{RP}(\ell)$.

With these definitions, the dual of the primal subproblem (20) is

$$\begin{aligned} \underset{\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t}{\text{minimize}} \quad & \sum_{i \in P(t)} \alpha_{t,i} x_i + \sum_{i \in P(t)} \beta_{t,i}(1 - x_i) + \gamma_t && \text{(EC.2a)} \\ \text{subject to} \quad & \sum_{i \in \mathbf{LP}(\ell)} \alpha_{t,i} + \sum_{i \in \mathbf{RP}(\ell)} \beta_{t,i} + \gamma_t \geq r_\ell, \quad \forall\, \ell \in \mathbf{leaves}(t), && \text{(EC.2b)} \\ & \alpha_{t,i} \geq 0, \quad \forall\, i \in P(t), && \text{(EC.2c)} \\ & \beta_{t,i} \geq 0, \quad \forall\, i \in P(t). && \text{(EC.2d)} \end{aligned}$$

In the case that $\mathbf{x}$ is integer, we can obtain optimal solutions to the primal subproblem (20) and its dual (EC.2) in closed form.

Theorem EC.3. *Fix $t \in F$ and let $\mathbf{x} \in \{0,1\}^n$. Let $\mathbf{y}_t$ be defined as*

$$y_{t,\ell} = \begin{cases} 1 \text{ if } \ell = \ell^*, \\ 0 \text{ otherwise}, \end{cases} \tag{EC.3}$$

and let $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *be defined as*

$$\alpha_{t,i} = \begin{cases} \max\{0, \max_{\ell \in \mathbf{left}(i)} r_{t,\ell} - r_{t,\ell^*}\}, & \text{if } i \in \mathbf{RP}(\ell^*), \\ 0 & \text{otherwise,} \end{cases} \tag{EC.4}$$

$$\beta_{t,i} = \begin{cases} \max\{0, \max_{\ell \in \mathbf{right}(i)} r_{t,\ell} - r_{t,\ell^*}\}, & \text{if } i \in \mathbf{LP}(\ell^*), \\ 0 & \text{otherwise,} \end{cases} \tag{EC.5}$$

$$\gamma_t = r_{t,\ell^*}. \tag{EC.6}$$

Then:

a) $\mathbf{y}_t$ *is a feasible solution for problem* (20)*;*

b) $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *is a feasible solution for problem* (EC.2)*; and*

c) $\mathbf{y}_t$ *and* $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ *are optimal for problems* (20) *and* (EC.2)*, respectively.*

EC.2. Proofs

EC.2.1. Proof of NP-Hardness

We prove this by showing that MAX 3SAT problem reduces to the decision forest assortment optimization problem. In the MAX 3SAT problem, one has K binary variables, $x_1, \ldots, x_K$, and is given a Boolean formula of the form $c_1 \wedge c_2 \wedge \cdots \wedge c_M$, where $\wedge$ denotes "and". Each clause is a disjunction involving three literals, where a literal is either one of the binary variables or its negation, and the literals involve distinct binary variables. For example, a clause could be $x_5 \vee \neg x_7 \vee x_8$, where $\vee$ denotes "or". The MAX 3SAT problem is to find the assignment of the variables $x_1, \ldots, x_K$ so as to maximize the number of clauses $c_1, \ldots c_T$ that are true.

Given an instance of the MAX 3SAT problem, we show how the problem can be transformed into an instance of the decision forest assortment optimization problem (2).

Consider an instance of problem (2) with $n = K + 1$ products. Each of the first K products corresponds to one of the binary variables; the last $K + 1$th product is necessary to ensure that the revenue of the assortment can correspond to the number of satisfied clauses. Assume that the marginal revenues of the products are set so that $\bar{r}_1 = \cdots = \bar{r}_K = 0$, and $\bar{r}_{K+1} = 1$. For each clause $m \in \{1, \ldots, M\}$, introduce a tree t_m which is constructed by the following procedure:

1. Set the root node of the tree to be a split node involving product $K + 1$. The right child of the root node is a leaf node with 0 (the no-purchase option) as the purchase decision. (The left child node will be defined in the next step.)
2. For the first literal, create a split at the left child node of the root, with the corresponding binary variable's index as the product (e.g., if the literal is x_7 or $\neg x_7$, the split node has product 7). If the literal is the binary variable itself (i.e., x_k), we set the left child node of the split to be a leaf node with product $K + 1$ as the purchase decision. Otherwise, if the literal is the negation of the binary variable (i.e., $\neg x_k$), we set the right child node of the split to be a leaf node with product $K + 1$ as the purchase decision.
3. For the other child node of the split created in Step 2, repeat Step 2 with the second literal.
4. For the other child node of the split created in Step 3, repeat Step 2 with the third literal.
5. Lastly, for the other child node of the third split node in Step 4 corresponding to the third literal, set the purchase decision to be 0.

Figure EC.1 visualizes the structure of the tree for the example clause $x_5 \vee \neg x_7 \vee x_8$. Applying the above procedure for each clause results in a forest F consisting of M trees. Lastly, we set the probability distribution $\boldsymbol{\lambda}$ by setting $\lambda_t = 1/M$ for each tree $t \in F$.

We note that in the resulting instance of problem (2), any optimal assortment must include produce $K+1$: due to the structure of the trees, the expected revenue is exactly equal to 0 if $K+1$ is not included in the assortment, but by including $K+1$ and including/excluding products from $\{1, \ldots, K\}$ in accordance with one of the clauses, one can obtain an expected revenue of at least

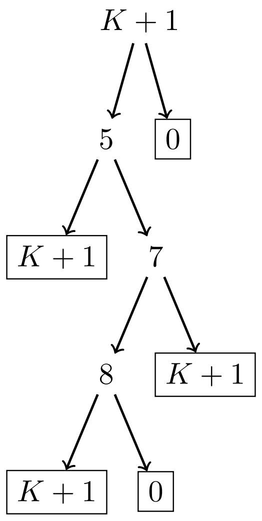


Figure EC.1 Example of tree that corresponds to the clause $x_5 \vee \neg x_7 \vee x_8$.

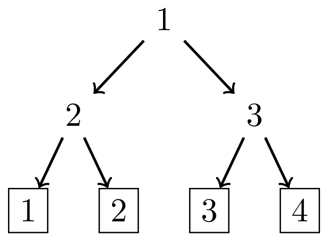


Figure EC.2 Purchase decision tree for proof of Proposition 2. The numbers inside the split nodes indicate the split product (i.e., $v(t,s)$). The numbers inside the leaf nodes (enclosed in squares) index the leaves (i.e., the leaves are numbered from 1 to 4).

$1/M$. (For example, if the set of clauses includes the clause $x_5 \vee \neg x_7 \vee x_8$ shown in Figure EC.1, then any assortment S that includes products 5 and 8 and does not include product 7 automatically has an expected revenue of at least $1/M$.)

Note that given an optimal assortment $S \subseteq \mathcal{N} = \{1, \ldots, K+1\}$, we immediately obtain an assignment $\mathbf{x}$ for the MAX 3SAT problem by setting $x_i = \mathbb{I}\{i \in S\}$. The revenue obtained from each tree t_m corresponding to clause m, which is given by $\sum_{j=1}^{K+1} \bar{r}_j \mathbb{I}\{\hat{A}(t_m, S) = j\}$, is exactly 1 if the assignment $\mathbf{x}$ that corresponds to S satisfies clause m, and 0 otherwise; this holds because the optimal assortment must include product $K+1$. Since each tree t_m has a probability of $1/M$, the expected revenue of the assortment S therefore corresponds to the fraction of the M clauses that are satisfied by the assignment $\mathbf{x}$. Thus, it follows that an assignment $\mathbf{x}$ that corresponds to an optimal assortment S is an optimal solution of MAX 3SAT. Since the MAX 3SAT problem is NP-Complete (Garey and Johnson 1979), it follows that problem (2) is NP-Hard. □

EC.2.2. Proof of Proposition 2

Consider a decision forest consisting of only one tree, of the form shown in Figure EC.2. The feasible region of the LO relaxation of problem (4) is the set of solutions $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^2 \times \mathbb{R}^4$ given by the following family of constraints:

$$y_1 \leq x_1, \tag{EC.7a}$$
$$y_1 \leq x_2, \tag{EC.7b}$$
$$y_2 \leq x_1, \tag{EC.7c}$$

$$y_2 \leq 1 - x_2, \tag{EC.7d}$$
$$y_3 \leq 1 - x_1, \tag{EC.7e}$$
$$y_3 \leq x_3, \tag{EC.7f}$$
$$y_4 \leq 1 - x_1, \tag{EC.7g}$$
$$y_4 \leq 1 - x_3, \tag{EC.7h}$$
$$y_1 + y_2 + y_3 + y_4 = 1, \tag{EC.7i}$$
$$x_1 \leq 1, \tag{EC.7j}$$
$$x_2 \leq 1, \tag{EC.7k}$$
$$x_3 \leq 1, \tag{EC.7l}$$
$$x_1, x_2, x_3, y_1, y_2, y_3, y_4 \geq 0. \tag{EC.7m}$$

(Note that for simplicity, we drop the subscript t from all of the y variables.)

It can be verified that the following solution is an extreme point of this polyhedron:

$$x_1 = 0.5, \tag{EC.8a}$$
$$x_2 = 0.5, \tag{EC.8b}$$
$$x_3 = 0, \tag{EC.8c}$$
$$y_1 = 0.5, \tag{EC.8d}$$
$$y_2 = 0.5, \tag{EC.8e}$$
$$y_3 = 0, \tag{EC.8f}$$
$$y_4 = 0. \tag{EC.8g}$$

Since this extreme point $(\mathbf{x}, \mathbf{y})$ does not satisfy $\mathbf{x} \in \{0,1\}^3$, we conclude that $\mathcal{F}_{\text{LeafMIO}}$ is not integral in general. $\square$

EC.2.3. Proof of Proposition 4

The feasible region $\mathcal{F}_{\text{SplitMIO}}$ of the LO relaxation of problem (4) is the set of $(\mathbf{x}, \mathbf{y})$ solutions to the following system of inequalities:

$$\sum_{\ell \in \mathbf{leaves}} y_\ell \leq 1, \tag{EC.9a}$$
$$\sum_{\ell \in \mathbf{leaves}} -y_\ell \leq -1, \tag{EC.9b}$$
$$\sum_{\ell \in \mathbf{left}(s)} y_\ell - x_{v(s)} \leq 0, \quad \forall\, s \in \mathbf{splits}, \tag{EC.9c}$$
$$\sum_{\ell \in \mathbf{right}(s)} y_\ell + x_{v(s)} \leq 1, \quad \forall\, s \in \mathbf{splits}, \tag{EC.9d}$$
$$x_i \leq 1, \quad \forall\, i \in \mathcal{N}, \tag{EC.9e}$$
$$x_i \geq 0, \quad \forall\, i \in \mathcal{N}, \tag{EC.9f}$$
$$y_\ell \geq 0, \quad \forall\, \ell \in \mathbf{leaves}. \tag{EC.9g}$$

(Since $|F| = 1$, we drop the index t to simplify the notation.) In the above, note that the unit sum constraint on $\mathbf{y}$ has been re-written as a pair of inequalities, and that all constraints from problem (4) have been re-arranged to have the variables on one side. This system of inequalities can be further written compactly as

$$\mathbf{A} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \leq \mathbf{b}, \tag{EC.10}$$
$$\mathbf{x}, \mathbf{y} \geq \mathbf{0}. \tag{EC.11}$$

To show that $\mathcal{F}_{\text{SplitMIO}}$ is integral, we will prove that the matrix $\mathbf{A}$ is totally unimodular. We do so using the following standard characterization of total unimodularity (see Bertsimas and Weismantel 2005):

Proposition EC.1 **(Corollary 3.2 of 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 part is a vector with entries only 0, +1, and -1.*

To simplify our notation, we will work with algebraic expressions in terms of $\mathbf{x}$ and $\mathbf{y}$ rather than rows of the matrix $\mathbf{A}$. We have the following four primitive expressions:

$$A(s), s \in \mathbf{splits}: \quad \sum_{\ell \in \mathbf{left}(s)} y_\ell - x_{v(s)}, \tag{EC.12}$$

$$B(s), s \in \mathbf{splits}: \quad \sum_{\ell \in \mathbf{right}(s)} y_\ell + x_{v(s)}, \tag{EC.13}$$

$$C(i), i \in \mathcal{N}: \quad x_i, \tag{EC.14}$$

$$D(1): \quad \sum_{\ell \in \mathbf{leaves}} y_\ell, \tag{EC.15}$$

$$D(2): \quad \sum_{\ell \in \mathbf{leaves}} -y_\ell. \tag{EC.16}$$

Thus, a collection of rows Q of the matrix $\mathbf{A}$ can be viewed as a collection of each of the four types of expressions above:

$$S_A \subseteq \mathbf{splits}, \tag{EC.17}$$

$$S_B \subseteq \mathbf{splits}, \tag{EC.18}$$

$$S_C \subseteq \mathcal{N}, \tag{EC.19}$$

$$S_D \subseteq \{1, 2\}. \tag{EC.20}$$

To establish the condition in Proposition EC.1, we need to show that given S_A, S_B, S_C, S_D, we can partition these expressions into two groups R_+ and R_- such that the difference of the two groups,

$$\sum_{e \in R_+} e - \sum_{e \in R_-} e = \sum_{i \in \mathcal{N}} v_i x_i + \sum_{\ell \in \mathbf{leaves}} w_\ell y_\ell, \tag{EC.21}$$

is such that each $v_i \in \{-1, 0, +1\}$ and each $w_\ell \in \{-1, 0, +1\}$. We proceed in several steps.

Step 1. We begin by assigning the A and B expressions. Before doing so, we require some additional notation. Define $d^* = \max_{s \in S_A \cup S_B} d(s)$, where $d(s)$ is the depth of split s and we assume the depth of the root node is 1. Let us define the sets $S_A(d)$ and $S_B(d)$ as

$$S_A(d) = \{s \in S_A \mid d(s) = d\}, \tag{EC.22}$$

$$S_B(d) = \{s \in S_B \mid d(s) = d\}, \tag{EC.23}$$

for each depth $d \in \{1, \ldots, d^*\}$. These are the sets of splits in S_A and S_B, respectively, that are at a particular depth. Let us also define $\mathbf{LD}(s)$ and $\mathbf{RD}(s)$ to be the sets of splits in $S_A \cup S_B$ that are to the left and right, respectively, of split $s \in S_A \cup S_B$. (The splits in $\mathbf{LD}(s)$ are all those that can be reached by proceeding to the left child of split s; similarly, $\mathbf{RD}(s)$ is the set of splits reachable by going to the right of split s.) Finally, define $\sigma : S_A \cup S_B \to \{-1, +1\}$ to be a mapping that is specified according to the following procedure:

```
for $d = 1, \dots, d^*$ do
    for $s \in S_A(d)$ do
        Set $\sigma(s') = (-1)\sigma(s)$ for $s' \in \mathbf{LD}(s)$
    end for
    for $s \in S_B(d)$ do
        Set $\sigma(s') = (-1)\sigma(s)$ for $s' \in \mathbf{RD}(s)$
    end for
end for
```

Now, assign the A and B expressions as follows:

- Assign $A(s)$ to R_+ for each $s \in S_A$ with $\sigma(s) = +1$;
- Assign $A(s)$ to R_- for each $s \in S_A$ with $\sigma(s) = -1$;
- Assign $B(s)$ to R_+ for each $s \in S_B$ with $\sigma(s) = +1$;
- Assign $B(s)$ to R_- for each $s \in S_B$ with $\sigma(s) = -1$.

For this assignment of the expressions in S_A and S_B, every y_ℓ coefficient in $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ will be either 0 or +1. This follows because the sets of left and right leaves $\mathbf{left}(s)$ and $\mathbf{right}(s)$ are nested. In particular, if $s' \in \mathbf{LD}(s)$, then we will have that $\mathbf{left}(s') \subseteq \mathbf{left}(s)$ and $\mathbf{right}(s') \subseteq \mathbf{left}(s)$. Similarly, if $s' \in \mathbf{RD}(s)$, then we will have that $\mathbf{left}(s') \subseteq \mathbf{right}(s)$ and $\mathbf{right}(s') \subseteq \mathbf{right}(s)$.

In addition, by the assumption that there is at most one split s in **splits** such that $v(s) = i$, we are also guaranteed that the coefficient of every x_i in $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ will be $\{-1, 0, +1\}$. In particular, if s is in both S_A and S_B (i.e., we were given the expression $A(s)$ and $B(s)$), then observe that by the procedure for setting σ above, we are guaranteed to assign both $A(s)$ and $B(s)$ to the same set (they cannot be assigned to different sets). This means that the coefficients of $x_{v(s)}$ in $A(s)$ and $B(s)$ will cancel out, leaving $x_{v(s)}$ with a coefficient of 0.

Step 2. We next assign the C expressions. After Step 1, we are guaranteed that the coefficient of each x_i in $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ is 0, -1 or +1. Since each $C(i)$ expression involves only one variable (x_i), it is straightforward to assign these expressions to R_+ and R_- to ensure that every variable's coefficient in $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ is 0, -1 or +1. For completeness, we give the procedure below – for each $i \in S_C$:

- If $v(s) \neq i$ for all splits $s \in \mathbf{splits}$, then x_i does not appear in any A or B expressions and its coefficient after Step 1 is just 0; thus, $C(i)$ can be arbitrarily assigned to R_+ or R_-.
- If there exists an $s \in S_A \cup S_B$ such that $v(s) = i$, then:
  - —If $s \in S_A \cup S_B$, then $A(s)$ and $B(s)$ were both assigned to R_+ and R_-, and so the coefficient of x_i will be 0 due to cancellation; thus, $C(i)$ can again be arbitrarily assigned to R_+ or R_-.
  - —If $s \in S_A$, $s \notin S_B$, and $\sigma(s) = +1$, then the coefficient of x_i is -1 after Step 1; thus, $C(i)$ should be assigned to R_+.
  - —If $s \in S_A$, $s \notin S_B$, and $\sigma(s) = -1$, then the coefficient of x_i is +1 after Step 1; thus, $C(i)$ should be assigned to R_-.
  - —If $s \notin S_A$, $s \in S_B$, and $\sigma(s) = +1$, then the coefficient of x_i is +1 after Step 1; thus, $C(i)$ should be assigned to R_-.
  - —If $s \notin S_A$, $s \in S_B$, and $\sigma(s) = -1$, then the coefficient of x_i is -1 after Step 1; thus, $C(i)$ should be assigned to R_+.

Step 3. Lastly, we assign the D expressions. This step is also straightforward:

- If $S_D = \{1, 2\}$, then assign $D(1)$ and $D(2)$ to R_+; since $D(1)$ is just the negative of $D(2)$, the two expressions will cancel out, and the expression $\sum_{e \in R_+} e - \sum_{e \in R_-} e$ will remain unchanged.
- If $S_D = \{1\}$, then assign $D(1)$ to R_-; since the coefficient of each y_ℓ is 0 or +1 after Step 2, this will ensure that the coefficient of each y_ℓ is either -1 or 0.
- If $S_D = \{2\}$, then assign $D(2)$ to R_-; since the coefficient of each y_ℓ is 0 or +1 after Step 2, this will ensure that the coefficient of each y_ℓ is either -1 or 0.

After completing Step 3, we have assigned all of the expressions in S_A, S_B, S_C, S_D to the sets R_+ and R_- in a way that each expression is assigned to exactly one of the two sets, and no expression

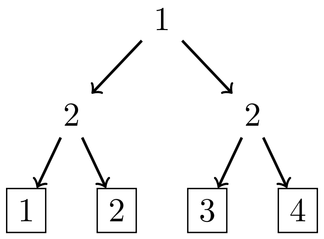


Figure EC.3 Purchase decision tree for proof of Proposition 5. The numbers on the split nodes correspond to products that are used for splitting (i.e., $v(t,s)$). The numbers inside the leaf nodes (enclosed in squares) index the leaves (i.e., the leaves are indexed from 1 to 4).

is unassigned. Moreover, the difference of the two expressions, $\sum_{e\in R_+} e - \sum_{e\in R_-} e$, is such that the coefficient of every x_i and y_ℓ variable is in $\{0,-1,+1\}$. By Proposition EC.1, this establishes that the matrix $\mathbf{A}$ is totally unimodular. We now employ another standard result:

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

To use this result, we simply have to establish that the feasible region of the polyhedron defined in (EC.9) is nonempty.

To do so, we explicitly construct a feasible solution to (EC.9). Let r be the root node of the tree. Set $x_i = 0.5$ for all $i\in\mathcal{N}$. Fix any leaf $\ell'\in\mathbf{left}(r)$ and any leaf $\ell''\in\mathbf{right}(r)$, and set $y_{\ell'}=0.5$, $y_{\ell''}=0.5$, and $y_\ell = 0$ for all $\ell\in\mathbf{leaves}\setminus\{\ell',\ell''\}$. It is straightforward to verify that this solution satisfies the system of inequalities (EC.9): the y_ℓ's sum to 1 and are nonnegative by construction, and each $x_i\in[0,1]$ by construction. For constraints (EC.9c) and (EC.9d), note that since ℓ' and ℓ'' are on opposite sides of the root node, it is impossible for ℓ' and ℓ'' to both belong to $\mathbf{left}(s)$ and $\mathbf{right}(s)$ for any split s; armed with this fact, it is straightforward to establish the two constraints.

Since we have established that $\mathbf{A}$ is totally unimodular and that the set $\mathcal{F}_{\text{SPLITMIO}}$ is nonempty, invoking Proposition EC.2 concludes the proof. □

EC.2.4. Proof of Proposition 5

Consider a decision forest consisting of only $|F|=1$ tree, of the form shown in Figure EC.3. The feasible region of the LO relaxation of problem (4) is the set of solutions $(\mathbf{x},\mathbf{y})\in\mathbb{R}^2\times\mathbb{R}^4$ given by the following family of constraints (where we again drop the subscript t for simplicity):

$$y_1 \le x_2, \tag{EC.24a}$$
$$y_2 \le 1 - x_2, \tag{EC.24b}$$
$$y_3 \le x_2, \tag{EC.24c}$$
$$y_4 \le 1 - x_2, \tag{EC.24d}$$
$$y_1 + y_2 \le x_1, \tag{EC.24e}$$
$$y_3 + y_4 \le 1 - x_1, \tag{EC.24f}$$
$$y_1 + y_2 + y_3 + y_4 = 1, \tag{EC.24g}$$
$$x_1 \le 1, \tag{EC.24h}$$
$$x_2 \le 1, \tag{EC.24i}$$
$$x_1, x_2, y_1, y_2, y_3, y_4 \ge 0. \tag{EC.24j}$$

It can be verified that the following solution is an extreme point of this polyhedron:

$$x_1 = 0.5, \tag{EC.25a}$$

$$x_2 = 0.5, \tag{EC.25b}$$
$$y_1 = 0.5, \tag{EC.25c}$$
$$y_2 = 0, \tag{EC.25d}$$
$$y_3 = 0, \tag{EC.25e}$$
$$y_4 = 0.5. \tag{EC.25f}$$

Since this extreme point is not integer, this establishes that even when $|F| = 1$, $\mathcal{F}_{\text{SplitMIO}}$ can be non-integral when a product appears in more than one split. □

EC.2.5. Proof of Proposition 7

Define $\Delta^{\mathbf{leaves}}$ to be the ($|\mathbf{leaves}| - 1$)-dimensional unit simplex:

$$\Delta^{\mathbf{leaves}} = \{\mathbf{y} \in \mathbb{R}^{|\mathbf{leaves}|} \mid \sum_{\ell \in \mathbf{leaves}} y_\ell = 1; y_\ell \geq 0, \forall\ \ell \in \mathbf{leaves}\}. \tag{EC.26}$$

In addition, for any $S \subseteq \mathbf{leaves}$, define $Q(S) = \{\mathbf{y} \in \Delta^{\mathbf{leaves}} \mid y_\ell \leq 0 \text{ for } \ell \in \mathbf{leaves} \setminus S\}$. We write the combinatorial disjunctive constraint over the ground set **leaves** as

$$\text{CDC}(\mathbf{leaves}) = \bigcup_{\ell \in \mathbf{leaves}} Q(\{\ell\}). \tag{EC.27}$$

Consider now the optimization problem

$$\begin{aligned} \underset{\mathbf{y}}{\text{maximize}} \quad & \sum_{\ell \in \mathbf{leaves}} r_\ell y_\ell & \text{(EC.28a)} \\ \text{subject to} \quad & \mathbf{y} \in \text{CDC}(\mathbf{leaves}). & \text{(EC.28b)} \end{aligned}$$

We will re-formulate this problem into a mixed-integer optimization problem. To do this, we claim that CDC(**leaves**) can be written as the following pairwise independent branching scheme:

$$\bigcup_{\ell \in \mathbf{leaves}} Q(\{\ell\}) = \bigcup_{i=1}^{n} \left(Q(L_i) \cup Q(R_i)\right), \tag{EC.29}$$

where

$$L_i = \{\ell \in \mathbf{leaves} \mid \ell \in \mathbf{left}(s) \text{ for some } s \text{ with } v(s) = i\}, \tag{EC.30}$$
$$R_i = \{\ell \in \mathbf{leaves} \mid \ell \in \mathbf{right}(s) \text{ for some } s \text{ with } v(s) = i\}. \tag{EC.31}$$

Note that (EC.29) is equivalent to the statement

$$\bigcup_{\ell \in \mathbf{leaves}} \{\ell\} = \bigcup_{i=1}^{n} \left((\mathbf{leaves} \setminus L_i) \cup (\mathbf{leaves} \setminus R_i)\right). \tag{EC.32}$$

To establish (EC.32), it is sufficient to prove the following equivalence:

$$\{\ell\} = \bigcap_{i \in I(\ell)} (\mathbf{leaves} \setminus R_i) \cap \bigcap_{i \in E(\ell)} (\mathbf{leaves} \setminus L_i) \cap \bigcap_{\substack{i \in \mathcal{N}: \\ i \notin I(\ell) \cup E(\ell)}} (\mathbf{leaves} \setminus L_i), \tag{EC.33}$$

where $I(\ell)$ and $E(\ell)$ are defined as

$$I(\ell) = \{i \in \mathcal{N} \mid \ell \in \bigcup_{s: v(s) = i} \mathbf{left}(s)\}, \tag{EC.34}$$
$$E(\ell) = \{i \in \mathcal{N} \mid \ell \in \bigcup_{s: v(s) = i} \mathbf{right}(s)\}. \tag{EC.35}$$

We now prove (EC.33).

Equation (EC.33)*, $\subseteq$ direction:* For $i \in I(\ell)$, we have that $\ell \in \bigcup_{s:v(s)=i} \mathbf{left}(s)$. This means that there exists $\bar{s}$ such that $\ell \in \mathbf{left}(\bar{s})$ and $v(\bar{s}) = i$. Since $\ell \in \mathbf{left}(\bar{s})$, this means that $\ell \notin \mathbf{right}(\bar{s})$ (a leaf cannot be to the left and to the right of any split). Moreover, ℓ cannot be in $\mathbf{right}(s)$ for any other s with $v(s) = i$, because this would mean that product i appears more than once along the path to leaf ℓ, violating Assumption 1. Therefore, $\ell \in \mathbf{leaves} \setminus R_i$ for any $i \in I(\ell)$.

For $i \in E(\ell)$, we have that $\ell \in \bigcup_{s:v(s)=i} \mathbf{right}(s)$. This means that there exists a split $\bar{s}$ such that $\ell \in \mathbf{right}(\bar{s})$ and $v(\bar{s}) = i$. Since $\ell \in \mathbf{right}(\bar{s})$, we have that $\ell \notin \mathbf{left}(\bar{s})$. In addition, ℓ cannot be in $\mathbf{left}(s)$ for any other s with $v(s) = i$. Therefore, $\ell \in \mathbf{leaves} \setminus L_i$ for any $i \in E(\ell)$.

Lastly, for any $i \notin I(\ell) \cup E(\ell)$, note that product i does not appear in any split along the path from the root of the tree to leaf ℓ. Therefore, for any s with $v(s) = i$, it will follow that either $\mathbf{left}(s) \subseteq \mathbf{left}(s')$ for some s' for which $\ell \in \mathbf{right}(s')$, or $\mathbf{left}(s) \subseteq \mathbf{right}(s')$ for some s' for which $\ell \in \mathbf{left}(s')$ – in other words, there is a split s' such that every leaf in $\mathbf{left}(s)$ is to one side of s' and ℓ is on the other side of s'. This means that ℓ' cannot be in $\mathbf{left}(s)$ for any s with $v(s) = i$, or equivalently, $\ell \in \mathbf{leaves} \setminus L_i$ for any $i \notin I(\ell) \cup E(\ell)$.

Equation (EC.33)*, $\supseteq$ direction:* To prove this, we will prove the contrapositive, which is

$$\{\ell' \in \mathbf{leaves} \mid \ell' \neq \ell\} \subseteq \bigcup_{i \in I(\ell)} R_i \cup \bigcup_{i \in E(\ell)} L_i \cup \bigcup_{\substack{i \in \mathcal{N}: \\ i \notin I(\ell) \cup E(\ell)}} L_i.$$

A straightforward result (see Lemma EC.1 from Mišić 2020) is that

$$\{\ell' \in \mathbf{leaves} \mid \ell' \neq \ell\} = \bigcup_{s:\ell \in \mathbf{left}(s)} \mathbf{right}(s) \cup \bigcup_{s:\ell \in \mathbf{right}(s)} \mathbf{left}(s).$$

Thus, if $\ell' \neq \ell$, then we have that $\ell' \in \mathbf{right}(s)$ for some s such that $\ell \in \mathbf{left}(s)$, or $\ell' \in \mathbf{left}(s)$ for some s such that $\ell \in \mathbf{right}(s)$. Let $i^* = v(s)$. In the first case, since $\ell \in \mathbf{left}(s)$, we have that $i^* \in I(\ell)$, and we thus have

$$\begin{aligned}
\mathbf{right}(s) &\subseteq \bigcup_{s':v(s')=i^*} \mathbf{right}(s') \\
&= R_{i^*} \\
&\subseteq \bigcup_{i \in I(\ell)} R_i \cup \bigcup_{i \in E(\ell)} L_i \cup \bigcup_{\substack{i' \in \mathcal{N}: \\ i' \notin I(\ell) \cup E(\ell)}} L_i.
\end{aligned}$$

In the second case, since $\ell \in \mathbf{right}(s)$, we have that $i^* \in E(\ell)$, and thus we have

$$\begin{aligned}
\mathbf{left}(s) &\subseteq \bigcup_{s':v(s')=i^*} \mathbf{left}(s') \\
&= L_{i^*} \\
&\subseteq \bigcup_{i \in I(\ell)} R_i \cup \bigcup_{i \in E(\ell)} L_i \cup \bigcup_{\substack{i' \in \mathcal{N}: \\ i' \notin I(\ell) \cup E(\ell)}} L_i.
\end{aligned}$$

This establishes the validity of the pairwise independent branching scheme (EC.32). Thus, a valid formulation for problem (EC.28) (see formulation (9) in Vielma et al. 2010, formulation (14) in Vielma and Nemhauser 2011 and formulation (13) in Huchette and Vielma 2019) is

$$\underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{\ell \in \mathbf{leaves}} r_\ell y_\ell \tag{EC.36a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}} y_\ell = 1, \tag{EC.36b}$$

$$\sum_{\ell \in L_i} y_\ell \leq x_i, \quad \forall\, i \in \mathcal{N}, \tag{EC.36c}$$

$$\sum_{\ell \in R_i} y_\ell \leq 1 - x_i, \quad \forall\, i \in \mathcal{N}, \tag{EC.36d}$$

$$y_\ell \geq 0, \quad \forall\, \ell \in \mathbf{leaves}, \tag{EC.36e}$$

$$x_i \in \{0,1\}, \quad \forall\, i \in \mathcal{N}. \tag{EC.36f}$$

Observe that, by the definition of L_i and R_i, formulation (EC.36) is identical to ProductMIO when $|F| = 1$. By invoking Theorem 1 from Vielma et al. (2010) with appropriate modifications, we can assert that formulation (EC.36) is integral. Therefore, in the special case that $|F| = 1$, formulation ProductMIO is always integral. $\square$

EC.2.6. Proof of Theorem 1

Proof of part (a) (feasibility): By definition, the solution produced by Algorithm 1 produces a solution $\mathbf{y}_t$ that never violates constraints (13c) and (13d). In addition, at each stage of Algorithm (1), the quantities $x_{v(t,s)}$ and $1 - x_{v(t,s)}$ are always nonnegative, and the quantity $1 - \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}$ is never allowed to become negative; thus, the solution $\mathbf{y}_t$ that is produced will satisfy the nonnegativity constraint (13e). The only constraint that needs to be verified is the unit sum constraint (13b).

Notice that by the definition of Algorithm 1, $\mathbf{y}_t$ will satisfy the unit sum constraint if and only if a C event occurs in Algorithm 1. To show that the unit sum constraint is satisfied, let us assume that it is not. This means that in the execution of Algorithm 1, a C event never occurs, and for every leaf, either a $A_{\ell,s}$ or a $B_{\ell,s}$ event occurs for some s.

For any split node j of a tree t, let $\mathbf{leftchild}(j)$ and $\mathbf{rightchild}(j)$ denote its left and right child nodes respectively, and let $\mathbf{root}(t)$ denote the root node of the tree. Consider the leaf ℓ^* that is obtained by the following procedure:

```
Procedure 1:
1. Set $j \leftarrow \mathbf{root}(t)$.
2. If $j \in \mathbf{leaves}(t)$, return $\ell^* = j$. Otherwise, go to Step 3.
3. If $x_{v(t,j)} \geq 0.5$, then set $j \leftarrow \mathbf{leftchild}(j)$, and return to Step 2.
   Otherwise, set $j \leftarrow \mathbf{rightchild}(j)$, and return to Step 2.
```

The leaf ℓ^* that is produced by Procedure 1 is useful for the following reason. Upon termination of Algorithm 1, the hypothesis that a C event never occurs means that we will have

$$y_{t,\ell^*} = \min\left\{ \min_{s:\ell^* \in \mathbf{left}(s)} x_{v(t,s)},\ \min_{s:\ell^* \in \mathbf{right}(s)} (1 - x_{v(t,s)}) \right\}. \tag{EC.37}$$

Note that in the above, the minimum will be equal to $x_{v(t,s)}$ for some s satisfying $\ell^* \in \mathbf{left}(s)$, or it will be equal to $1 - x_{v(t,s)}$ for some s satisfying $\ell^* \in \mathbf{right}(s)$. We now consider these two cases separately.

Case 1: $y_{t,\ell^*} = x_{v(t,s^*)}$ for some s^* for which $\ell^* \in \mathbf{left}(s^*)$. In this case, consider the following procedure for identifying another leaf ℓ':

```
Procedure 2.A:
1. Set $j \leftarrow \mathbf{root}(t)$.
2. If $j \in \mathbf{leaves}(t)$, terminate with $\ell' = j$. Otherwise, go to Step 3.
3. If $j = s^*$, set $j \leftarrow \mathbf{rightchild}(j)$, and return to Step 2. Otherwise, go to Step 4.
4. If $x_{v(t,j)} \geq 0.5$, then set $j \leftarrow \mathbf{leftchild}(j)$, and return to Step 2.
   Otherwise, set $j \leftarrow \mathbf{rightchild}(j)$, and return to Step 2.
```

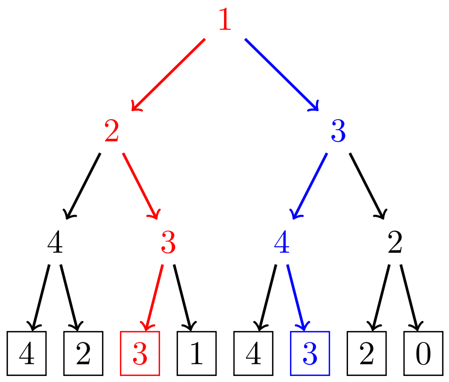


Figure EC.4 Example of Procedure 1 and Procedure 2.A for a tree, where $\mathbf{x} = (x_1, x_2, x_3, x_4) = (0.6, 0.3, 0.7, 0.1)$.

Procedure 2.A will return a leaf ℓ' for which the following will be true after the termination of Algorithm 1:

$$y_{t,\ell'} = \min \left\{ \underbrace{\min_{s \neq s^* : \ell' \in \mathbf{left}(s^*)} x_{v(t,s)}}_{(a)}, \underbrace{\min_{s : \ell' \in \mathbf{right}(s^*)} (1 - x_{v(t,s)})}_{(b)}, \underbrace{1 - x_{v(t,s^*)}}_{(c)} \right\} \tag{EC.38}$$

$$= 1 - x_{v(t,s^*)} \tag{EC.39}$$

where the second equality follows because, by the definition of Procedure 1, we know that $x_{v(t,s)} \geq 0.5$ for s such that $\ell^* \in \mathbf{left}(s)$, and $x_{v(t,s)} < 0.5$ or equivalently, $1 - x_{v(t,s)} \geq 0.5$ for s such that $\ell^* \in \mathbf{right}(s)$. Thus, in the above, terms (a) and (b) will both be at least 0.5, while term (c) is strictly less than 0.5. For this reason, $y_{t,\ell'}$ must be equal to $1 - x_{v(t,s^*)}$.

Observe that $y_{t,\ell^*} = x_{v(t,s^*)}$ and $y_{t,\ell'} = 1 - x_{v(t,s^*)}$, which means that $y_{t,\ell^*} + y_{t,\ell'} = 1$. Thus, if Algorithm 1 had encountered leaf ℓ^* followed by leaf ℓ', then by the definition of Algorithm 1, a C event should have been triggered at ℓ', contradicting our assumption that a C event never occurs. Similarly, if ℓ' was encountered before ℓ^*, then a C event should have occurred at ℓ^*, again resulting in a contradiction.

Figure EC.4 provides an example of Procedure 1 and Procedure 2.A applied to a tree. In this example, $\mathbf{x} = (x_1, x_2, x_3, x_4) = (0.6, 0.3, 0.7, 0.1)$. In this example, the sequence of red nodes is the path traversed by Procedure 1. This results in the leaf ℓ^*, which is the red leaf in the figure, whose value is $y_{t,\ell^*} = \min\{0.6, 1 - 0.3, 0.7\} = 0.6$. This value is exactly equal to $x_{v(t,s)}$ when s is equal to the root node, so we have that s^* is equal to the root node. We now apply Procedure 2.A, which traverses the sequence of nodes indicated in blue, and returns the blue leaf as ℓ', for which $y_{t,\ell'} = \min\{1 - 0.6, 0.7, 1 - 0.1\} = 0.4$.

Case 2: $y_{t,\ell^*} = 1 - x_{v(t,s^*)}$ for some s^* for which $\ell^* \in \mathbf{right}(s^*)$. In this case, consider the following procedure that is similar to Procedure 2.A above:

```
Procedure 2.B:
1. Set $j \leftarrow \mathbf{root}(t)$.
2. If $j \in \mathbf{leaves}(t)$, terminate with $\ell' = j$. Otherwise, go to Step 3.
3. If $j = s^*$, set $j \leftarrow \mathbf{leftchild}(j)$, and return to Step 2. Otherwise, go to Step 4.
4. If $x_{v(t,j)} \geq 0.5$, then set $j \leftarrow \mathbf{leftchild}(j)$, and return to Step 2.
   Otherwise, set $j \leftarrow \mathbf{rightchild}(j)$, and return to Step 2.
```

In the same way as for Case 1, we can show that

$$\begin{aligned} y_{t,\ell'} &= \min \left\{ \underbrace{\min_{s:\ell' \in \mathbf{left}(s^*)} x_{v(t,s)}}_{(a)}, \underbrace{\min_{s \neq s^*:\ell' \in \mathbf{right}(s^*)} (1 - x_{v(t,s)})}_{(b)}, \underbrace{x_{v(t,s^*)}}_{(c)} \right\} \\ &= x_{v(t,s^*)} \end{aligned}$$

Similarly to Case 1, we can again see that $y_{t,\ell^*} + y_{t,\ell'} = 1 - x_{v(t,s^*)} + x_{v(t,s^*)} = 1$, which implies that a C event must have occurred when either ℓ' or ℓ^* was checked. Thus, we again reach a contradiction.

These two cases establish that $\mathbf{y}_t$ must satisfy the unit sum constraint and therefore, that $\mathbf{y}_t$ is a feasible solution of problem (13).

Proof of part (b) (extreme point): To show that $\mathbf{y}_t$ is an extreme point, let us assume that $\mathbf{y}_t$ is not an extreme point. Then, there exist solutions $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$ different from $\mathbf{y}_t$ and a weight $\theta \in (0,1)$ such that $\mathbf{y}_t = \theta \mathbf{y}_t^1 + (1-\theta)\mathbf{y}_t^2$.

Let ℓ^* be the first leaf checked by Algorithm 1 at which $y_{t,\ell^*} \neq y_{t,\ell^*}^1$ and $y_{t,\ell^*} \neq y_{t,\ell^*}^2$. Such a leaf must exist because $\mathbf{y}_t \neq \mathbf{y}_t^1$ and $\mathbf{y}_t \neq \mathbf{y}_t^2$, and because $\mathbf{y}_t$ is the convex combination of $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$. Without loss of generality, let us further assume that

$$y_{t,\ell^*}^1 < y_{t,\ell^*} < y_{t,\ell^*}^2.$$

By definition, Algorithm 1 sets each $y_{t,\ell}$ to the largest it can be without violating the left split constraints (4c) and the right split constraints (4d), and ensuring that $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}$ does not exceed 1. Since $y_{t,\ell^*}^2 > y_{t,\ell^*}$, and since $\mathbf{y}_t^2$ and $\mathbf{y}_t$ are equal for all leaves checked before ℓ^*, this implies that $\mathbf{y}_t^2$ either violates constraint (3c), violates constraint (3d), or is such that $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} > 1$. This implies that $\mathbf{y}_t^2$ cannot be a feasible solution, which contradicts the assumption that $\mathbf{y}_t^2$ is a feasible solution, and ultimately contradicts $\mathbf{y}_t$ not being an extreme point. □

EC.2.7. Proof of Theorem 2

Proof of part (a) (feasibility): We first establish constraint (14c). First, observe that when for any split s and leaf ℓ such that $A_{s,\ell} \notin \mathcal{E}$, then $\alpha_{t,s,\ell} = 0$, which automatically satisfies the constraint. For any (s,ℓ) such that $A_{t,s,\ell} \in \mathcal{E}$, observe that it must be the case that the leaf $f(A_{s,\ell})$ that is checked when $A_{s,\ell}$ occurs was checked before the leaf $f(C)$ that is checked when the C event occurs; this is because Algorithm 1 terminates when the C event occurs. As a result, it must be that $r_{t,f(A_{s,\ell})} \geq r_{t,f(C)}$, since the leaves are checked in decreasing order of revenue. As a result, $\alpha_{t,s,\ell} = r_{t,f(A_{s,\ell})} - \gamma_t = r_{t,f(A_{s,\ell})} - r_{t,f(C)} \geq 0$, which establishes constraint (14c).

Constraint (14d) (that each $\beta_{t,s,\ell}$ is nonnegative) follows by similar reasoning as constraint (14c); for brevity, we omit the steps.

This leaves constraint (14b). Let ℓ be a leaf. There are three collectively exhaustive cases to consider: (1) ℓ is a leaf such that $A_{s,\ell} \in \mathcal{E}$ for some $s \in \mathbf{splits}(t)$; (2) ℓ is a leaf such that $B_{s,\ell} \in \mathcal{E}$ for some $s \in \mathbf{splits}(t)$; and (3) ℓ is a leaf such that $r_{t,\ell} \leq r_{t,f(C)}$. Note that these are collectively exhaustive because every leaf ℓ with $r_{t,\ell} > r_{t,f(C)}$ is a leaf that is checked before the final leaf $f(C)$, and thus by the definition of Algorithm 1, either an $A_{s,\ell}$ or a $B_{s,\ell}$ event occurs for some split s.

In the first case, if ℓ is a leaf such that $A_{s,\ell} \in \mathcal{E}$ for some $s \in \mathbf{splits}(t)$, then let $s_{A,\ell}$ be that split. We then have

$$\begin{aligned} &\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t \\ &\geq \alpha_{t,s_{A,\ell},\ell} + \gamma_t \\ &= r_{t,\ell} - \gamma_t + \gamma_t \\ &= r_{t,\ell} \end{aligned}$$

where the first step follows by the nonnegativity of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ and the fact that $s_{A,\ell} \in \mathbf{LS}(\ell)$; and the second step follows by how the dual procedure (Algorithm 2) sets $\alpha_{t,s,\ell}$ when $A_{s,\ell} \in \mathcal{E}$.

Similarly, in the second case, if ℓ is a leaf such that $B_{s,\ell} \in \mathcal{E}$ for some $s \in \mathbf{splits}(t)$, then letting $s_{B,\ell}$ be that split, we similarly have

$$\begin{aligned}&\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t \\ &\geq \beta_{t,s_{B,\ell},\ell} + \gamma_t \\ &= r_{t,\ell} - \gamma_t + \gamma_t \\ &= r_{t,\ell}.\end{aligned}$$

Finally, in the third case, if ℓ is such that $r_{t,\ell} \leq r_{t,f(C)}$, then the nonnegativity of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ immediately gives us that

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t \geq \gamma_t = r_{t,f(C)} \geq r_{t,\ell}.$$

This establishes that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to problem (14).

Proof of part (b) (basic feasible solution): To establish this, we will use the equivalence between extreme points and basic feasible solutions. A feasible solution $\mathbf{z}$ in a polyhedron $P = \{\mathbf{z} \in \mathbb{R}^m \mid \mathbf{Az} \leq \mathbf{b}\}$ is a basic feasible solution if there are m linearly independent active constraints at $\mathbf{z}$.

First, let us define the sets L_A and L_B as follows:

$$\begin{aligned}L_A &= \{\ell \in \mathbf{leaves}(t) \mid A_{s,\ell} \in \mathcal{E} \text{ for some } s \in \mathbf{splits}(t)\}, \\ L_B &= \{\ell \in \mathbf{leaves}(t) \mid B_{s,\ell} \in \mathcal{E} \text{ for some } s \in \mathbf{splits}(t)\}.\end{aligned}$$

Additionally, let us define $s_{A,\ell}$ to be the split for which an $A_{s,\ell}$ event occurs for leaf $\ell \in L_A$, and $s_{B,\ell}$ to be the split for which an $B_{s,\ell}$ event occurs for leaf $\ell \in L_B$. We note that for each leaf ℓ, by the definition of Algorithm 1, there is at most one event of the form $A_{s,\ell}$, $B_{s,\ell}$ or C that can occur. Thus, an immediate identity is

$$|L_A| + |L_B| + 1 = |\mathcal{E}|,$$

where on the left hand side, the first term counts the number of $A_{s,\ell}$ events, the second counts the number of $B_{s,\ell}$ events that have occurred, and the 1 corresponds to the single C event that must occur.

Now, consider the following system of equations:

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t = r_{t,\ell}, \quad \forall \ell \in L_A, \tag{EC.40}$$

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t = r_{t,\ell}, \quad \forall \ell \in L_B, \tag{EC.41}$$

$$\sum_{s \in \mathbf{LS}(f(C))} \alpha_{t,s,f(C)} + \sum_{s \in \mathbf{RS}(f(C))} \beta_{t,s,f(C)} + \gamma_t = r_{t,f(C)}, \tag{EC.42}$$

$$\alpha_{t,s,\ell} = 0, \quad \forall s \in \mathbf{splits}(t), \ell \in \mathbf{left}(s) \text{ such that } A_{s,\ell} \notin \mathcal{E}, \tag{EC.43}$$

$$\beta_{t,s,\ell} = 0, \quad \forall s \in \mathbf{splits}(t), \ell \in \mathbf{right}(t) \text{ such that } B_{s,\ell} \notin \mathcal{E}. \tag{EC.44}$$

Observe that each equation corresponds to one of the constraints of problem (14) being made to hold at equality. Observe that there are in total $\sum_{s \in \mathbf{splits}(t)} |\mathbf{left}(s)| + \sum_{s \in \mathbf{splits}(t)} |\mathbf{right}(s)| + 1$ variables. There are $|L_A| + |L_B| + 1 + \left[\sum_{s \in \mathbf{splits}(t)} |\mathbf{left}(s)| - |L_A|\right] + \left[\sum_{s \in \mathbf{splits}(t)} |\mathbf{right}(s)| - |L_B|\right] =$

$\sum_{s \in \mathbf{splits}(t)} |\mathbf{left}(s)| + \sum_{s \in \mathbf{splits}(t)} |\mathbf{right}(s)| + 1$ equations. We now show that the only solution to this system of equations is exactly the solution produced by Algorithm 2.

First, observe that by the property that at most one event of the form $A_{s,\ell}$, $B_{s,\ell}$ or C can occur for the leaf $\ell = f(C)$, equations (EC.42) - (EC.44) imply that

$$\begin{aligned}
&\sum_{s \in \mathbf{LS}(f(C))} \alpha_{t,s,f(C)} + \sum_{s \in \mathbf{RS}(f(C))} \beta_{t,s,f(C)} + \gamma_t \\
&= 0 + 0 + \gamma_t \\
&= \gamma_t \\
&= r_{t,f(C)},
\end{aligned}$$

which is exactly how Algorithm 2 sets γ_t.

Second, observe again that by the property that at most one event of the form $A_{s,\ell}$, $B_{s,\ell}$ or C can occur for a leaf $\ell \in L_A$, equations (EC.40), (EC.43) and (EC.44) imply that for any leaf $\ell \in L_A$,

$$\begin{aligned}
&\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t \\
&= \alpha_{t,s_{A,\ell},\ell} + \gamma_t \\
&= r_{t,\ell},
\end{aligned}$$

or equivalently, that $\alpha_{t,s_{A,\ell},\ell} = r_{t,\ell} - \gamma_t$, which is exactly how Algorithm 2 sets $\alpha_{t,s_{A,\ell},\ell}$.

Similarly, for any leaf $\ell \in L_B$, equations (EC.41), (EC.43) and (EC.44) imply

$$\begin{aligned}
&\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t \\
&= \beta_{t,s_{B,\ell},\ell} + \gamma_t \\
&= r_{t,\ell},
\end{aligned}$$

or equivalently, $\beta_{t,s_{B,\ell},\ell} = r_{t,\ell} - \gamma_t$, which again exactly agrees with Algorithm 2.

Finally, for any s, ℓ such that $A_{s,\ell} \notin \mathcal{E}$, equation (EC.43) exactly matches how Algorithm 2 sets $\alpha_{t,s,\ell}$ for such (s,ℓ) combinations. Similarly, equation (EC.44) exactly matches how Algorithm 2 sets $\beta_{t,s,\ell}$ for (s,ℓ) pairs for which $B_{s,\ell} \notin \mathcal{E}$.

Since the solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ that is completely determined by equations (EC.40) - (EC.44) is identical to the one produced by Algorithm (2), it follows that this solution is an extreme point. □

EC.2.8. Proof of Theorem 3

We will prove this by showing that the two solutions $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ satisfy complementary slackness. The complementary slackness conditions for this problem are

$$(x_{v(t,s)} - y_{t,\ell}) \cdot \alpha_{t,s,\ell} = 0, \quad \forall\, s \in \mathbf{splits}(t),\ \ell \in \mathbf{left}(s), \tag{EC.45}$$

$$(1 - x_{v(t,s)} - y_{t,\ell}) \cdot \beta_{t,s,\ell} = 0, \quad \forall\, s \in \mathbf{splits}(t),\ \ell \in \mathbf{right}(s), \tag{EC.46}$$

$$y_{t,\ell} \cdot \Big(\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s,\ell} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s,\ell} + \gamma_t - r_{t,\ell} \Big) = 0, \quad \forall\, \ell \in \mathbf{leaves}(t). \tag{EC.47}$$

We now verify each of these conditions.

Condition (EC.45)**:** For this condition, observe that if $A_{s,\ell} \notin \mathcal{E}$, then by the definition of Algorithm 2, $\alpha_{t,s,\ell}$ will be equal to the default value of zero, and the condition will automatically hold. If $A_{s,\ell} \in \mathcal{E}$, then by the definition of the primal procedure (Algorithm 1) $y_{t,\ell}$ will be set to q^* which is equal to $x_{v(t,s)}$. Thus, we will have that $x_{v(t,s)} - y_{t,\ell} = 0$ and the condition is again satisfied.

Condition (EC.46)**:** This condition follows by analogous reasoning as condition (EC.45).

Condition (EC.47)**:** For this condition, we can see that if $y_{t,\ell} = 0$, then the condition is immediately satisfied; thus, let us assume that $y_{t,\ell} > 0$. In this case, it must be that when ℓ was checked by Algorithm 1, that either an $A_{s,\ell}$ event occurred for some split s, a $B_{s,\ell}$ event occurred for some split s or a C event occurred. As shown in the proof of Theorem EC.2.7 (see equations (EC.40) - (EC.42)), for any leaf for which such an event occurs, the constraint (14b) is satisfied at equality and thus condition (EC.47) must hold as well.

Since all three conditions hold, it follows that the solutions $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ produced by Algorithms 1 and 2 are optimal for their respective problems. □

EC.2.9. Proof of Theorem EC.1 (LeafMIO closed form for integer x)

Proof of part (a) (primal feasibility): Observe that by construction, $\mathbf{y}_t$ automatically satisfies the nonnegativity constraint (13e) and the unit sum constraint (13b). For constraints (13c) and (13d), observe that for any $\ell \neq \ell^*$, these constraints are automatically satisfied because $y_{t,\ell} = 0$ whereas $x_{v(t,s)}$ and $1 - x_{v(t,s)}$ can only be either 0 or 1. For the case that $\ell = \ell^*$, observe that if $\ell^* \in \mathbf{left}(s)$ for some split s, then it must be that $x_{v(t,s)} = 1$, so the constraint $y_{t,\ell^*} \leq x_{v(t,s)}$ is satisfied. Similarly, if $\ell^* \in \mathbf{right}(s)$ for some split s, then it must be that $x_{v(t,s)} = 0$, so the constraint $y_{t,\ell^*} \leq 1 - x_{v(t,s)}$ is satisfied. Thus, $\mathbf{y}_t$ is a feasible solution of problem (13).

Proof of part (b) (dual feasibility): First, observe that by the definition of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$, they are automatically nonnegative, and so constraints (14c) and (14d) are satisfied. To verify constraint (14b), observe that for any $\ell \neq \ell^*$, it is either the case that $\ell \in \mathbf{right}(s')$ for some $s' \in \mathbf{LS}(\ell^*)$, or $\ell \in \mathbf{left}(s')$ for some $s' \in \mathbf{RS}(\ell^*)$. In the first case, we have:

$$\begin{aligned} &\sum_{s:\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} + \sum_{s:\ell \in \mathbf{right}(s)} \beta_{t,s,\ell} + \gamma_t \\ &\geq \beta_{t,s,\ell} + \gamma_t \\ &\geq r_{t,\ell} - r_{t,\ell^*} + r_{t,\ell^*} \\ &= r_{t,\ell} \end{aligned}$$

where the first inequality follows by the nonnegativity of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ and the fact that $\ell \in \mathbf{right}(s')$, and the second inequality follows by the definition of $\boldsymbol{\beta}_t$ and the fact that $s' \in \mathbf{LS}(\ell^*)$. In the second case, where $\ell \in \mathbf{left}(s')$ for some $s' \in \mathbf{RS}(\ell^*)$, similar logic lets us establish that

$$\begin{aligned} &\sum_{s:\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} + \sum_{s:\ell \in \mathbf{right}(s)} \beta_{t,s,\ell} + \gamma_t \\ &\geq \alpha_{t,s',\ell} + \gamma_t \\ &\geq r_{t,\ell} - r_{t,\ell^*} + r_{t,\ell^*} \\ &= r_{t,\ell}. \end{aligned}$$

This establishes that constraint (14b) holds for any leaf ℓ other than ℓ^*. When $\ell = \ell^*$, we very simply have

$$\begin{aligned} &\sum_{s:\ell^* \in \mathbf{left}(s)} \alpha_{t,s,\ell^*} + \sum_{s:\ell^* \in \mathbf{right}(s)} \beta_{t,s,\ell} + \gamma_t \\ &= 0 + 0 + \gamma_t \\ &= r_{t,\ell^*}, \end{aligned}$$

where the first step follows from the definition of $\alpha_{t,s,\ell}$ and $\beta_{t,s,\ell}$ (note that from the statement of Theorem EC.1, α_{t,s,ℓ^*} will be zero for any $s \in \mathbf{LS}(\ell^*)$, or equivalently, any s such that $\ell^* \in \mathbf{left}(s)$, and similarly, β_{t,s,ℓ^*} will be zero for any $s \in \mathbf{RS}(\ell^*)$). Thus, this establishes that constraint (14b) holds for every leaf ℓ, and thus, that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution of the dual subproblem (14).

Proof of part (c) (optimality): To establish optimality, we simply need to check that the primal and dual solutions obtain the same objective; by weak duality, we will thus establish that the two solutions are optimal for their respective problems. For the primal solution $\mathbf{y}_t$, it is immediately clear that its objective is r_{t,ℓ^*}. For the dual solution, we have

$$\begin{aligned}
&\sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} x_{v(t,s)} + \sum_{s \in \mathbf{splits}(t)} \sum_{\ell \in \mathbf{right}(s)} \beta_{t,s,\ell}(1 - x_{v(t,s)}) + \gamma_t \\
&= \sum_{s \in \mathbf{RS}(\ell^*)} \sum_{\ell \in \mathbf{left}(s)} \alpha_{t,s,\ell} x_{v(t,s)} + \sum_{s \in \mathbf{RS}(\ell^*)} \sum_{\ell \in \mathbf{right}(s)} \beta_{t,s,\ell}(1 - x_{v(t,s)}) + \gamma_t \\
&= 0 + 0 + \gamma_t \\
&= r_{t,\ell^*},
\end{aligned}$$

where the first step follows because $\alpha_{t,s,\ell}$ is automatically zero for any $s \notin \mathbf{RS}(\ell^*)$ and $\beta_{t,s,\ell}$ is automatically zero for any $s \notin \mathbf{LS}(\ell^*)$; the second step follows because for any $s \in \mathbf{RS}(\ell^*)$, $x_{v(t,s)}$ will be 0 (recall that $x_{v(t,s)} = 0$ means that the product is not in the assortment, which implies that we must proceed to the right of any split $s \in \mathbf{RS}(\ell^*)$), and similarly, for any $s \in \mathbf{LS}(\ell^*)$, $x_{v(t,s)}$ will be 1; and the final step follows by the definition of γ_t. This establishes that $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for problems (13) and (14) respectively. □

EC.2.10. Proof of Theorem 4

Proof of part (b) (feasibility): By definition, the solution produced by Algorithm 3 produces a solution $\mathbf{y}_t$ that satisfies the left and right split constraints (17c) and (17d). With regard to the nonnegativity constraint (17e), we can see that at each stage of Algorithm 3, the quantities $x_{v(t,s)} - \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell}$, $1 - x_{v(t,s)} - \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell}$ and $1 - \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}$ never become negative; thus, the solution $\mathbf{y}_t$ produced upon termination satisfies the nonnegativity constraint (17e).

The only constraint that remains to be verified is constraint (17b), which requires that $\mathbf{y}_t$ adds up to 1. Observe that it is sufficient for a C event to occur during the execution of Algorithm 3 to ensure that constraint (17b) is satisfied. We will show that a C event must occur during the execution of Algorithm 3.

We proceed by contradiction. For the sake of a contradiction, let us suppose that a C event does not occur during the execution of the algorithm. Note that under this assumption, for any split s, it is impossible that the solution $\mathbf{y}_t$ produced by Algorithm 3 satisfies

$$\begin{aligned}
x_{v(t,s)} &= \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell}, \\
1 - x_{v(t,s)} &= \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell},
\end{aligned}$$

as this would imply that

$$\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} + \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} = x_{v(t,s)} + 1 - x_{v(t,s)} = 1;$$

by the definition of Algorithm 3, this would have triggered a C event at one of the leaves in $\mathbf{left}(s) \cup \mathbf{right}(s)$.

Thus, this means that at every split, either $\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} < x_{v(t,s)}$ or $\sum_{\ell \in \mathbf{right}(s)} < 1 - x_{v(t,s)}$. Using this property, let us identify a leaf ℓ^* using the following procedure:

```
Procedure 3:
1. Set $j \leftarrow \mathbf{root}(t)$.
2. If $j \in \mathbf{leaves}(t)$, terminate with $\ell^* = j$.
   Otherwise, proceed to Step 3.
3. If $\sum_{\ell \in \mathbf{left}(j)} y_{t,\ell} < x_{v(t,j)}$, set $j \leftarrow \mathbf{leftchild}(j)$.
   Otherwise, set $j \leftarrow \mathbf{rightchild}(j)$.
4. Repeat Step 2.
```

Note that by our observation that at most one of the left or right split constraints can be satisfied at equality for any split s, Procedure 3 above is guaranteed to terminate with a leaf ℓ^* such that:

$$y_{t,\ell^*} \leq \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} < x_{v(t,s)}, \quad \forall\ s \in \mathbf{splits}(t) \text{ such that } \ell^* \in \mathbf{left}(s),$$

$$y_{t,\ell^*} \leq \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} < 1 - x_{v(t,s)}, \quad \forall\ s \in \mathbf{splits}(t) \text{ such that } \ell^* \in \mathbf{right}(s).$$

However, this is impossible, because Algorithm 3 always sets each leaf $y_{t,\ell}$ to the highest value it can be without violating any of the left or right split constraints; the above conditions imply that y_{t,ℓ^*} could have been set higher, which is not possible. We thus have a contradiction, and it must be the case that a C event occurs.

Proof of part (b) (extreme point): To show that $\mathbf{y}_t$ is an extreme point, let us assume that $\mathbf{y}_t$ is not an extreme point. Then, there exist feasible solutions $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$ different from $\mathbf{y}_t$ and a weight $\theta \in (0,1)$ such that $\mathbf{y}_t = \theta \mathbf{y}_t^1 + (1-\theta)\mathbf{y}_t^2$.

Let ℓ^* be the first leaf checked by Algorithm 3 at which $y_{t,\ell^*} \neq y_{t,\ell^*}^1$ and $y_{t,\ell^*} \neq y_{t,\ell^*}^2$. Such a leaf must exist because $\mathbf{y}_t \neq \mathbf{y}_t^1$ and $\mathbf{y}_t \neq \mathbf{y}_t^2$, and because $\mathbf{y}_t$ is the convex combination of $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$. Without loss of generality, let us further assume that

$$y_{t,\ell^*}^1 < y_{t,\ell^*} < y_{t,\ell^*}^2.$$

By definition, Algorithm 3 sets $y_{t,\ell}$ at each iteration to the largest it can be without violating the left split constraints (17c) and the right split constraints (17d), and ensuring that $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}$ does not exceed 1. Since $y_{t,\ell^*}^2 > y_{t,\ell^*}$, and since $\mathbf{y}_t^2$ and $\mathbf{y}_t$ are equal for all leaves checked before ℓ^*, this implies that $\mathbf{y}_t^2$ either violates constraint (17c), violates constraint (17d), or is such that $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} > 1$. This implies that $\mathbf{y}_t^2$ cannot be a feasible solution, which contradicts the assumption that $\mathbf{y}_t^2$ is a feasible solution. □

EC.2.11. Proof of Theorem 5 (SplitMIO dual is BFS)

Proof of part (a) (feasibility): Before we prove the result, we first establish a helpful property of the events that are triggered during the execution of Algorithm 3.

Lemma EC.1. *Let $s_1, s_2 \in \mathbf{splits}(t)$, $s_1 \neq s_2$, such that s_2 is a descendant of s_1. Suppose that $e_1 = A_{s_1}$ or $e_1 = B_{s_1}$, and that $e_2 = A_{s_2}$ or $e_2 = B_{s_2}$. If e_1 and e_2 occur during the execution of Algorithm 3, then $r_{t,f(e_1)} \leq r_{t,f(e_2)}$.*

Proof: We will prove this by contradiction. Suppose that we have two splits s_1 and s_2 and events e_1 and e_2 as in the statement of the lemma, and that $r_{t,f(e_2)} < r_{t,f(e_1)}$. This implies that leaf $f(e_1)$ is checked before leaf $f(e_2)$. When leaf $f(e_1)$ is checked, the event e_1 occurs, which implies that either the left split constraint (17c) becomes tight (if $e_1 = A_{s_1}$) or the right split constraint (17d) becomes tight (if $e_1 = B_{s_1}$) at s_1. In either case, since s_2 is a descendant of s_1, the leaf $f(e_2)$ must be contained in the left leaves of split s_1 (if $e = A_{s_1}$) or the right leaves of split s_1 (if $e_1 = B_{s_1}$). Thus, when leaf $f(e_2)$ is checked, the event e_2 cannot occur, because q_{s_1} in Algorithm 3 will be zero (implying that $q_{A,B} = 0$), and so s^* cannot be equal to s_2 because s_1 is a shallower split that attains the minimum of $q_{A,B} = 0$. □

To establish that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is feasible for the SPLITMIO dual subproblem (18), we will first show that the $\alpha_{t,s}$ variables are nonnegative.

Fix $s \in \mathbf{splits}(t)$. If $A_s \notin \mathcal{E}$, then $\alpha_{t,s} = 0$, and constraint (18c) is satisfied. If $A_s \in \mathcal{E}$, then consider the split $\tilde{s}$ defined as follows:

$$\tilde{s} = \arg\min_{s'} \left[\{ d(s') \mid s' \in \mathbf{LS}(f(A_s)),\ d(s') < d,\ A_{s'} \in \mathcal{E} \} \cup \{ d(s') \mid s' \in \mathbf{RS}(f(A_s)),\ d(s') < d,\ B_{s'} \in \mathcal{E} \} \right],$$

where we recall that $d = d(s)$ is the depth of split s. In words, $\tilde{s}$ is the shallowest split (i.e., closest to the root) along the path of splits from the root node to split s such that either an $A_{\tilde{s}}$ event occurs or a $B_{\tilde{s}}$ event occurs for split $\tilde{s}$. There are three possible cases that can occur here, which we now handle.

Case 1: $\tilde{s} \in \mathbf{LS}(f(A_s))$. In this case, $A_{\tilde{s}} \in \mathcal{E}$, and we have

$$\begin{aligned}
\alpha_{t,s} &= r_{t,f(A_s)} - \left[\sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ d(s') < d, \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_s)): \\ d(s') < d, \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - \left[\alpha_{t,\tilde{s}} + \sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ d(s') < d(\tilde{s}), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_s)): \\ d(s') < d(\tilde{s}), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - \left[\alpha_{t,\tilde{s}} + \sum_{\substack{s' \in \mathbf{LS}(f(A_{\tilde{s}})): \\ d(s') < d(\tilde{s}), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_{\tilde{s}})): \\ d(s') < d(\tilde{s}), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - r_{t,f(A_{\tilde{s}})} \\
&\geq 0,
\end{aligned}$$

where the first step follows by the definition of $\alpha_{t,s}$ in Algorithm 4; the second step follows by the definition of $\alpha_{t,\tilde{s}}$ as the deepest split for which an A or B event occurs that is at a depth lower than s; the third step by the fact that the left splits and right splits of $f(A_{\tilde{s}})$ at a depth below $d(\tilde{s})$ are the same as the left and right splits of $f(A_s)$ at a depth below $d(\tilde{s})$; and the fourth step follows from the definition of $\alpha_{t,\tilde{s}}$ in Algorithm 4. The inequality follows by Lemma EC.1.

Case 2: $\tilde{s} \in \mathbf{RS}(f(A_s))$. In this case, $B_{\tilde{s}} \in \mathcal{E}$, and analogously to Case 1, we have:

$$\alpha_{t,s} = r_{t,f(A_s)} - \left[\sum_{\substack{s' \in \mathbf{LS}(f(A_s)): \\ d(s') < d, \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_s)): \\ d(s') < d, B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right]$$

$$
\begin{aligned}
&= r_{t,f(A_s)} - \left[\beta_{t,\tilde{s}} + \sum_{\substack{s' \in \mathbf{LS}(f(A_s)):\\ d(s') < d(\tilde{s}),\\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(A_s)):\\ d(s') < d(\tilde{s}),\\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - \left[\beta_{t,\tilde{s}} + \sum_{\substack{s' \in \mathbf{LS}(f(B_{\tilde{s}})):\\ d(s') < d(\tilde{s}),\\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(f(B_{\tilde{s}})):\\ d(s') < d(\tilde{s}),\\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \right] \\
&= r_{t,f(A_s)} - r_{t,f(B_{\tilde{s}})} \\
&\geq 0.
\end{aligned}
$$

Case 3: $\tilde{s}$ is undefined because the underlying sets are empty. In this case, $\alpha_{t,s} = r_{t,f(A_s)} - \gamma_t$. In this case, we have

$$\alpha_{t,s} = r_{t,f(A_s)} - \gamma_t = r_{t,f(A_s)} - r_{t,f(C)} \geq 0,$$

where the inequality follows because $f(C)$ is the last leaf to be checked before Algorithm 3 terminates, and thus it must be that $r_{t,f(A_s)} \geq r_{t,f(C)}$.

This establishes that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ satisfy constraint (18c). Constraint (18d) can be shown in an almost identical fashion; for brevity, we omit the steps.

We thus only need to verify constraint (18b). Let $\ell \in \mathbf{leaves}(t)$. Here, there are four mutually exclusive and collectively exhaustive cases to consider.

Case 1: $r_{t,\ell} \leq r_{t,f(C)}$. In this case we have

$$
\begin{aligned}
&\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t \\
&\geq \gamma_t \\
&= r_{t,f(C)} \\
&\geq r_{t,\ell}.
\end{aligned}
$$

Case 2: $r_{t,\ell} > r_{t,f(C)}$ and $\ell = f(A_s)$ for some $s \in \mathbf{splits}(t)$. In this case, we have

$$
\begin{aligned}
&\sum_{s' \in \mathbf{LS}(\ell)} \alpha_{t,s'} + \sum_{s' \in \mathbf{RS}(\ell)} \beta_{t,s'} + \gamma_t \\
&\geq \alpha_{t,s} + \sum_{\substack{s' \in \mathbf{LS}(\ell):\\ d(s') < d(s),\\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(\ell):\\ d(s') < d(s),\\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \\
&= r_{t,f(A_s)} \\
&= r_{t,\ell},
\end{aligned}
$$

where the first step follows by the nonnegativity of $\alpha_{t,s'}$ and $\beta_{t,s'}$ for all s', and the second step by the definition of $\alpha_{t,s}$ in Algorithm 4.

Case 3: $r_{t,\ell} > r_{t,f(C)}$ and $\ell = f(B_s)$ for some $s \in \mathbf{splits}(t)$. By similar logic as case 2, we have

$$\begin{aligned}
&\sum_{s' \in \mathbf{LS}(\ell)} \alpha_{t,s'} + \sum_{s' \in \mathbf{RS}(\ell)} \beta_{t,s'} + \gamma_t \\
&\geq \beta_{t,s} + \sum_{\substack{s' \in \mathbf{LS}(\ell): \\ d(s') < d(s), \\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(\ell): \\ d(s') < d(s), \\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t \\
&= r_{t,f(B_s)} \\
&= r_{t,\ell}.
\end{aligned}$$

Case 4: $r_{t,\ell} > r_{t,f(C)}$ and ℓ is not equal to $f(A_s)$ or $f(B_s)$ for any split s. In this case, when leaf ℓ is checked by Algorithm 3, the algorithm reaches line 17 where s^* is determined and e is set to either A_{s^*} or B_{s^*}, and it turns out that e is already in $\mathcal{E}$. If $e = A_{s^*}$, then this means that leaf $f(A_{s^*})$ was checked before leaf ℓ, and that $r_{t,\ell} \leq r_{t,f(A_{s^*})}$. We thus have

$$\begin{aligned}
&\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t \\
&\geq \alpha_{t,s^*} + \sum_{\substack{s \in \mathbf{LS}(\ell): \\ d(s) < d(s^*), \\ A_s \in \mathcal{E}}} \alpha_{t,s} + \sum_{\substack{s \in \mathbf{RS}(\ell): \\ d(s) < d(s^*), \\ B_s \in \mathcal{E}}} \beta_{t,s} + \gamma_t \\
&= \alpha_{t,s^*} + \sum_{\substack{s \in \mathbf{LS}(f(A_{s^*})): \\ d(s) < d(s^*), \\ A_s \in \mathcal{E}}} \alpha_{t,s} + \sum_{\substack{s \in \mathbf{RS}(f(A_{s^*})): \\ d(s) < d(s^*), \\ B_s \in \mathcal{E}}} \beta_{t,s} + \gamma_t \\
&= r_{t,f(A_{s^*})} \\
&\geq r_{t,\ell},
\end{aligned}$$

where the first equality follows because ℓ and $f(A_{s^*})$, by virtue of being to the left of s^*, share the same left and right splits at depths lower than $d(s^*)$. Similarly, if $e = B_{s^*}$, then the leaf $f(B_{s^*})$ was checked before ℓ, which means that $r_{t,\ell} \leq r_{t,f(B_{s^*})}$; in this case, we have

$$\begin{aligned}
&\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t \\
&\geq \beta_{t,s^*} + \sum_{\substack{s \in \mathbf{LS}(\ell): \\ d(s) < d(s^*), \\ A_s \in \mathcal{E}}} \alpha_{t,s} + \sum_{\substack{s \in \mathbf{RS}(\ell): \\ d(s) < d(s^*), \\ B_s \in \mathcal{E}}} \beta_{t,s} + \gamma_t \\
&\geq \beta_{t,s^*} + \sum_{\substack{s \in \mathbf{LS}(f(B_{s^*})): \\ d(s) < d(s^*), \\ A_s \in \mathcal{E}}} \alpha_{t,s} + \sum_{\substack{s \in \mathbf{RS}(f(B_{s^*})): \\ d(s) < d(s^*), \\ B_s \in \mathcal{E}}} \beta_{t,s} + \gamma_t \\
&= r_{t,f(B_{s^*})} \\
&\geq r_{t,\ell}.
\end{aligned}$$

We have thus shown that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution to the SplitMIO dual subproblem (18).

Proof of part (b) (extreme point): To prove this, we will use the equivalence between extreme points and basic feasible solutions, and show that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a basic feasible solution of problem (18).

Define the sets L_A and L_B as

$$L_A = \{\ell \in \mathbf{leaves}(t) \mid \ell = f(A_s) \text{ for some } s \in \mathbf{splits}(t)\},$$
$$L_B = \{\ell \in \mathbf{leaves}(t) \mid \ell = f(B_s) \text{ for some } s \in \mathbf{splits}(t)\}.$$

Consider the following system of equations:

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t = r_{t,\ell}, \quad \forall \ell \in L_A, \tag{EC.48}$$

$$\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t = r_{t,\ell}, \quad \forall \ell \in L_B, \tag{EC.49}$$

$$\sum_{s \in \mathbf{LS}(f(C))} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(f(C))} \beta_{t,s} + \gamma_t = r_{t,f(C)}, \tag{EC.50}$$

$$\alpha_{t,s} = 0, \quad \forall \ s \text{ such that } A_s \notin \mathcal{E}, \tag{EC.51}$$

$$\beta_{t,s} = 0, \quad \forall \ s \text{ such that } B_s \notin \mathcal{E}. \tag{EC.52}$$

Observe that each equation corresponds to a constraint from problem (18) made to hold at equality. In addition, we note that there are $|L_A| + |L_B| + 1 + (|\mathbf{splits}(t)| - |L_A|) + (|\mathbf{splits}(t)| - |L_B|) = 2|\mathbf{splits}(t)| + 1$ equations, which is exactly the number of variables. We will show that the unique solution implied by this system of equations is exactly the solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ that is produced by Algorithm 4.

In order to establish this, we first establish a couple of useful results.

Lemma EC.2. *Suppose that $e \in \mathcal{E}$, $\ell = f(e)$ and $e = A_s$ or $e = B_s$ for some $s \in \mathbf{splits}(t)$. Then:*
a) $A_{s'} \notin \mathcal{E}$ for all $s' \in \mathbf{LS}(\ell)$ such that $d(s') > d(s)$; and
b) $B_{s'} \notin \mathcal{E}$ for all $s' \in \mathbf{RS}(\ell)$ such that $d(s') > d(s)$.

Proof of Lemma EC.2: We will prove this by contradiction. Without loss of generality, let us suppose that there exists an $A_{s'}$ event in $\mathcal{E}$ where $s' \in \mathbf{LS}(\ell)$ and $d(s') > d(s)$. (The case where there exists an $B_{s'}$ event in $\mathcal{E}$ where $s' \in \mathbf{RS}(\ell)$ and $d(s') > d(s)$ can be shown almost identically.)

Since $A_{s'} \in \mathcal{E}$, consider the leaf $\ell' = f(A_{s'})$. There are now two possibilities for when Algorithm 3 checks leaf ℓ':

1. **Case 1**: Leaf ℓ' is checked after leaf ℓ. In this case, in the iteration of Algorithm 3 corresponding to leaf ℓ', it will be the case that $q_s = 0$ because the left constraint (17c) at split s (if $e = A_s$) or the right constraint (17d) at split s (if $e = B_s$) became tight when leaf ℓ was checked. As a result, $q_{A,B} = 0$ in the iteration for leaf ℓ'. This implies that s' cannot be the lowest depth split that attains the minimum q_s value of $q_{A,B}$, because $q_s = 0$, and s has a depth lower than s', which contradicts the fact that the $A_{s'}$ event occurred.
2. **Case 2**: Leaf ℓ' is checked before leaf ℓ. In this case, consider the value of q_s when leaf ℓ is checked in Algorithm 3.

If $q_s > 0$, then there is immediately a contradiction, because $q_{s'} = 0$ when leaf ℓ is checked (this is true because the left split constraint (17c) at s' became tight after leaf ℓ' was checked), and thus it is impossible that $s^* = s$.

If $q_s = 0$, then this implies that $x_{v(t,s)} = 0$. This would imply that $q_s = 0$ when leaf ℓ' was checked, which would imply that s^* cannot be s' when leaf ℓ' is checked because s is at a lower depth than s'.

Thus, in either case, we arrive at a contradiction, which completes the proof. □

Lemma EC.3. *Suppose that $\ell = f(C)$. Then:*
a) $A_{s'} \notin \mathcal{E}$ for all $s' \in \mathbf{LS}(\ell)$; and
b) $B_{s'} \notin \mathcal{E}$ for all $s' \in \mathbf{RS}(\ell)$.

Proof of Lemma EC.3: We proceed by contradiction. Suppose that A_s occurs for some $s \in \mathbf{LS}(\ell)$ or that B_s occurs for some $s \in \mathbf{LS}(\ell)$; in the former case, let $e = A_s$, and in the latter case, let $e = B_s$. Let $\ell' = f(e)$. Then ℓ' must be checked before ℓ by Algorithm 3, since the algorithm always terminates after a C event occurs. Consider what happens when Algorithm 3 checks leaf ℓ:

1. **Case 1**: $q_C > 0$. This is impossible, because if e occurs, then q_s when leaf ℓ is checked would have to be 0, which would imply that $q_{A,B} < q_C$ and that a C event could not have occurred when ℓ was checked.
2. **Case 2**: $q_C = 0$. This is also impossible, because it implies that the unit sum constraint (17b) was satisfied at an earlier iteration, which would have triggered the C event at a leaf that was checked before ℓ.

We thus have that $A_{s'}$ does not occur for any $s' \in \mathbf{LS}(\ell)$ and $B_{s'}$ does not occur for any $s' \in \mathbf{RS}(\ell)$, as required. □

With these two lemmas in hand, we now return to the proof of Theorem EC.2.11 (b). Observe now that by using Lemmas EC.2 and EC.3 and using equations (EC.56) and (EC.57), the system of equations (EC.48)-(EC.52) is equivalent to

$$\alpha_{t,s} + \sum_{\substack{s' \in \mathbf{LS}(f(A_s)):\\ d(s') < d(s),\\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(\ell):\\ d(s') < d(s),\\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t = r_{t,f(A_s)}, \quad \forall s \text{ such that } A_s \in \mathcal{E}, \tag{EC.53}$$

$$\beta_{t,s} + \sum_{\substack{s' \in \mathbf{LS}(\ell):\\ d(s') < d(s),\\ A_{s'} \in \mathcal{E}}} \alpha_{t,s'} + \sum_{\substack{s' \in \mathbf{RS}(\ell):\\ d(s') < d(s),\\ B_{s'} \in \mathcal{E}}} \beta_{t,s'} + \gamma_t = r_{t,f(B_s)}, \quad \forall s \text{ such that } B_s \in \mathcal{E}, \tag{EC.54}$$

$$\gamma_t = r_{t,f(C)}, \tag{EC.55}$$

$$\alpha_{t,s} = 0, \quad \forall\ s \text{ such that } A_s \notin \mathcal{E}, \tag{EC.56}$$

$$\beta_{t,s} = 0, \quad \forall\ s \text{ such that } B_s \notin \mathcal{E}.. \tag{EC.57}$$

We now observe that the solution implied by this system of equations is exactly the solution produced by Algorithm 4. We thus establish that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a basic feasible solution of problem (18), and thus an extreme point. □

EC.2.12. Proof of Theorem 6 (SplitMIO primal and dual are optimal)

To prove that the $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ produced by Algorithms 3 and 4 are optimal for their respective problems, we show that they satisfy complementary slackness. The complementary slackness conditions for problems (17) and (18) are

$$\alpha_{t,s} \cdot \left(x_{v(t,s)} - \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \right) = 0, \quad \forall\ s \in \mathbf{splits}(t), \tag{EC.58}$$

$$\beta_{t,s} \cdot \left(1 - x_{v(t,s)} - \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \right) = 0, \quad \forall\ s \in \mathbf{splits}(t), \tag{EC.59}$$

$$y_{t,\ell} \cdot \left(\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t - r_{t,\ell} \right) = 0, \quad \forall\ \ell \in \mathbf{leaves}(t). \tag{EC.60}$$

Condition (EC.58): If $\alpha_{t,s} = 0$, then the condition is trivially satisfied. If $\alpha_{t,s} > 0$, then this implies that $A_s \in \mathcal{E}$. This means that the left split constraint (17c) at s became tight after leaves $f(A_s)$ was checked, which implies that $\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} = x_{v(t,s)}$ or equivalently, that $x_{v(t,s)} - \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} = 0$, which again implies that the condition is satisfied.

Condition (EC.59): This follows along similar logic to condition (EC.58), only that we use the fact that $\beta_{t,s} > 0$ implies that a B_s event occurred and that the right split constraint (17d) at s became tight.

Condition (EC.60): If $y_{t,\ell} = 0$, then the condition is trivially satisfied. If $y_{t,\ell} > 0$, then either $\ell = f(C)$, or $\ell = f(A_s)$ for some split $s \in \mathbf{LS}(\ell)$, or $\ell = f(B_s)$ for some split $s \in \mathbf{RS}(\ell)$. In any of these three cases, as shown in the proof of part (b) of Theorem 5, the dual constraint (18b) holds with equality for any such leaf ℓ. Thus, we have that $\sum_{s\in\mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s\in\mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t - r_{t,\ell} = 0$, and the condition is again satisfied.

Since complementary slackness holds, $\mathbf{y}_t$ is feasible for the primal problem (17) (by Theorem EC.2.10), and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is feasible for the dual problem (18) (by Theorem 5, it follows that $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for their respective problems. □

EC.2.13. Proof of Theorem EC.2 (SplitMIO primal and dual are closed form solvable for binary x)

Proof of part (a): Observe that by construction, $\mathbf{y}_t$ automatically satisfies the unit sum constraint (17b) and the nonnegativity constraint (17e). We thus need to verify constraints (17c) and (17d).

For constraint (17c), observe that for any split $s \notin \mathbf{LS}(\ell^*)$, it must be that $\ell^* \notin \mathbf{left}(s)$. Thus, we will have

$$\sum_{\ell\in\mathbf{left}(s)} y_{t,\ell} = 0,$$

which means that constraint (17c) is automatically satisfied, because the right hand side $x_{v(t,s)}$ is always at least 0. On the other hand, for any split $s \in \mathbf{LS}(\ell^*)$, we will have that $x_{v(t,s)} = 1$, and that

$$\sum_{\ell\in\mathbf{left}(s)} y_{t,\ell} = \sum_{\ell\in\mathbf{left}(s):\ell\neq\ell^*} y_{t,\ell} + y_{t,\ell^*} = 1,$$

which implies that constraint (17c) is satisfied. Similar reasoning can be used to establish that constraint (17d) holds. This establishes that $\mathbf{y}_t$ is indeed a feasible solution of problem (17).

Proof of part (b): By construction, $\alpha_{t,s} \geq 0$ and $\beta_{t,s} \geq 0$ for all $s \in \mathbf{splits}(t)$, so constraints (18c) and (18d) are satisfied. To verify constraint (18b), fix a leaf $\ell \in \mathbf{leaves}(t)$. If $\ell \neq \ell^*$, then either $\ell \in \mathbf{left}(s')$ for some $s' \in \mathbf{RS}(\ell^*)$ or $\ell \in \mathbf{right}(s')$ for some $s' \in \mathbf{LS}(\ell^*)$. If $\ell \in \mathbf{left}(s')$ for some $s' \in \mathbf{RS}(\ell^*)$, then

$$\begin{aligned}
&\sum_{s:\ell\in\mathbf{left}(s)} \alpha_{t,s} + \sum_{s:\ell\in\mathbf{right}(s)} \beta_{t,s} + \gamma_t \\
&\geq \alpha_{t,s'} + \gamma_t \\
&\geq \max_{\ell'\in\mathbf{left}(s')} r_{t,\ell'} - r_{t,\ell^*} + r_{t,\ell^*} \\
&\geq r_{t,\ell}
\end{aligned}$$

where the first inequality follows because $\ell \in \mathbf{left}(s')$ and the fact that all $\alpha_{t,s}$ and $\beta_{t,s}$ variables are nonnegative; the second follows by how the dual solution is defined in the statement of the theorem; and the third by the definition of the maximum. Similarly, if $\ell \in \mathbf{right}(s')$ for some $s' \in \mathbf{LS}(\ell^*)$, then by similar reasoning we have

$$\begin{aligned}
&\sum_{s:\ell\in\mathbf{left}(s)} \alpha_{t,s} + \sum_{s:\ell\in\mathbf{right}(s)} \beta_{t,s} + \gamma_t \\
&\geq \beta_{t,s'} + \gamma_t \\
&\geq \max_{\ell'\in\mathbf{right}(s')} r_{t,\ell'} - r_{t,\ell^*} + r_{t,\ell^*}
\end{aligned}$$

$$\begin{aligned}&\geq r_\ell - r_{t,\ell^*} + r_{t,\ell^*}\\ &= r_{t,\ell}.\end{aligned}$$

Lastly, if $\ell = \ell^*$, then we automatically have

$$\begin{aligned}&\sum_{s:\ell^* \in \mathbf{left}(s)} \alpha_{t,s} + \sum_{s:\ell^* \in \mathbf{right}(s)} \beta_{t,s} + \gamma_t\\ &\geq \gamma_t\\ &= r_{t,\ell^*}.\end{aligned}$$

Thus, we have established that constraint (18b) is satisfied for all leaves ℓ, and thus $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ as defined in the statement of the theorem is a feasible solution of the dual (18).

Proof of part (c): To establish that the two solutions are optimal, by weak duality it is sufficient to show that the two solutions attain the same objective values in their respective problems. For the primal solution $\mathbf{y}_t$, it is immediately clear that its objective is r_{t,ℓ^*}. For the dual solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$, we have

$$\begin{aligned}&\sum_{s \in \mathbf{splits}(t)} \alpha_{t,s} x_{v(t,s)} + \sum_{s \in \mathbf{splits}(t)} \beta_{t,s}(1 - x_{v(t,s)}) + \gamma_t\\ &= \sum_{s \in \mathbf{RS}(\ell^*)} \alpha_{t,s} x_{v(t,s)} + \sum_{s \in \mathbf{LS}(\ell^*)} \beta_{t,s}(1 - x_{v(t,s)}) + \gamma_t\\ &= 0 + 0 + \gamma_t\\ &= r_{t,\ell^*}\end{aligned}$$

where the first step follows because $\alpha_{t,s} = 0$ for $s \notin \mathbf{RS}(\ell^*)$ and $\beta_{t,s} = 0$ for $s \notin \mathbf{LS}(\ell^*)$; the second step follows by the fact that $x_{v(t,s)} = 0$ for $s \in \mathbf{RS}(\ell^*)$ and $x_{v(t,s)} = 1$ for $s \in \mathbf{LS}(\ell^*)$; and the final step follows just by the definition of γ_t. This establishes that $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for their respective problems, which concludes the proof. □

EC.2.14. Proof of Proposition 8 (ProductMIO primal and dual are NOT greedy solvable in general)

To see that the ProductMIO primal subproblem (20) is not greedy solvable, consider an instance where $\mathcal{N} = \{1, 2, 3\}$, and the revenues of these products are $\bar{r}_1 = 20$, $\bar{r}_2 = 19$ and $\bar{r}_3 = 18$. Consider the tree shown in Figure EC.5. Labeling the leaves as 1, 2, 3, 4, 5 and 6 from left to right, the

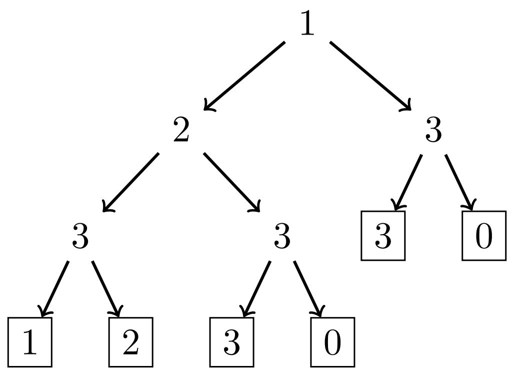


Figure EC.5 Structure of tree for which the ProductMIO primal subproblem is not solvable via a greedy algorithm.

primal subproblem (20) can be explicitly written as

$$\begin{aligned}
\underset{\mathbf{y}}{\text{maximize}} \quad & 20y_1 + 19y_2 + 18y_3 + 18y_5 && \text{(EC.61a)}\\
\text{subject to} \quad & y_1 + y_2 + y_3 + y_4 \leq 0.5 \quad (= x_1) && \text{(EC.61b)}\\
& y_5 + y_6 \leq 0.5 \quad (= 1 - x_1) && \text{(EC.61c)}\\
& y_1 + y_2 \leq 0.5 \quad (= x_2) && \text{(EC.61d)}\\
& y_3 + y_4 \leq 0.5 \quad (= 1 - x_2) && \text{(EC.61e)}\\
& y_1 + y_3 + y_5 \leq 0.5 \quad (= x_3) && \text{(EC.61f)}\\
& y_2 + y_4 + y_6 \leq 0.5 \quad (= 1 - x_3) && \text{(EC.61g)}\\
& y_1, \dots, y_6 \geq 0, && \text{(EC.61h)}
\end{aligned}$$

where we omit the subscript t to simplify notation. When we apply the greedy algorithm to solve this LO problem, we can see that there are multiple orderings of the leaves in decreasing revenue:

$$\begin{gathered}
1,2,3,5,4,6\\
1,2,5,3,4,6\\
1,2,3,5,6,4\\
1,2,5,3,6,4
\end{gathered}$$

For any of these orderings, the greedy solution will turn out to be

$$\begin{aligned}
y_1 &= 0.5,\\
y_2 &= 0,\\
y_3 &= 0,\\
y_4 &= 0,\\
y_5 &= 0,\\
y_6 &= 0.5,
\end{aligned}$$

resulting in an objective value of $y_1 \times \bar{r}_1 + y_6 \times 0 = (0.5)(20) = 10$. However, the actual optimal solution of problem (EC.61) turns out to be

$$\begin{aligned}
y_1^* &= 0,\\
y_2^* &= 0.5,\\
y_3^* &= 0,\\
y_4^* &= 0,\\
y_5^* &= 0.5,\\
y_6^* &= 0,
\end{aligned}$$

for which the objective value is $y_2^* \times \bar{r}_2 + y_5^* \times \bar{r}_3 = (0.5)(19) + (0.5)(18) = 18.5$. This shows that in general, the ProductMIO primal subproblem cannot be solved to optimality via the same type of greedy algorithm as for LeafMIO and SplitMIO. □

EC.2.15. Proof of Theorem EC.3 (ProductMIO primal and dual are closed form solvable for binary x)

Proof of part (a) (primal feasibility): The solution $\mathbf{y}_t$ clearly satisfies the unit sum constraint (20b). For $i \in \mathbf{LP}(\ell^*)$, we know that $x_i = 1$, and by the definition of ℓ^*, we have that

$$\sum_{\ell \in \mathbf{left}(i)} y_{t,\ell} = y_{t,\ell^*} + \sum_{\ell \in \mathbf{left}(i): \ell \neq \ell^*} y_{t,\ell} = 1,$$

which implies that the left split constraint (20c) is satisfied for product i. Similarly, for $i \in \mathbf{RP}(\ell^*)$, we know that $x_i = 0$ (or equivalently, $1 - x_i = 1$, and we again have that $\sum_{\ell \in \mathbf{right}(i)} y_{t,\ell} = 1$, which implies that the right split constraint (20d) is satisfied at product i.

For $i \notin \mathbf{LP}(\ell^*)$, we know that $\ell^* \notin \mathbf{left}(i)$, and thus $\sum_{\ell \in \mathbf{left}(i)} y_{t,\ell} = 0$, which implies that constraint (20c) is automatically satisfied (the right hand side is x_i which can only be 0 or 1). Similarly, for $i \notin \mathbf{RP}(\ell^*)$, we know that $\ell^* \notin \mathbf{right}(i)$ and that $\sum_{\ell \in \mathbf{right}(i)} y_{t,\ell} = 0$, which similarly implies that constraint (20d) is satisfied (the right hand side is $1 - x_i$, which can only be 0 or 1). This establishes that $\mathbf{y}_t$ is a feasible solution of problem (20).

Proof of part (b) (dual feasibility): By construction, $\alpha_{t,i} \geq 0$ and $\beta_{t,i} \geq 0$ for all products i. Thus, we only need to verify the dual constraint (EC.2b). If $\ell = \ell^*$, then the constraint is immediately satisfied, because $\gamma_t = r_{t,\ell^*}$ and $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ are nonnegative. If $\ell \neq \ell^*$, then either $\ell \in \mathbf{left}(i')$ for some $i' \in \mathbf{RP}(\ell^*)$ or $\ell \in \mathbf{right}(i')$ for some $i' \in \mathbf{LP}(\ell^*)$. In the former case, using the definition of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ and the nonnegativity of $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$, we have

$$\begin{aligned}
&\sum_{i \in \mathbf{LP}(\ell)} \alpha_{t,i} + \sum_{i \in \mathbf{RP}(\ell)} \beta_{t,i} + \gamma_t \\
&\geq \alpha_{t,i'} + \gamma_t \\
&\geq \max_{\ell' \in \mathbf{left}(i')} r_{t,\ell'} - r_{\ell^*} + r_{\ell^*} \\
&\geq r_{t,\ell}
\end{aligned}$$

In the latter case, we similarly have

$$\begin{aligned}
&\sum_{i \in \mathbf{LP}(\ell)} \alpha_{t,i} + \sum_{i \in \mathbf{RP}(\ell)} \beta_{t,i} + \gamma_t \\
&\geq \beta_{t,i'} + \gamma_t \\
&\geq \max_{\ell' \in \mathbf{right}(i')} r_{t,\ell'} - r_{\ell^*} + r_{\ell^*} \\
&\geq r_{t,\ell}.
\end{aligned}$$

This establishes that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution of problem (EC.2).

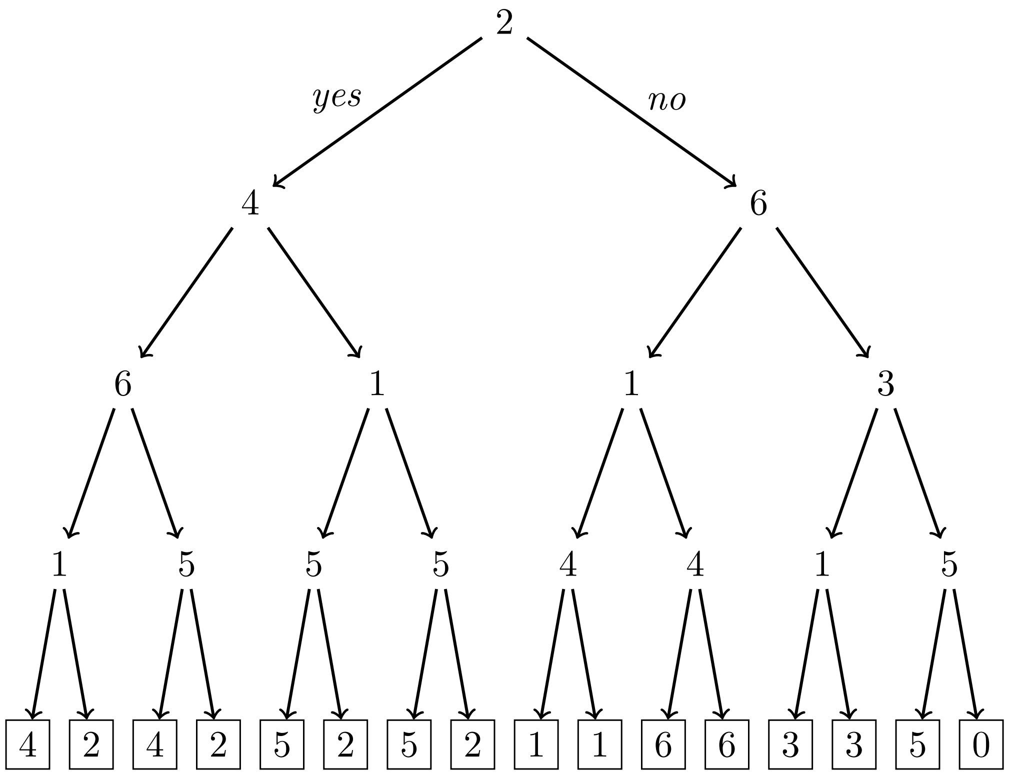
Proof of part (c) (optimality): We prove this by showing that the two solutions have the same objective value. For the primal solution, it is clear that its objective value is r_{t,ℓ^*}. For the dual solution, we have:

$$\begin{aligned}
&\sum_{i \in P(t)} \alpha_{t,i} x_i + \sum_{i \in P(t)} \beta_{t,i}(1 - x_i) + \gamma_t \\
&= \sum_{i \in \mathbf{RP}(\ell^*)} \alpha_{t,i} x_i + \sum_{i \in \mathbf{LP}(\ell^*)} \beta_{t,i}(1 - x_i) + \gamma_t \\
&= r_{t,\ell^*},
\end{aligned}$$

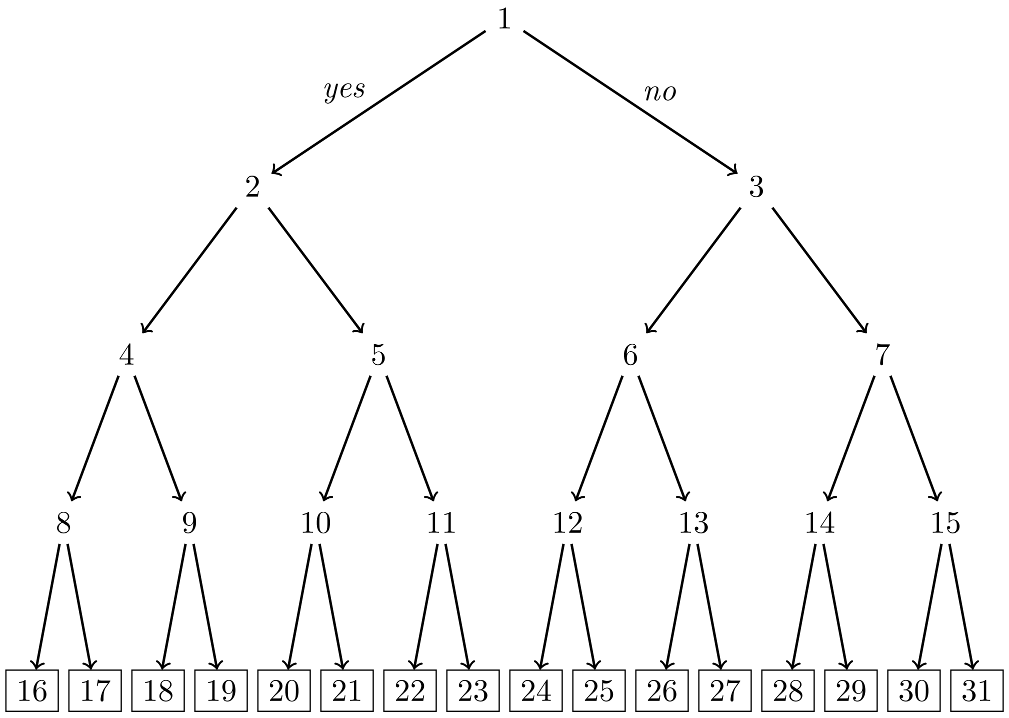
where the first step follows because $\alpha_{t,i}$ is defined to be zero when $i \notin \mathbf{RP}(\ell^*)$ and $\beta_{t,i}$ is defined to be zero when $i \notin \mathbf{LP}(\ell^*)$; and the second step follows because $x_i = 0$ when $i \in \mathbf{RP}(\ell^*)$ and $x_i = 1$ when $i \in \mathbf{LP}(\ell^*)$. This establishes that $\mathbf{y}_t$ and $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ are optimal for their respective problems. □

EC.3. Example of Benders algorithms for SplitMIO

In this section, we provide a small example of the primal-dual procedure (Algorithms 3 and 4) for solving the SplitMIO subproblem.



(a) Purchase decision tree.



(b) Indexing of nodes in tree.

Figure EC.6 **Tree used in example of SplitMIO primal-dual algorithms. The top figure shows the purchase decision tree, in terms of the products at each split, and the purchase decision at each leaf. The bottom figure shows the indexing of nodes (for example, 8 corresponds to the split node that is furthest to the bottom and to the left, while 30 corresponds to the second leaf from the right).**

Suppose that $n = 6$, and that $\mathbf{x} = (x_1, \ldots, x_6) = (0.62, 0.45, 0.32, 0.86, 0.05, 0.35)$. Suppose that $\bar{\mathbf{r}} = (\bar{r}_1, \ldots, \bar{r}_6) = (97, 72, 89, 50, 100, 68)$. Suppose that the purchase decision tree t has the form given in Figure EC.6a; in addition, suppose that the splits and leaves are indexed as in Figure EC.6b.

We first run Algorithm 3 on the problem, which carries out the steps shown below in Table EC.1. For this execution of the procedure, we assume that the following ordering of leaves (encoded by τ) is used:

$$20, 22, 30, 24, 25, 28, 29, 17, 19, 21, 23, 26, 27, 16, 18, 31.$$

Iteration	Values of q_C and $q_{A,B}$	Steps
$\ell = 20$	$q_C = 1.0$, $q_{A,B} = 0.05$	Set $y_{20} \leftarrow 0.05$ A_{10} event
$\ell = 22$	$q_C = 0.95$, $q_{A,B} = 0.05$	Set $y_{22} \leftarrow 0.05$ A_{11} event
$\ell = 30$	$q_C = 0.90$, $q_{A,B} = 0.05$	Set $y_{30} \leftarrow 0.05$ A_{15} event
$\ell = 24$	$q_C = 0.85$, $q_{A,B} = 0.35$	Set $y_{24} \leftarrow 0.35$ A_3 event
$\ell = 25$	$q_C = 0.5$, $q_{A,B} = 0.0$	Set $y_{25} \leftarrow 0.0$
$\ell = 28$	$q_C = 0.5$, $q_{A,B} = 0.15$	Set $y_{28} \leftarrow 0.15$ B_1 event
$\ell = 29$	$q_C = 0.35$, $q_{A,B} = 0.0$	Set $y_{29} \leftarrow 0.0$
$\ell = 17$	$q_C = 0.35$, $q_{A,B} = 0.35$	Set $y_{17} \leftarrow 0.35$ C event **break**

Table EC.1 Steps of primal procedure (Algorithm 3).

After running the procedure, the primal solution $\mathbf{y}$ is

$$\begin{aligned}
y_{16} &= 0.0\\
y_{17} &= 0.35\\
y_{18} &= 0.0\\
y_{19} &= 0.0\\
y_{20} &= 0.05\\
y_{21} &= 0.0\\
y_{22} &= 0.05\\
y_{23} &= 0.0\\
y_{24} &= 0.35\\
y_{25} &= 0.0\\
y_{26} &= 0.0\\
y_{27} &= 0.0\\
y_{28} &= 0.15\\
y_{29} &= 0.0
\end{aligned}$$

$$y_{30} = 0.05$$
$$y_{31} = 0.0$$

The event set is $\mathcal{E} = \{A_{10}, A_{11}, A_{15}, A_3, B_1, C\}$, and the function $f : \mathcal{E} \to$ **leaves** is defined as

$$\begin{aligned}
f(A_{10}) &= 20,\\
f(A_{11}) &= 22,\\
f(A_{15}) &= 30,\\
f(A_3) &= 24,\\
f(B_1) &= 28,\\
f(C) &= 17.
\end{aligned}$$

We now run Algorithm 4, which carries out the steps shown below in Table EC.2.

Phase	Calculation
Initialization	$\alpha_s \leftarrow 0$, $\beta_s \leftarrow 0$ for all s
Set γ	$\gamma \leftarrow r_{17} = 72$
Loop: $d = 1$	$\beta_1 \leftarrow r_{28} - \gamma = 89 - 72 = 17$
Loop: $d = 2$	$\alpha_3 \leftarrow r_{24} - \gamma - \beta_1 = 97 - 72 - 17 = 8$
Loop: $d = 4$	$\alpha_{10} \leftarrow r_{20} - \gamma = 100 - 72 = 28$
	$\alpha_{11} \leftarrow r_{22} - \gamma = 100 - 72 = 28$
	$\alpha_{15} \leftarrow r_{30} - \gamma - \beta_1 = 100 - 72 - 17 = 11$

Table EC.2 Steps of dual procedure (Algorithm 4).

After running the procedure, the dual solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ is

$$\begin{aligned}
&\gamma = 72 &&\\
&\alpha_1 = 0 && \beta_1 = 17\\
&\alpha_2 = 0 && \beta_2 = 0\\
&\alpha_3 = 8 && \beta_3 = 0\\
&\alpha_4 = 0 && \beta_4 = 0\\
&\alpha_5 = 0 && \beta_5 = 0\\
&\alpha_6 = 0 && \beta_6 = 0\\
&\alpha_7 = 0 && \beta_7 = 0\\
&\alpha_8 = 0 && \beta_8 = 0\\
&\alpha_9 = 0 && \beta_9 = 0
\end{aligned}$$

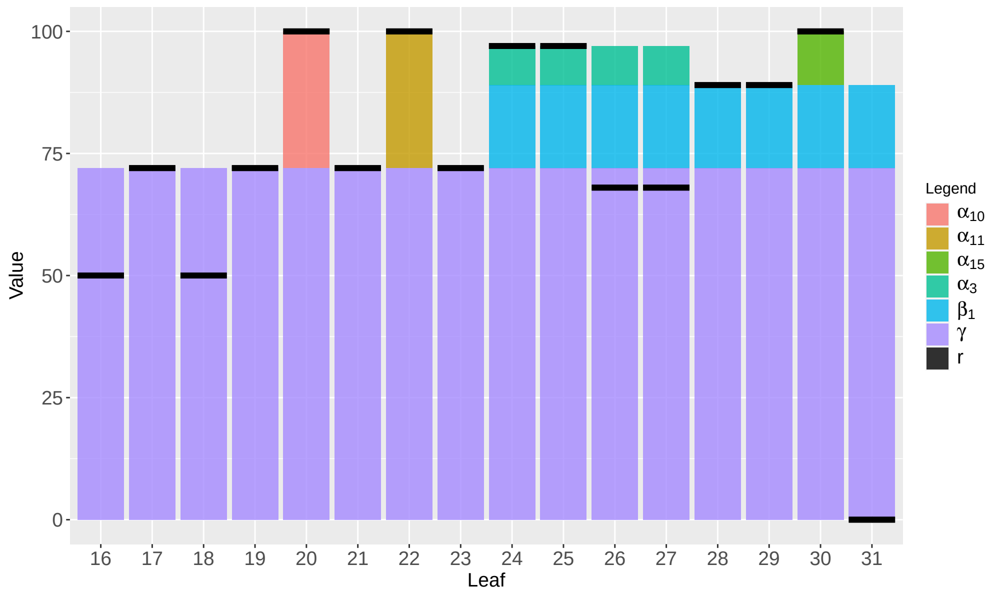


Figure EC.7 Visualization of feasibility of dual solution in SplitMIO algorithm example.

$$\begin{aligned} \alpha_{10} &= 28 & \beta_{10} &= 0 \\ \alpha_{11} &= 28 & \beta_{11} &= 0 \\ \alpha_{12} &= 0 & \beta_{12} &= 0 \\ \alpha_{13} &= 0 & \beta_{13} &= 0 \\ \alpha_{14} &= 0 & \beta_{14} &= 0 \\ \alpha_{15} &= 11 & \beta_{15} &= 0 \end{aligned}$$

The feasibility of the dual solution is visualized in Figure EC.7. The colored bars correspond to the different dual variables; a colored bar appears multiple times when the variable participates in multiple dual constraints. The height of the black lines for each leaf indicates the value of r_ℓ, while the total height of the colored bars at a leaf corresponds to the value $\gamma + \sum_{s \in \mathbf{LS}(\ell)} \alpha_s + \sum_{s \in \mathbf{RS}(\ell)} \beta_s$ (the left hand side of the dual constraint (18b). For each leaf, the total height of the colored bars exceeds the black line, which indicates that all dual constraints are satisfied.

The objective value of the primal solution is

$$\begin{aligned} & r_{20} \times y_{20} + r_{22} \times y_{22} + r_{30} \times y_{30} + r_{24} \times y_{24} + r_{28} \times y_{28} + r_{17} \times y_{17} \\ &= 100 \times 0.05 + 100 \times 0.05 + 100 \times 0.05 + 97 \times 0.35 + 89 \times 0.15 + 72 \times 0.35 \\ &= 87.5 \end{aligned}$$

The objective value of the dual solution is

$$\begin{aligned} & \gamma + (1 - x_2) \times \beta_1 + x_6 \times \alpha_3 + x_5 \times \alpha_{10} + x_5 \times \alpha_{11} + x_5 \times \alpha_{15} \\ &= 72.0 + 0.55 \times 17 + 0.35 \times 8 + 0.05 \times 28 + 0.05 \times 28 + 0.05 \times 11 \\ &= 87.5 \end{aligned}$$

EC.4. Additional Numerical Results

EC.4.1. Additional Results for Section 5.3

Table EC.3 provides the same results as Table 2 for the T1 and T2 instances.

Type	$\|F\|$	$\|\textbf{leaves}(t)\|$	$\tilde{G}_{\text{LeafMIO}}$	$\tilde{G}_{\text{SplitMIO}}$	$\tilde{G}_{\text{ProductMIO}}$	T_{LeafMIO}	T_{SplitMIO}	$T_{\text{ProductMIO}}$
T1	50	8	0.0	0.0	0.0	0.1	0.0	0.0
T1	50	16	0.0	0.0	0.0	0.4	0.2	0.0
T1	50	32	0.0	0.0	0.0	2.4	0.8	0.2
T1	50	64	0.0	0.0	0.0	40.9	7.5	1.4
T1	100	8	0.0	0.0	0.0	0.3	0.1	0.0
T1	100	16	0.0	0.0	0.0	3.7	1.5	0.3
T1	100	32	0.0	0.0	0.0	158.1	61.8	5.9
T1	100	64	1.8	0.4	0.0	3071.6	2700.6	178.7
T1	200	8	0.0	0.0	0.0	1.2	0.6	0.1
T1	200	16	0.0	0.0	0.0	334.2	380.5	13.8
T1	200	32	4.4	4.9	0.2	3600.1	3600.1	2093.5
T1	200	64	11.7	10.8	7.7	3600.1	3600.1	3600.0
T1	500	8	0.0	0.0	0.0	130.9	241.9	6.3
T1	500	16	8.1	12.0	5.0	3600.1	3600.2	3600.1
T1	500	32	17.9	19.6	14.7	3602.5	3600.1	3600.0
T1	500	64	22.9	22.6	19.8	3600.6	3601.4	3600.1
T2	50	8	0.0	0.0	0.0	0.1	0.0	0.0
T2	50	16	0.0	0.0	0.0	0.5	0.2	0.2
T2	50	32	0.0	0.0	0.0	5.4	0.8	0.8
T2	50	64	0.0	0.0	0.0	859.9	84.7	56.8
T2	100	8	0.0	0.0	0.0	0.3	0.1	0.1
T2	100	16	0.0	0.0	0.0	23.4	3.0	3.3
T2	100	32	2.9	0.0	0.0	3334.3	1436.0	767.4
T2	100	64	9.1	6.9	6.6	3600.3	3600.1	3600.0
T2	200	8	0.0	0.0	0.0	3.4	0.6	0.6
T2	200	16	6.1	1.8	1.4	3566.4	2683.8	2347.2
T2	200	32	15.0	12.9	12.6	3600.2	3600.1	3600.1
T2	200	64	20.5	18.5	18.5	3600.2	3600.2	3600.1
T2	500	8	1.9	0.0	0.0	3000.0	981.1	591.6
T2	500	16	20.8	19.1	18.8	3600.1	3600.3	3600.1
T2	500	32	27.9	25.4	25.0	3600.2	3600.1	3600.1
T2	500	64	33.5	30.7	30.4	3601.4	3600.1	3600.1

Table EC.3 Comparison of final optimality gaps and computation times for LeafMIO, SplitMIO and ProductMIO, for T1 and T2 instances.

EC.4.2. Comparison to Heuristic Approaches

In this experiment, we compare the performance of the three formulations to heuristic approaches. We will consider three different heuristic approaches:

1. LS: A local search heuristic, which starts from the empty assortment, and in each iteration moves to the neighboring assortment which improves the expected revenue the most. The neighborhood of assortments consists of those assortments obtained by adding a new product to the current assortment, or removing one of the existing products from the assortment. The heuristic terminates when there is no assortment in the neighborhood of the current one that provides an improvement.
2. LS10: This heuristic involves running LS from ten randomly chosen starting assortments. Each assortment is chosen uniformly at random from the set of 2^n possible assortments. After the ten repetitions, the assortment with the best expected revenue is retained.
3. ROA: This heuristic involves finding the optimal revenue ordered assortment. More formally, we define $S_k = \{i_1, \dots, i_k\}$, where $i_1, \dots, i_n$ corresponds to an ordering of the products so that $r_{i_1} \geq r_{i_2} \geq \dots \geq r_{i_n}$, and we find $\arg\max_{S \in \{S_1, \dots, S_n\}} R^{(F, \boldsymbol{\lambda})}(S)$.

We compare these heuristics against the best integer solution obtained by each of our three MIO formulations, leading to a total of six methods for each instance. We measure the performance of the solution corresponding to approach $\mathcal{M}$ using the metric $\bar{G}_{\mathcal{M}}$, which is defined as

$$\bar{G}_{\mathcal{M}} = 100\% \times \frac{Z_{BestUB} - Z_{\mathcal{M}}}{Z_{BestUB}},$$

where Z_{BestUB} is the best (lowest) upper bound obtained from among the three MIO formulations, and $Z_{\mathcal{M}}$ is the objective value of the solution returned by approach $\mathcal{M}$.

Table EC.4 shows the performance of the six approaches – LeafMIO, SplitMIO, ProductMIO, LS, LS10 and ROA – for each family of instances. The gaps are averaged over the twenty instances for each combination of instance type, $|F|$ and $|\mathbf{leaves}(t)|$. We can see from this table that in general, for the small instances, the solutions obtained by the MIO formulations are either optimal or near optimal, while the solutions produced by the heuristic approaches are quite suboptimal. For cases where the gap is zero for the MIO formulations, the gap of LS ranges from 1.5% to 14.4%; the LS10 heuristic improves on this, due to its use of restarting and randomization, but still does not perform as well as the MIO solutions (gaps ranging from 0.5 to 7.4%). For the larger instances, where the gap of the MIO solutions is larger, LS and LS10 still tend to perform worse. Across all of the instances, ROA achieves much higher gaps than all of the other approaches (ranging from 14.6 to 40.6%). Overall, these results suggest that the very general structure of the decision forest model poses significant difficulty to standard heuristic approaches, and highlight the value of using exact approaches over inexact/heuristic approaches to the assortment optimization problem.

Type	$\|F\|$	$\|\mathbf{leaves}(t)\|$	$\bar{G}_{\text{LeafMIO}}$	$\bar{G}_{\text{SplitMIO}}$	$\bar{G}_{\text{ProductMIO}}$	$\bar{G}_{\text{LS}}$	$\bar{G}_{\text{LS10}}$	$\bar{G}_{\text{ROA}}$
T1	50	8	0.0	0.0	0.0	8.1	2.3	26.8
T1	50	16	0.0	0.0	0.0	9.8	4.0	31.2
T1	50	32	0.0	0.0	0.0	9.0	4.8	27.5
T1	50	64	0.0	0.0	0.0	10.0	6.4	28.6
T1	100	8	0.0	0.0	0.0	3.9	2.2	26.0
T1	100	16	0.0	0.0	0.0	6.6	3.5	26.2
T1	100	32	0.0	0.0	0.0	8.5	6.2	25.6
T1	100	64	0.0	0.0	0.0	9.9	6.8	24.5
T1	200	8	0.0	0.0	0.0	3.5	1.2	19.9
T1	200	16	0.0	0.0	0.0	5.5	2.7	21.7
T1	200	32	0.2	0.2	0.2	7.9	5.1	22.4
T1	200	64	8.2	7.8	7.7	14.6	13.7	25.6
T1	500	8	0.0	0.0	0.0	1.5	0.5	14.6
T1	500	16	5.0	5.2	5.0	8.5	6.6	19.5
T1	500	32	15.1	15.0	14.7	19.0	17.1	28.0
T1	500	64	20.3	20.3	19.8	23.3	22.2	30.4
T2	50	8	0.0	0.0	0.0	13.8	2.7	31.5
T2	50	16	0.0	0.0	0.0	11.6	4.3	32.2
T2	50	32	0.0	0.0	0.0	10.0	4.9	31.1
T2	50	64	0.0	0.0	0.0	11.6	6.8	30.4
T2	100	8	0.0	0.0	0.0	5.5	1.6	28.1
T2	100	16	0.0	0.0	0.0	8.2	3.6	30.8
T2	100	32	0.0	0.0	0.0	8.9	5.7	31.3
T2	100	64	7.1	6.6	6.6	17.3	12.2	31.8
T2	200	8	0.0	0.0	0.0	3.8	1.0	23.1
T2	200	16	1.4	1.4	1.4	6.8	3.9	25.3
T2	200	32	12.9	12.6	12.5	18.5	16.3	34.0
T2	200	64	19.3	18.3	18.5	24.4	21.2	37.0
T2	500	8	0.0	0.0	0.0	2.0	0.5	15.6
T2	500	16	18.9	18.8	18.7	21.5	19.6	31.7
T2	500	32	26.3	25.2	25.0	28.2	25.8	35.9
T2	500	64	32.4	30.5	30.4	31.4	29.5	38.4
T3	50	8	0.0	0.0	0.0	13.2	3.3	33.1
T3	50	16	0.0	0.0	0.0	14.4	5.9	34.9
T3	50	32	0.0	0.0	0.0	12.6	6.1	33.7
T3	50	64	0.0	0.0	0.0	13.9	7.4	33.0
T3	100	8	0.0	0.0	0.0	8.0	1.4	30.2
T3	100	16	0.0	0.0	0.0	10.0	3.4	32.6
T3	100	32	0.0	0.0	0.0	10.4	3.9	32.0
T3	100	64	3.6	3.5	3.5	13.1	8.9	33.4
T3	200	8	0.0	0.0	0.0	4.0	0.8	25.8
T3	200	16	0.0	0.0	0.0	6.5	2.2	26.5
T3	200	32	8.7	8.6	8.6	15.4	11.1	33.1
T3	200	64	16.0	15.7	15.6	23.0	18.3	35.6
T3	500	8	0.0	0.0	0.0	2.6	0.5	15.8
T3	500	16	13.7	13.7	13.8	17.4	14.4	27.9
T3	500	32	23.7	22.9	23.0	26.5	23.4	34.8
T3	500	64	31.7	30.9	30.8	33.0	30.2	40.6

Table EC.4 Comparison of integer solutions from LeafMIO, SplitMIO and ProductMIO against heuristic solutions from LS, LS10 and ROA.