Optimization of Tree Ensembles

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

Tree ensemble models such as random forests and boosted trees are among the most widely used and practically successful predictive models in applied machine learning and business analytics. Although such models have been used to make predictions based on exogenous, uncontrollable independent variables, they are increasingly being used to make predictions where the independent variables are controllable and are also decision variables. In this paper, we study the problem of tree ensemble optimization: given a tree ensemble that predicts some dependent variable using controllable independent variables, how should we set these variables so as to maximize the predicted value? We formulate the problem as a mixed-integer optimization problem. We theoretically examine the strength of our formulation, provide a hierarchy of approximate formulations with bounds on approximation quality and exploit the structure of the problem to develop two large-scale solution methods, one based on Benders decomposition and one based on iteratively generating tree split constraints. We test our methodology on real data sets, including two case studies in drug design and customized pricing, and show that our methodology can efficiently solve large-scale instances to near or full optimality, and outperforms solutions obtained by heuristic approaches. In our drug design case, we show how our approach can identify compounds that efficiently trade-off predicted performance and novelty with respect to existing, known compounds. In our customized pricing case, we show how our approach can efficiently determine optimal store-level prices under a random forest model that delivers excellent predictive accuracy.

Key words: tree ensembles; random forests; mixed-integer optimization; drug design; customized pricing.

1. Introduction

A decision tree is a form of predictive model used for predicting a dependent variable Y using a collection of independent variables $\mathbf{X} = (X_1, \ldots, X_n)$. To make a prediction, we start at the root of the tree, and check a query (e.g., "Is $X_3 \leq 5.6$?") at the root of the tree; depending on the answer to the query (true/false) we proceed to another node. We then check the new node's query; the process continues until we reach a leaf node, where the tree outputs a prediction. A generalization of this type of model, called a tree ensemble model, involves making this type of prediction from each of a collection of trees and aggregating the individual predictions into a single prediction (for example, by taking a weighted average of the predictions for a regression setting, or by taking a majority vote of the predictions for a classification setting).

Many types of tree ensemble models have been proposed in the machine learning literature; the most notable examples are random forests and boosted trees. Tree ensemble models are extremely attractive due to their ability to model complex, nonlinear relationships between the independent variables $\mathbf{X}$ and the dependent variable Y. As a result, tree ensemble models in general, have gained or are gaining widespread popularity in a number of application areas; examples include chemistry (Svetnik et al. 2003), genomics (Díaz-Uriarte and De Andres 2006), ecology (Elith et al. 2008, Cutler et al. 2007), economics (Varian 2014, Bajari et al. 2015), marketing (Lemmens and Croux 2006) and operations management (Ferreira et al. 2015).

In many applications of tree ensemble models, and predictive models in general, the independent variables that are used for prediction are exogenous and beyond our control as the decision maker. For example, one might build a random forest model to predict whether a patient is at risk of developing a disease based on the patient's age, blood pressure, family history of the disease and so

on. Clearly, features like age and family history are not amenable to intervention. Such models are typically used for some form of post hoc action or prioritization. In the disease risk example, one can use the random forest model to rank patients by decreasing predicted risk of an imminent acute event, and this ranking can be used to determine which patients should receive some intervention (e.g., closer monitoring, treatment with a particular drug and so on).

In an increasing number of predictive modeling applications, however, some of the independent variables are controllable; that is to say, those independent variables are also *decision* variables. We provide a couple of examples:

1. **Design of drug therapies**. In a healthcare context, one might be interested in building a model to predict patient response given a particular drug therapy, and then using such a model to find the optimal drug therapy. The dependent variable is some metric of the patient's health, and the independent variables may specify the drug therapy (which drugs and in what doses) and characteristics of the patient. A recent example of such an approach is the paper of Bertsimas et al. (2016b) that considers the design of clinical trials for combination drug chemotherapy for gastric cancer; the first step involves estimating a model (specifically, a ridge regression model) of patient survival and toxicity using information about the drug therapy, and the second step involves solving an optimization problem to find the drug therapy that maximizes the predicted survival of the given patient group subject to a constraint on the predicted toxicity.

2. **Pricing/promotion planning**. In marketing and operations management, a fundamental problem is that of deciding which products should be promoted when and at what price. In such a context, the data might consist of weekly sales of each product in a category, and the prices of the products in that category during that week and in previous weeks; one might use this data to build a predictive model of demand as a function of the prices, and then optimize such a model to decide on a promotion schedule (for a recent example see, e.g., Cohen et al. 2017).

In this paper, we seek to unlock the prescriptive potential of tree ensemble models by considering the problem of *tree ensemble optimization*. This problem is concerned with the following question: given a tree ensemble model that predicts some quantity of interest using a set of controllable independent variables $\mathbf{X}$, how should we set the independent variables $\mathbf{X}$ so as to maximize the predicted value of Y? This problem is of significant practical interest because it allows us to leverage the high accuracy afforded by tree ensemble models to obtain high quality decisions. At the same time, the problem is also challenging and it is not obvious how to solve such a problem, due to the highly nonlinear and large-scale nature of tree ensemble models.

We make the following contributions:

1. We propose the problem of tree ensemble optimization problem and we show how to formulate this problem as a finite mixed-integer (MIO) optimization problem. The formulation can accommodate independent variables that are discrete, categorical variables as well as continuous, numeric variables. To the best of our knowledge, the problem of optimizing an objective function described as the prediction of a tree ensemble has not been previously proposed in either the machine learning or operations research communities.

2. From a theoretical standpoint, we develop a number of results that generally concern the tractability of the formulation. First, we prove that the tree ensemble optimization problem is in general NP-Hard. We then show that our proposed MIO formulation offers a tighter relaxation of the problem than an alternate MIO formulation, obtained by applying a standard linearization to a binary polynomial formulation of the problem. We develop a hierarchy of approximate formulations for the problem, obtained by truncating each tree in the ensemble to a depth d from the root node. We prove that the objective value of such an approximate formulation is an upper bound that improves as d increases, and show how to construct a complementary a priori lower bound that depends on the variability of each tree's prediction below the truncation depth d.

3. From a solution methodology standpoint, we present two different algorithms for tackling large-scale instances of our MIO formulation. The first is based on solving a Benders reformulation of the problem using constraint generation. Here, we analyze the structure of the Benders subproblem and show that it can be solved efficiently. The second is based on applying lazy constraint

generation directly to our MIO formulation. For this approach, we propose an efficient procedure for identifying violated constraints, which involves simply traversing each tree in the ensemble, and prove its correctness.

4. We evaluate the effectiveness of our formulation and solution methods computationally using an assortment of real data sets. We show that the full MIO formulation can be solved to full optimality for small to medium sized instances within minutes, and that our formulation is significantly stronger in terms of relaxation bound and solution time than the aforementioned alternate formulation. We also show that our approach often significantly outperforms a simple local search heuristic that does not guarantee optimality. Lastly, we show that our customized solution methods can drastically reduce the solution time of our formulation.

5. We provide a deeper showcase of the utility of our approach in two applications. The first is a case study in drug design, using a publicly available data set from Merck Research Labs (Ma et al. 2015). Here, we show that our approach can optimize large-scale tree ensemble models with thousands of independent variables to full or near optimality within a two hour time limit, and can be used to construct a Pareto frontier of compounds that efficiently trade off predicted performance and similarity to existing, already-tested compounds. The second is a case study in customized pricing using a supermarket scanner data set (Montgomery 1997). Here, we show that a random forest model leads to considerable improvements in out-of-sample prediction accuracy than two state-of-the-art models based on hierarchical Bayesian regression, and that our optimization approach can be used to efficiently determine provably optimal prices at the individual store level within minutes.

The rest of the paper is organized as follows. In Section 2, we survey some of the related literature to this work. In Section 3, we present our formulation of the tree ensemble optimization problem as an MIO problem, and provide theoretical results on the structure of this problem. In Section 4, we present two solution approaches for large-scale instances of the tree ensemble optimization problem. In Section 5, we present the results of our computational experiments with real data sets, and in Sections 6 and 7 we present the results of our two case studies in drug design and customized pricing, respectively. In Section 8, we conclude.

2. Literature review

Decision tree models became popular in machine learning with the introduction of two algorithms, ID3 (iterative dichotomiser; see Quinlan 1986) and CART (classification and regression tree; see Breiman et al. 1984). Decision tree models gained popularity due to their interpretability, but were found to be generally less accurate than other models such as linear and logistic regression. A number of ideas were thus consequently proposed for improving the accuracy of decision tree models, which are all based on constructing an *ensemble* of tree models. Breiman (1996) proposed the idea of bootstrap aggregation, or *bagging*, where one builds a collection of predictive models, each trained with a bootstrapped sample of the original training set; the predictions of each model are then aggregated into a single prediction (for classification, this is by majority vote; for regression, this is by averaging). The motivation for bagging is that it reduces the prediction error for predictive models that are unstable/highly sensitive to the training data (such as CART); indeed, Breiman (1996) showed that bagged regression trees can be significantly better than ordinary regression trees in out-of-sample prediction error. Later, Breiman (2001) proposed the random forest model, where one builds a collection of bagged CART tree for which the subset of features selected for splitting at each node of each tree is randomly sampled from the set of all features (the so-called *random subspace* method; see Ho 1998). Concurrently, a separate stream of literature has considered the idea of *boosting* (Schapire and Freund 2012), wherein one iteratively builds a weighted collection of basic predictive models (such as CART trees), with the goal of reducing the prediction error with each iteration.

Tree ensembles occupy a central place in machine learning because they generally work very well in practice. In a systematic comparison of 179 different prediction methods on a broad set

of benchmark data sets, Fernández-Delgado et al. (2014) found that random forests achieved best or near-best performance over all of these data sets. Boosted trees have been similarly successful: on the data science competition website Kaggle, one popular implementation of boosted trees, `XGBoost`, was used in more than half of the winning solutions in the year 2015 (Chen and Guestrin 2016). There exist robust and open source implementations of many tree ensemble models. For boosted trees, the R package `gbm` (Ridgeway 2006) and `XGBoost` are widely used; for random forests, the R package `randomForest` (Liaw and Wiener 2002) is extremely popular.

At the same time, there has been a significant effort in the machine learning research community to develop a theoretical foundation for tree ensemble methods; we briefly survey some of the work in this direction for random forests. For random forests, the original paper of Breiman (2001) developed an upper bound on the generalization error of a random forest. Later research studied the consistency of both simplified versions of the random forest model (for example, Biau et al. 2008) as well as the original random forest model (for example, Scornet et al. 2015). Recently, Wager and Walther (2015) showed that certain forms of regression trees and random forests converge uniformly over the feature space to the true regression function, while Wager and Athey (2015) considered how to use random forests for causal inference. For an excellent overview of recent theoretical advances in random forests, the reader is referred to Biau and Scornet (2016).

At the same time, there is an increasing number of papers originating in operations research where a predictive model is used to represent the effect of the decision, and one solves an optimization problem to find the best decision with respect to this predictive model. Aside from the papers already mentioned in clinical trials and promotion planning, we mention a few other examples. In pricing, data on historical prices and demand observed at those prices is often used to build a predictive model of demand as a function of price and to then determine the optimal price (see for example Besbes et al. 2010, Bertsimas and Kallus 2016). In assortment optimization, the paper of Bertsimas and Mišić (2016) considers the problem of selecting an assortment (a set of products) given data on historical assortments; the paper proposes an approach based on first estimating a ranking-based model of choice, and then solving an MIO model to find an assortment that maximizes the expected revenue predicted by this ranking-based model of choice.

In the research literature where predictive models are used to understand and subsequently optimize decisions, the closest paper conceptually to this one is the paper of Ferreira et al. (2015). This paper considers the problem of pricing weekly sales for an online fashion retailer. To solve the problem, the paper builds a random forest model of the demand of each style included in a sale as a function of the style's price and the average price of the other styles. The paper then formulates an MIO problem to determine the optimal prices for the styles to be included in the sale, where the revenue is based on the price and the demand (as predicted by the random forest) of each style. The MIO formulation of Ferreira et al. (2015) does not explicitly model the random forest prediction mechanism – instead, one computes the prediction of the random forest for each style at each of its possible prices and at each possible average price, and these predictions enter the model as coefficients in the objective function. (The predictions could just as easily have come from a different form of predictive model, without changing the structure of the optimization problem.) In contrast, our MIO formulation explicitly represents the structure of each tree in the ensemble, allowing the prediction of each tree to be determined through the variables and constraints of the MIO. Although the modeling approach of Ferreira et al. (2015) is feasible for their pricing problem, it is difficult to extend this approach when there are many independent variables, as one would need to enumerate all possible combinations of values for them and compute the tree ensemble's prediction for each combination of values. To the best of our knowledge, our paper is the first to conceptualize the problem of how to optimize an objective function that is given by a tree ensemble.

Methodologically, the present paper is most related to the paper of Bertsimas and Mišić (2016). It turns out that the ranking-based model considered in Bertsimas and Mišić (2016) can be understood as a type of tree ensemble model; as such, the MIO formulation of Bertsimas and Mišić (2016) can be regarded as a special case of the more general formulation that we analyze here. Some

of the theoretical results found in Bertsimas and Mišić (2016) – specifically, those results on the structure of the Benders cuts – are generalized in the present paper to tree ensemble models. Despite this similarity, the goals of the two papers are different. Bertsimas and Mišić (2016) considers an estimation and optimization approach specifically for assortment decisions, whereas in the present paper, we focus solely on an optimization framework that can be applied to *any* tree ensemble model, thus spanning a significantly broader range of application domains. Indeed, later in the paper we will present two different case studies – one on drug design (Section 6) and one on customized pricing (Section 7) – to illustrate the broad applicability of tree ensemble optimization.

Finally, we note that there is a growing literature on the use of mixed-integer optimization for the purpose of estimating decision tree models and other forms of statistical models. Specifically, Bertsimas and Dunn (2017) consider an exact MIO approach to constructing CART trees. Outside of decision tree models, Bertsimas and King (2015) consider an MIO approach to model selection in linear regression; and Bertsimas et al. (2016a) consider an MIO approach to best subset selection in linear regression. While the present paper is related to this previous work in that it also leverages the technology of MIO, the goal of the present paper is different. The above papers focus on the estimation of trees and other statistical models, whereas our paper is focused on optimization, namely, how to determine the optimal *decision* with respect to a given, fixed tree ensemble model.

3. Model

We begin by providing some background on tree ensemble models in Section 3.1 and defining the tree ensemble optimization problem. We then present our mixed-integer optimization model in Section 3.2. We provide results on the strength of our formulation in Section 3.3. Finally, in Section 3.4, we describe a hierarchy of approximate formulations based on depth truncation.

3.1. Background

In this section, we provide some background on tree-based models. We are given the task of predicting a dependent variable Y using the independent variables $X_1, \ldots, X_n$; for convenience, we use $\mathbf{X}$ to denote the vector of independent variables. We let $\mathcal{X}_i$ denote the domain of independent variable i and let $\mathcal{X} = \prod_{i=1}^{n} \mathcal{X}_i$ denote the domain of $\mathbf{X}$. An independent variable i may be a numeric variable or a categorical variable.

A decision tree is a model for predicting the dependent variable Y using the independent variable $\mathbf{X}$ by checking a collection of *splits*. A split is a condition or query on a single independent variable that is either true or false. More precisely, for a numeric variable i, a split is a query of the form

$$\text{Is } X_i \leq a?$$

for some $a \in \mathbb{R}$. For a categorical variable i, a split is a query of the form

$$\text{Is } X_i \in A?$$

where $A \subseteq \mathcal{X}_i$ is a set of levels of the categorical variable. The splits are arranged in the form of a tree, with each split node having two child nodes. The left child corresponds to the split condition being true, while the right child corresponds to the condition being false. To make a prediction for an observation with the independent variable $\mathbf{X}$, we start at the split at the root of the tree and check whether $\mathbf{X}$ satisfies the split condition; if it is true, we move to the left child, and if it is false, we move to the right child. At the new node, one checks the split again, and move again to the corresponding node. This process continues until we reach a leaf of the tree. The prediction that we make is the value corresponding to the leaf we have reached. An example of a decision tree and a prediction being made is given in Figure 1.

In this paper, we will focus on predictive models that are ensembles or collections of decision trees. We assume that there are T trees, where each tree is indexed from 1 to T. Each tree t has a weight λ_t, and its prediction is denoted by the function f_t; for the independent variable $\mathbf{X}$, the

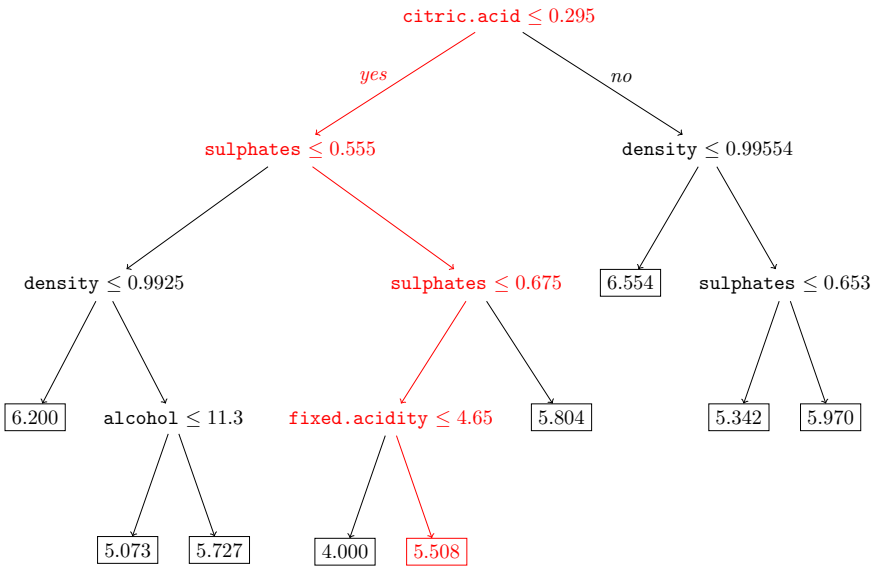

Figure 1 **Example of a decision tree based on the `winequalityred` data set (see Section 5.1). The goal is to predict the quality rating of a (red) wine using chemical properties of the wine. The nodes and edges colored in red indicate how an observation with `citric.acid` $= 0.22$, `density` $= 0.993$, `sulphates` $= 0.63$, `alcohol` $= 10.6$, `fixed.acidity` $= 4.9$, is mapped to a prediction (value of 5.508).**

prediction of tree t is $f_t(\mathbf{X})$. For an observation with independent variable $\mathbf{X}$, the prediction of the ensemble of trees is given by

$$\sum_{t=1}^{T} \lambda_t f_t(\mathbf{X}).$$

The optimization problem that we would like to solve is to find the value of the independent variable $\mathbf{X}$ that maximizes the ensemble prediction:

$$\underset{\mathbf{X} \in \mathcal{X}}{\text{maximize}} \sum_{t=1}^{T} \lambda_t f_t(\mathbf{X}). \tag{1}$$

We shall make several assumptions about the tree ensemble model $\sum_{t=1}^{T} \lambda_t f_t(\cdot)$ and our tree ensemble optimization problem. First, we assume that we are only making a single, one-time decision and that the tree model is fixed. Extending our approach to the setting of multistage decisions is an interesting direction for future research. Second, we assume that the tree model is an accurate representation of the outcome when we make the decision $\mathbf{X}$. In practice, some care must be taken here because depending on how the tree ensemble model is estimated and the nature of the data, the prediction $\sum_{t=1}^{T} \lambda_t f_t(\mathbf{X})$ may not necessarily be an accurate estimate of the causal effect of setting the independent variables to $\mathbf{X}$. This issue has been the focus of some recent work in prescriptive analytics (see Bertsimas and Kallus 2016, Kallus 2016). Our goal in this paper is to address only the question of optimization – how to efficiently and scalably optimize a tree ensemble function $\sum_{t=1}^{T} \lambda_t f_t(\cdot)$ – which is independent of such statistical questions. As such, we will assume that the tree ensemble model we are given at the outset is beyond suspicion.

Problem (1) is very general, and one question we may have is whether it is theoretically tractable or not. Our first theoretical result answers this question in the negative.

PROPOSITION 1. *The tree ensemble optimization problem* (1) *is NP-Hard.*

The proof of Proposition 1, given in Section EC.1.2 of the ecompanion, follows by reducing the minimum vertex cover problem to problem (1).

3.2. Optimization model

We now present an MIO formulation of problem (1). Before we present the model, we will require some additional notation. We let $\mathcal{N}$ denote the set of numeric variables and $\mathcal{C}$ denote the set of categorical variables; we have that $\mathcal{N} \cup \mathcal{C} = \{1, \ldots, n\}$.

For each numeric variable $i \in \mathcal{N}$, let $\mathcal{A}_i$ denote the set of unique split points, that is, the set of values a such that $X_i \leq a$ is a split condition in some tree in the ensemble $\{f_t\}_{t=1}^T$. Let $K_i = |\mathcal{A}_i|$ be the number of unique split points. Let $a_{i,j}$ denote the jth smallest split point of variable i, so that $a_{i,1} < a_{i,2} < \cdots < a_{i,K_i}$.

For each categorical variable $i \in \mathcal{C}$, recall that $\mathcal{X}_i$ is the set of possible values of i. For convenience, let us also use K_i in this case to denote the size of $\mathcal{X}_i$ (i.e., $K_i = |\mathcal{X}_i|$) and use the values $1, 2, \ldots, K_i$ to denote the possible levels of variable i.

Let $\mathbf{leaves}(t)$ be the set of leaves or terminal nodes of tree t. Let $\mathbf{splits}(t)$ denote the set of splits of tree t (non-terminal nodes). Recall that the left branch of the split corresponds to "yes" or "true" to the split query, and the right branch corresponds to "no" or "false". Therefore, for each split s in S_t, we let $\mathbf{left}(s)$ be the set of leaves that are accessible from the left branch (all of the leaves for which the condition of split s must be true), and $\mathbf{right}(s)$ be the set of leaves that are accessible from the right branch (all of the leaves for which the condition of split s must be false). For each split s, we let $\mathbf{V}(s) \in \{1, \ldots, n\}$ denote the variable that participates in split s, and let $\mathbf{C}(s)$ denote the set of values of variable i that participate in the split query of s. Specifically, if $\mathbf{V}(s)$ is numeric, then $\mathbf{C}(s) = \{j\}$ for some $j \in \{1, \ldots, K_{\mathbf{V}(s)}\}$, which corresponds to the split query $X_i \leq a_{i,j}$. If $\mathbf{V}(s)$ is categorical, then $\mathbf{C}(s) \subseteq \{1, \ldots, K_{\mathbf{V}(s)}\}$, which corresponds to the query $X_i \in \mathbf{C}(s)$.

Recall that λ_t is the weight of tree t. For each tree t, we denote the set of leaves by $\mathbf{leaves}(t)$. For each leaf $\ell \in \mathbf{leaves}(t)$, we use $p_{t,\ell}$ to denote the prediction that tree t makes when an observation reaches leaf ℓ.

We now define the decision variables of the problem. There are two sets of decision variables. The first set is used to specify the independent variable value $\mathbf{X}$. For each categorical independent variable $i \in \mathcal{C}$ and each category/level $j \in \mathcal{X}_i$, we let $x_{i,j}$ be 1 if independent variable i is set to level j, and 0 otherwise. For each numeric independent variable $i \in \mathcal{N}$ and each $j \in \{1, \ldots, K_i\}$, we let $x_{i,j}$ be 1 if independent variable i is set to a value less than or equal to the jth split point, and 0 otherwise. Mathematically,

$$\begin{aligned} x_{i,j} &= \mathbb{I}\{X_i = j\}, \quad \forall i \in \mathcal{C},\ j \in \{1, \ldots, K_i\}, \\ x_{i,j} &= \mathbb{I}\{X_i \leq a_{i,j}\}, \quad \forall i \in \mathcal{N},\ j \in \{1, \ldots, K_i\}. \end{aligned}$$

We use $\mathbf{x}$ to denote the vector of $x_{i,j}$ values.

The second set of decision variables is used to specify the prediction of each tree t. For each tree t and each leaf $\ell \in \mathbf{leaves}(t)$, we let $y_{t,\ell}$ be a binary decision variable that is 1 if the observation encoded by $\mathbf{x}$ belongs to/falls into leaf ℓ of tree t, and 0 otherwise.

With these definitions, the mixed-integer optimization can be written as follows:

$$\underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{t=1}^{T} \sum_{\ell \in \mathbf{leaves}(t)} \lambda_t \cdot p_{t,\ell} \cdot y_{t,\ell} \tag{2a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, \quad \forall\ t \in \{1, \ldots, T\}, \tag{2b}$$

$$\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}, \quad \forall\ t \in \{1, \ldots, T\},\ s \in \mathbf{splits}(t), \tag{2c}$$

$$\sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j},$$

$$\forall\ t \in \{1,\dots,T\},\ s \in \mathbf{splits}(t), \tag{2d}$$

$$\sum_{j=1}^{K_i} x_{i,j} = 1, \quad \forall\ i \in \mathcal{C}, \tag{2e}$$

$$x_{i,j} \leq x_{i,j+1}, \quad \forall\ i \in \mathcal{N},\ j \in \{1,\dots,K_i - 1\}, \tag{2f}$$

$$x_{i,j} \in \{0,1\}, \quad \forall\ i \in \{1,\dots,n\},\ j \in \{1,\dots,K_i\} \tag{2g}$$

$$y_{t,\ell} \geq 0, \quad \forall\ t \in \{1,\dots,T\},\ \ell \in \mathbf{leaves}(t). \tag{2h}$$

In order of appearance, the constraints have the following meaning. Constraint (2b) ensures that the observation falls in exactly one of the leaves of each tree t. Constraint (2c) ensures that, if $\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} = 0$, then $y_{t,\ell}$ is forced to zero for all $\ell \in \mathbf{left}(s)$; in words, the condition of the split is false, so the observation cannot fall into any leaf to the left of split s, as this would require the condition to be true. Similarly, constraint (2d) ensures that if the condition of split s *is* satisfied, then $y_{t,\ell}$ is forced to zero for all $\ell \in \mathbf{right}(s)$; in words, the condition of the split is true, so the observation cannot fall into any leaf to the right of split s, as this would require the condition to be false. Constraint (2e) ensures that for each categorical variable $i \in \mathcal{C}$, exactly one of the K_i levels is selected. Constraint (2f) requires that if the numeric independent variable i is less than or equal to the jth lowest split point, then it must also be less than or equal to the $j+1$th lowest split point. Constraint (2g) defines each $x_{i,j}$ to be binary, while constraint (2h) defines each $y_{t,\ell}$ to be nonnegative. The objective represents the prediction of the ensemble of trees on the observation that is encoded by $\mathbf{x}$.

We now comment on several features of the model. The first is that the $y_{t,\ell}$ variables, despite having a binary meaning, are defined as continuous variables. The reason for this is that when $\mathbf{x}$ is binary, the constraints automatically force $\mathbf{y}$ to be binary. We will formally state this result later (Proposition 4 of Section 4.1). As a result, the only binary variables are those in $\mathbf{x}$, of which there are $\sum_{i=1}^{n} K_i$. Recall that for categorical independent variables, K_i is the number of levels, whereas for numeric independent variables, K_i is the number of unique split points found in the tree ensemble $\{f_t\}_{t=1}^{T}$.

The second is that our formulation does not model the exact value of each numeric independent variable $i \in \mathcal{N}$. In contrast, the formulation only models where the variable is in relation to the unique split points in $\mathcal{A}_i$ – for example, $x_{i,1} = 1$ indicates that independent variable i is set to be less than or equal to the first lowest split point. The reason for this is that each decision tree function $f_t(\cdot)$ is a piecewise constant function and therefore the ensemble tree function $\sum_{t=1}^{T} \lambda_t f_t(\mathbf{X})$ is also piecewise constant. Thus, for the purpose of optimizing the function $\sum_{t=1}^{T} \lambda_t f_t(\mathbf{X})$, it is not necessary to explicitly maintain the value X_i of each numeric independent variable i.

The third is that numeric independent variables are modeled in terms of an inequality, that is, $x_{i,j} = \mathbb{I}\{X_i \leq a_{i,j}\}$. Alternatively, one could model numeric independent variables by using $x_{i,j}$ to represent whether X_i is between two consecutive split points, e.g.,

$$\begin{aligned}
x_{i,1} &= \mathbb{I}\{X_i \leq a_{i,1}\},\\
x_{i,2} &= \mathbb{I}\{a_{i,1} < X_i \leq a_{i,2}\},\\
x_{i,3} &= \mathbb{I}\{a_{i,2} < X_i \leq a_{i,3}\},\\
&\vdots\\
x_{i,K_i} &= \mathbb{I}\{a_{i,K_i-1} < X_i \leq a_{i,K_i}\},\\
x_{i,K_i+1} &= \mathbb{I}\{a_{i,K_i} > X_i\}.
\end{aligned}$$

One would then re-define the set $\mathbf{C}(s)$ for each split involving the variable i so as to include all of the relevant j values under this new encoding. The advantage of our choice of encoding – using $x_{i,j} = \mathbb{I}\{X_i \leq a_{i,j}\}$ – is that the resulting formulation enhances the power of branching on fractional

values of $x_{i,j}$ and leads to more balanced branch-and-bound trees (Vielma 2015). This type of encoding has been used successfully in scheduling and transportation applications (so-called "by" variables, representing an event happening by some period t; see for example Bertsimas et al. 2011); for further details, the reader is referred to Vielma (2015).

3.3. Theoretical properties

One question that we can ask, having described formulation (2), is whether there exist alternate MIO formulations for problem (1). We now describe one such alternate formulation, which involves relating the $\mathbf{y}$ and $\mathbf{x}$ variables in a different way.

In particular, for any leaf ℓ of any tree t, let $\mathbf{LS}(\ell)$ be the set of splits for which leaf ℓ is on the left side (i.e., s such that $\ell \in \mathbf{left}(s)$), and $\mathbf{RS}(\ell)$ be the set of splits for which leaf ℓ is on the right side (i.e., s such that $\ell \in \mathbf{right}(s)$). The ensemble tree optimization problem can then be formulated as the following problem:

$$\begin{aligned}
\underset{\mathbf{x}}{\text{maximize}} \quad & \sum_{t=1}^{T} \sum_{\ell \in \mathbf{leaves}(t)} \lambda_t \cdot p_{t,\ell} \cdot \prod_{s \in \mathbf{LS}(\ell)} \left(\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right) \\
& \qquad \cdot \prod_{s \in \mathbf{RS}(\ell)} \left(1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right) && \text{(3a)} \\
\text{subject to} \quad & \text{constraints (2e)-(2g).} && \text{(3b)}
\end{aligned}$$

The above problem is a binary polynomial problem. Note that the product term, $\prod_{s \in \mathbf{LS}(\ell)} \left(\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right) \cdot \prod_{s \in \mathbf{RS}(\ell)} \left(1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right)$, is exactly 1 if the observation is mapped to leaf ℓ of tree t, and 0 otherwise. The standard linearization of problem (3) (see Crama 1993) is the following MIO:

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \sum_{t=1}^{T} \sum_{\ell \in \mathbf{leaves}(t)} \lambda_t \cdot p_{t,\ell} \cdot y_{t,\ell} && \text{(4a)} \\
\text{subject to} \quad & y_{t,\ell} \leq \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}, \\
& \qquad \forall t \in \{1,\ldots,T\},\ \ell \in \mathbf{leaves}(t),\ s \in \mathbf{LS}(\ell), && \text{(4b)} \\
& y_{t,\ell} \leq 1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}, \\
& \qquad \forall t \in \{1,\ldots,T\},\ \ell \in \mathbf{leaves}(t),\ s \in \mathbf{RS}(\ell), && \text{(4c)} \\
& y_{t,\ell} \geq \sum_{s \in \mathbf{LS}(\ell)} \left(\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right) + \sum_{s \in \mathbf{RS}(\ell)} \left(1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right) \\
& \qquad - (|\mathbf{LS}(\ell)| + |\mathbf{RS}(\ell)| - 1), \\
& \qquad \forall t \in \{1,\ldots,T\},\ \ell \in \mathbf{leaves}(t), && \text{(4d)} \\
& \text{constraints (2e)-(2h).} && \text{(4e)}
\end{aligned}$$

Let Z^*_{LO} be the optimal value of the LO relaxation of problem (2) and let $Z^*_{LO,StdLin}$ be the optimal value of the LO relaxation of problem (4). The following result relates the two optimal values.

PROPOSITION 2. $Z^*_{LO} \leq Z^*_{LO,StdLin}$.

The proof of Proposition 2 (see Section EC.1.3) follows by showing that the optimal solution of the relaxation of problem (2) is a feasible solution for the relaxation of problem (4) and achieves an objective value of exactly Z^*_{LO}. The significance of Proposition 2 is that it establishes that formulation (2) is a stronger formulation of the tree ensemble optimization problem than formulation (4). This is desirable from a practical perspective, as stronger MIO formulations are generally faster to solve than weaker MIO formulations. We shall see in Section 5.2 that the difference in relaxation bounds can be substantial, and that problem (4) is significantly less tractable than our problem (2).

3.4. Depth d approximation

In this section, we describe a hierarchy of relaxations of problem (2) that are based on approximating each tree in the ensemble up to a particular depth.

The motivation for this hierarchy of relaxations comes from the following observation regarding the size of problem (2). In particular, a key driver of the size of problem (2) is the number of left and right split constraints ((2c) and (2d), respectively); these constraints are enforced for every single split in each tree in the ensemble. For a large number of trees that are deep (and thus have many splits), the resulting number of left and right split constraints will be large. At the same time, it may be reasonable to expect that if we do not represent each tree to its full depth, but instead only represent each tree up to some depth d and only include splits that occur before (and including) depth d, then we might still be obtain a reasonable solution to the original problem (2). In this section, we rigorously define this hierarchy of approximate formulations, and provide theoretical guarantees on how close such approximations are to the original formulation.

Let $\Omega = \{(t,s) \,|\, t \in \{1,\ldots,T\},\ s \in \mathbf{splits}(t)\}$ be the set of tree-split pairs. Let $\bar{\Omega} \subseteq \Omega$ be a subset of all possible tree-split pairs. The $\bar{\Omega}$ tree ensemble problem is defined as problem (2) where constraints (2c) and (2d) are restricted to $\bar{\Omega}$:

$$\underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad \sum_{t=1}^{T} \sum_{\ell \in \mathbf{leaves}(t)} \lambda_t \cdot p_{t,\ell} \cdot y_{t,\ell} \tag{5a}$$

$$\text{subject to} \quad \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}, \quad \forall\ (t,s) \in \bar{\Omega}, \tag{5b}$$

$$\sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}, \quad \forall\ (t,s) \in \bar{\Omega}, \tag{5c}$$

$$\text{constraints (2b), (2e) - (2h).} \tag{5d}$$

Solving problem (5) for a fixed $\bar{\Omega}$ will result in a solution $(\mathbf{x},\mathbf{y})$ that satisfies constraints (2c) and (2d) but only for those (t,s) pairs in $\bar{\Omega}$.

For any $d \in \mathbb{Z}_+$, let $\bar{\Omega}_d$ be the set of all tree-split pairs where the split is at a depth $d' \leq d$ (a depth of 1 corresponds to the split at the root node). Let $Z^*_{MIO,d}$ denote the objective value of problem (5) with $\bar{\Omega}_d$, i.e., all splits up to and including depth d, and let $d_{\max}$ be the maximum depth of any tree in the ensemble. The following simple result (see Section EC.1.4 for the proof) establishes that with increasing d, the objective value $Z^*_{MIO,d}$ is an increasingly tighter upper bound on Z^*_{MIO}, the objective value of problem (2) (where the split constraints are up to the full depth of each tree).

PROPOSITION 3. $Z^*_{MIO,1} \geq Z^*_{MIO,2} \geq \cdots \geq Z^*_{MIO,d_{\max}} = Z^*_{MIO}$.

We now show how to construct a complementary lower bound. Let $\mathbf{splits}(t,d)$ denote the set of splits at depth d; if the depth of the tree is strictly less than d, $\mathbf{splits}(t,d)$ is empty. Let us define the constant $\delta_{t,s}$ for each split $s \in \mathbf{splits}(t,d)$ of each tree t as

$$\delta_{t,s} = \max\left\{ \max_{\ell \in \mathbf{left}(s)} p_{t,\ell} - \min_{\ell \in \mathbf{left}(s)} p_{t,\ell},\ \max_{\ell \in \mathbf{right}(s)} p_{t,\ell} - \min_{\ell \in \mathbf{right}(s)} p_{t,\ell} \right\}. \tag{6}$$

The constant $\delta_{t,s}$ is an upper bound on the maximum error possible (due to the depth d truncation of the split constraints) in the prediction of tree t for the observation encoded by $\mathbf{x}$, given that the observation reaches split s. We define Δ_t as the maximum of the $\delta_{t,s}$ values over all the depth d splits of tree t:

$$\Delta_t = \max_{s \in \mathbf{splits}(t,d)} \delta_{t,s}.$$

(In the case that $\mathbf{splits}(t,d)$ is empty, we set $\Delta_t = 0$.)

Before stating our approximation guarantee, we note that given a solution $(\mathbf{x}, \mathbf{y})$ that solves problem (5) with $\bar{\Omega}_d$, it is possible to find a solution $(\mathbf{x}, \tilde{\mathbf{y}})$ that is a feasible solution for the full depth problem (2). This is a consequence of a result which we will state later (Proposition 4). Our approximation guarantee is given below.

THEOREM 1. *Suppose that $\lambda_t \geq 0$ for all $t \in \{1, \ldots, T\}$ and $d \in \mathbb{Z}_+$. Let $(\mathbf{x}, \mathbf{y})$ be the optimal solution of problem (5) with $\bar{\Omega}_d$. Let Z_d be the true objective of $\mathbf{x}$ when embedded within the full-depth problem (5). We then have*

$$Z^*_{MIO,d} - \sum_{t=1}^{T} \lambda_t \Delta_t \leq Z_d \leq Z^*_{MIO} \leq Z^*_{MIO,d}.$$

The above theorem, which we prove in Section EC.1.5, provides a guarantee on how suboptimal the $\bar{\mathbf{x}}$ solution, derived from the depth d problem (5) with $\bar{\Omega}_d$, is for the true (full depth) problem (2). Note that the requirement of λ_t being nonnegative is not particularly restrictive, as we can always make λ_t of a given tree t positive by negating the leaf predictions $p_{t,\ell}$ of that tree. This result is of practical relevance because it allows the decision maker to judiciously trade-off the complexity of the problem (represented by the depth d) against an a priori guarantee on the quality of the approximation. Moreover, the quantity $\sum_{t=1}^{T} \lambda_t \Delta_t$, which bounds the difference between $Z^*_{MIO,d}$ and Z_d, can be easily computed from each tree, allowing the bound to be readily implemented in practice. We shall see in Section 5.3 that although the lower bound can be rather conservative for small values of d, the true objective value of $\bar{\mathbf{x}}$ is often significantly better.

4. Solution methods

The optimization model that we presented in Section 3.2, although tractable for small to medium instances, can be difficult to solve directly for large instances. In this section, we present two solution approaches for tackling large-scale instances of problem (2). In Section 4.1, we present an approach based on Benders decomposition. In Section 4.2, we present an alternate approach based on iteratively generating the split constraints.

4.1. Benders decomposition

The first solution approach that we will consider is Benders decomposition. Recall that in problem (2), we have two sets of variables, $\mathbf{x}$ and $\mathbf{y}$; furthermore, $\mathbf{y}$ can be further partitioned as $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T)$, where $\mathbf{y}_t$ is the collection of $y_{t,\ell}$ variables corresponding to tree t. For any two trees t, t' with $t \neq t'$, notice that the variables $\mathbf{y}_t$ and $\mathbf{y}_{t'}$ do not appear together in any constraints; they are only linked together through the $\mathbf{x}$ variables.

The above observation suggests a Benders reformulation of problem (2). Let us re-write problem (2) as follows:

$$\underset{\mathbf{x}}{\text{maximize}} \quad \sum_{t=1}^{T} \lambda_t G_t(\mathbf{x}), \tag{7a}$$

$$\text{subject to} \quad \text{constraints (2e) - (2g)}, \tag{7b}$$

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

$$
\begin{aligned}
G_t(\mathbf{x}) = \quad & \underset{\mathbf{y}_t}{\text{maximize}} && \sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} \cdot y_{t,\ell} && \text{(8a)}\\
& \text{subject to} && \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1, && \text{(8b)}\\
& && \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}, \quad \forall\ s \in \mathbf{splits}(t), && \text{(8c)}\\
& && \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}, \quad \forall\ s \in \mathbf{splits}(t), && \text{(8d)}\\
& && y_{t,\ell} \geq 0, \quad \forall\ \ell \in \mathbf{leaves}(t). && \text{(8e)}
\end{aligned}
$$

The first result we will prove is of the form of the optimal solution to problem (8). To do this, we first provide a procedure for determining the leaf of the solution encoded by $\mathbf{x}$ for tree t, as Algorithm 1. For ease of exposition, we will denote this procedure applied to a particular observation encoded by $\mathbf{x}$ and a given tree t as GetLeaf$(\mathbf{x}, t)$. We use $\mathbf{leftchild}(\nu)$ to denote the left child of a split node ν, $\mathbf{rightchild}(\nu)$ to denote the right child of a split node ν, and $\mathbf{root}(t)$ to denote the root split node of tree t.

Algorithm 1 Procedure for determining the leaf to which tree t maps $\mathbf{x}$.

```
Initialize ν ← root(t)
while ν ∉ leaves(t) do
  if Σ_{j∈C(ν)} x_{V(s),j} = 1 then
    ν ← leftchild(ν)
  else
    ν ← rightchild(ν)
  end if
end while
return ν
```

Having defined GetLeaf, we now present our first theoretical result (see Section EC.1.6 for the proof).

Proposition 4. *Let* $\mathbf{x} \in \{0,1\}^{\sum_{i=1}^n K_i}$ *be a feasible solution of problem* (7)*. Let* $\ell^* =$ GetLeaf$(\mathbf{x}, t)$ *be the leaf into which* $\mathbf{x}$ *falls, and let* $\mathbf{y}_t$ *be the solution to problem* (8) *defined as*

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

The solution $\mathbf{y}_t$ *is the only feasible solution and therefore, the optimal solution, of problem* (8)*.*

Since problem (8) is feasible and has a finite optimal value, then by LO strong duality the optimal value of problem (8) is equal to the optimal value of its dual. The dual of subproblem (8) is

$$
\begin{aligned}
\underset{\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t}{\text{minimize}} \quad & \sum_{s \in \mathbf{splits}(t)} \alpha_{t,s} \left[\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right] \\
& + \sum_{s \in \mathbf{splits}(t)} \beta_{t,s} \left[1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right] + \gamma_t && \text{(9a)}
\end{aligned}
$$

$$\text{subject to} \quad \sum_{s:\ell \in \mathbf{left}(s)} \alpha_{t,s} + \sum_{s:\ell \in \mathbf{right}(s)} \beta_{t,s} + \gamma_t \geq p_{t,\ell}, \quad \forall\ \ell \in \mathbf{leaves}(t), \tag{9b}$$

$$\alpha_{t,s}, \beta_{t,s} \geq 0, \quad \forall\ s \in \mathbf{splits}(t). \tag{9c}$$

Letting $\mathcal{D}_t$ denote the set of dual feasible $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ for subproblem t, we can re-write problem (7) as

$$\underset{\mathbf{x},\boldsymbol{\theta}}{\text{maximize}} \sum_{t=1}^{T} \lambda_t \theta_t \tag{10a}$$

$$\text{subject to} \sum_{s \in \mathbf{splits}(t)} \alpha_{t,s} \left[\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right] + \sum_{s \in \mathbf{splits}(t)} \beta_{t,s} \left[1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right] + \gamma_t \geq \theta_t,$$

$$\forall\ (\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t) \in \mathcal{D}_t, \tag{10b}$$

$$\text{constraints (2e) - (2g).} \tag{10c}$$

We can now solve problem (10) using constraint generation. In particular, we start with constraint (10b) enforced at a subset of dual solutions $\bar{\mathcal{D}}_t \subseteq \mathcal{D}_t$, and solve problem (10). We then solve problem (9) for each tree t to determine if there exists a solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t) \in \mathcal{D}_t$ for which constraint (10b) is violated. If so, we add the constraint and solve the problem again. Otherwise, if no such $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is found for any tree, then the current $\mathbf{x}$ solution is optimal.

In the above constraint generation scheme, the key step is to find a dual subproblem solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ that is violated. With this motivation, we now prove a complementary result to Proposition 4 on the structure of an optimal solution to the dual problem (9) (see Section EC.1.7 for the proof).

PROPOSITION 5. *Let $\mathbf{x} \in \{0,1\}^{\sum_{i=1}^{n} K_i}$ be a feasible solution of problem (7) and $\mathbf{y}_t$ be the optimal solution of primal subproblem (8). Let $\ell^* = \textsc{GetLeaf}(\mathbf{x}, t)$. An optimal solution of dual subproblem (9) is then given as follows:*

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

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

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

The value of Proposition 5 is that we can check for violated constraints in problem (10) through a simple calculation, without invoking an LO solver. In our numerical experiments, our Benders solution method will consist of solving problem (10) by adding the constraints (10b) using lazy constraint generation. In this approach, we solve problem (10) while maintaining a single branch-and-bound tree. At each node, we check the integer solution by solving problem (9) using Proposition 5 for each tree t and determine if any constraints are violated; if so, we add those constraints to that node only.

4.2. Split constraint generation

Recall from Section 3.4 that when there are a large number of trees and each tree is deep, the total number of splits will be large, and the number of left and right split constraints will be large. However, for a given encoding $\mathbf{x}$, observe that we do not need all of the left and right split

constraints in order for $\mathbf{y}$ to be completely determined by $\mathbf{x}$. As an example, suppose for a tree t that s is the root split, and $\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} = 1$ (i.e., we take the left branch of the root split); in this case, the right split constraint (2d) will force all $y_{t,\ell}$ to zero for $\ell \in \mathbf{right}(s)$. It is clear that in this case, it is not necessary to include any left or right split constraint for any split node s' that is to the right of split s, because all of the $y_{t,\ell}$ values that could be affected by those constraints are already fixed to zero.

This suggests an alternate avenue to solving problem (2), based on iteratively generating the left and right split constraints. Rather than attempting to solve the full problem (2) with all of the left and right split constraints included in the model, start with a subset of left split constraints and a subset of right split constraints and solve the corresponding restricted version of problem (2). For the resulting solution $(\mathbf{x}, \mathbf{y})$, determine whether there exist any tree split pairs (t, s) for which the left split constraint (2c) or right split constraint (2d) are violated. If a violated left or right split constraint is found, add the corresponding left or right constraint to the formulation and solve it again. Repeat the procedure until no violated constraints are found, at which point we terminate with the current solution as the optimal solution.

The key question in such a proposal is: how do we efficiently determine violated constraints? The answer to this question comes from the following proposition.

PROPOSITION 6. *Let* $(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{\sum_{i=1}^{n} K_i} \times \mathbb{R}^{\sum_{t=1}^{T} |\mathbf{leaves}(t)|}$ *be a candidate solution to problem* (2) *that satisfies constraints* (2b) *and constraints* (2e) *to* (2h). *Let* $t \in \{1, \ldots, T\}$. *The solution* $(\mathbf{x}, \mathbf{y})$ *satisfies constraints* (2c) *and* (2d) *for all* $s \in \mathbf{splits}(t)$ *if and only if it satisfies constraint* (2c) *for* $s \in \mathbf{RS}(\ell^*)$ *and constraint* (2d) *for* $s \in \mathbf{LS}(\ell^*)$, *where* $\ell^* = \textsc{GetLeaf}(\mathbf{x}, t)$.

Proposition 6 (see Section EC.1.8 for the proof) states that, to check whether solution $(\mathbf{x}, \mathbf{y})$ satisfies the split constraints for tree t, it is only necessary to check the split constraints for those splits that are traversed when the observation encoded by $\mathbf{x}$ is mapped to a leaf by the action of GetLeaf. This is a simple but extremely useful result, because it implies that we can check for violated constraints simply by traversing the tree, in the same way that we do when we determine the leaf of $\mathbf{x}$.

Algorithm 2 provides the pseudocode of this procedure. This algorithm involves taking the observation encoded by $\mathbf{x}$ and walking it down tree t, following the splits along the way. For each split we encounter, we determine whether we should proceed to the left child ($\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} = 1$) or to the right child ($\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} = 0$). If we are going to the left ($s \in \mathbf{LS}(\ell^*)$ or equivalently, $\ell^* \in \mathbf{left}(s)$), then we check that $y_{t,\ell}$ is zero for all the leaves to the right of split s (constraint (2d)). If we are going to the right ($s \in \mathbf{RS}(\ell^*)$ or equivalently, $\ell^* \in \mathbf{right}(s)$), then we check that $y_{t,\ell}$ is zero for all the leaves to the left of split s (constraint (2c)). In words, we are traversing the tree as we would to make a prediction, and we are simply checking that there is no positive $y_{t,\ell}$ that is on the "wrong" side of any left or right split that we take. If we reach a leaf node, we can conclude that the current solution $(\mathbf{x}, \mathbf{y})$ does not violate any of the split constraints of tree t.

The above procedure can be used as part of a classical constraint generation scheme. Starting with some set of tree-split pairs $\bar{\Omega}$, we can solve problem (5) to obtain the solution $(\mathbf{x}, \mathbf{y})$ and check for violated constraints using Algorithm 2 for each tree t. If we find any tree-split pairs for which a split constraint is violated, we add them to $\bar{\Omega}$, re-solve problem (5) and repeat the process. If no violated tree-split pairs were found, we terminate with $(\mathbf{x}, \mathbf{y})$ as the optimal solution of problem (2).

Alternatively, we can also generate constraints using Algorithm 2 as part of a lazy constraint generation scheme, analogously to our Benders approach. In this approach, we check the integer solution of each node in the branch-and-bound tree using Algorithm 2 and determine if any split constraints are violated; if so, we add those constraints to that node only. We use this constraint generation approach in our numerical experiments in Section 5.4.

Algorithm 2 Procedure for verifying feasibility of candidate solution $(\mathbf{x}, \mathbf{y})$.

Require: Candidate solution $(\mathbf{x}, \mathbf{y})$, satisfying constraint (2b), (2e) - (2h)
  Initialize $\nu \leftarrow \textbf{root}(t)$
  while $\nu \notin \textbf{leaves}(t)$ **do**
    if $\sum_{j \in \mathbf{C}(\nu)} x_{\mathbf{V}(\nu),j} = 1$ **then**
      if $\sum_{\ell \in \textbf{right}(\nu)} y_{t,\ell} > 1 - \sum_{j \in \mathbf{C}(\nu)} x_{\mathbf{V}(\nu),j}$ **then**
        return Violated right constraint (2d) at split ν
      else
        Set $\nu \leftarrow \textbf{leftchild}(\nu)$
      end if
    else
      if $\sum_{\ell \in \textbf{left}(\nu)} y_{t,\ell} > \sum_{j \in \mathbf{C}(\nu)} x_{\mathbf{V}(\nu),j}$ **then**
        return Violated left constraint (2c) at split ν
      else
        Set $\nu \leftarrow \textbf{rightchild}(\nu)$
      end if
    end if
  end while

Data set	Source	Num. Vars.	Num. Obs.	Description
`winequalityred` *	Cortez et al. (2009)	11	1599	Predict quality of (red) wine
`concrete` **	Yeh (1998)	8	1030	Predict strength of concrete
`permeability` **	Kansy et al. (1998)	1069	165	Predict permeability of compound
`solubility` **	Tetko et al. (2001), Huuskonen (2000)	228	951	Predict solubility of compound

Table 1 Summary of real data sets used in numerical experiments. Note: * = accessed via UCI Machine Learning Repository (Lichman 2013); ** = accessed via `AppliedPredictiveModeling` package in R (Kuhn and Johnson 2014).

5. Computational experiments

In this section, we describe our first set of computational results. Section 5.1 provides the background of our experiments. Section 5.2 provides initial results on the full MIO formulation (2), while Section 5.3 provides results on the depth approximation scheme of Section 3.4. Finally, Section 5.4 compares the Benders and split generation solution methods against directly solving problem (2).

5.1. Background

We test our optimization formulation (2) and the associated solution methods from Section 4 using tree ensemble models estimated from real data sets. We consider several real data sets; the details are provided in Table 1. We wish to point out that in these data sets, the independent variables may in reality not be controllable. However, despite this, the data sets are still useful in that they will furnish us with real tree ensemble models for evaluating our optimization methodology.

We specifically focus on random forest models. Unless otherwise stated, all random forest models are estimated in R using the `randomForest` package (Liaw and Wiener 2002), using the default parameters. All linear and mixed-integer optimization models are formulated in the Julia programming language (Bezanson et al. 2012), using the `JuMP` package (Julia for Mathematical Programming; see Lubin and Dunning 2015), and solved using Gurobi 6.5 (Gurobi Optimization, Inc. 2015).

All experiments were executed on a late 2013 Apple Macbook Pro Retina laptop, with a quad-core 2.6GHz Intel i7 processor and 16GB of memory.

5.2. Full MIO formulation experiments

As part of our first experiment, we consider solving the unconstrained tree ensemble problem for each data set. For each data set, we consider optimizing the default random forest model estimated in R which uses 500 trees (the parameter `ntree` in `randomForest` is set to 500). For each data set, we also consider solving the tree ensemble problem using only the first T trees of the complete forest, where T ranges in $\{5, 10, 50, 100, 200\}$. For each data set and each value of T, we solve the MIO formulation (2), as well as its LO relaxation.

We compare our MIO formulation against two other approaches:

1. **Local search**: We solve the tree ensemble problem (1) using a local search heuristic. The details of this local search are provided in Section EC.2; at a high level, it starts from a randomly chosen initial solution, and iteratively improves the solution, one independent variable at a time, until a local optimum is reached. The heuristic is repeated from ten starting points, out of which we only retain the best (highest objective value) solution. We test such an approach to establish the value of our MIO-based approach, which obtains a globally optimal solution, as opposed to a locally optimal solution.
2. **Standard linearization MIO**: We solve the standard linearization MIO (4) and its relaxation, in order to obtain a relative sense of the strength of formulation (2). Due to this formulation being much harder to solve, we impose a 30 minute time limit on the solution time of the integer formulation.

We consider several metrics:

- $\mathcal{T}_{MIO}$: the time (in seconds) to solve the MIO (2).
- $\mathcal{T}_{StdLin,MIO}$: the time (in seconds) to solve the standard linearization MIO (4).
- $\mathcal{T}_{LS}$: the time (in seconds) to run the local search procedure (value reported is the total for ten starting points).
- G_{LS}: the gap of the local search solution; if Z_{LS} is the objective value of the local search solution and Z^* is the optimal objective value of problem (2), then

$$G_{LS} = 100\% \times (Z^* - Z_{LS})/Z^*.$$

- G_{LO}: the gap of the LO relaxation of problem (2); if Z_{LO} is the objective value of the LO relaxation and Z^* is the optimal integer objective as before, then

$$G_{LO} = 100\% \times (Z_{LO} - Z^*)/Z^*.$$

- $G_{StdLin,LO}$: the gap of the LO relaxation of the standard linearization MIO (4); if $Z_{StdLin,LO}$ is the optimal value of the relaxation, then

$$G_{StdLin,LO} = 100\% \times (Z_{StdLin,LO} - Z^*)/Z^*.$$

- $G_{StdLin,MIO}$: the optimality gap of the standard linearization MIO (4); if $Z_{StdLin,UB}$ and $Z_{StdLin,LB}$ are the best upper and lower bounds, respectively, of problem (4) upon termination, then

$$G_{StdLin,MIO} = 100\% \times (Z_{StdLin,UB} - Z_{StdLin,LB})/Z_{StdLin,UB}.$$

- N_{Levels}: the number of levels (i.e., dimension of $\mathbf{x}$), defined as $N_{Levels} = \sum_{i=1}^{n} K_i$.
- N_{Leaves}: the number of leaves (i.e., dimension of $\mathbf{y}$), defined as $N_{Leaves} = \sum_{t=1}^{T} |\mathbf{leaves}(t)|$.

Table 2 shows the solution time and problem size metrics, while Table 3 shows the gap metrics. From these two tables, we can draw several conclusions. First, the time required to solve problem (2) is very reasonable; in the most extreme case (`winequalityred`, $T = 500$), problem (2) can be solved to full optimality in about 20 minutes. (Note that no time limit was imposed on problem (2); all

Data set	T	N_{Levels}	N_{Leaves}	$\mathcal{T}_{MIO}$	$\mathcal{T}_{StdLin,MIO}$	$\mathcal{T}_{LS}$
`solubility`	10	1253	3157	0.1	215.2	0.2
	50	2844	15933	0.8	1800.3	1.8
	100	4129	31720	1.7	1801.8	8.8
	200	6016	63704	4.5	1877.8	33.7
	500	9646	159639	177.9	1800.3	147.0
`permeability`	10	2138	604	0.0	122.6	1.0
	50	2138	3056	0.1	1800.0	1.9
	100	2138	6108	0.2	1800.3	3.1
	200	2138	12214	0.5	1800.0	6.1
	500	2138	30443	2.7	1800.0	19.1
`winequalityred`	10	1370	3246	1.8	1800.1	0.0
	50	2490	16296	18.5	1800.1	0.6
	100	3000	32659	51.6	1800.1	2.5
	200	3495	65199	216.0	1800.2	11.4
	500	3981	162936	1159.7	1971.8	34.6
`concrete`	10	1924	2843	0.2	1800.8	0.1
	50	5614	14547	22.7	1800.1	1.3
	100	7851	29120	67.8	1800.1	4.3
	200	10459	58242	183.8	1800.2	20.2
	500	13988	145262	846.9	1809.4	81.6

Table 2 Solution times for tree ensemble optimization experiment.

values of $\mathcal{T}_{MIO}$ correspond to the time required to solve problem (2) to full optimality.) In contrast, the standard linearization problem (4) was only solved to full optimality in two out of twenty cases within the 30 minute time limit. In addition, for those instances where the solver reached the time limit, the optimality gap of the final integer solution, $G_{StdLin,MIO}$, was quite poor, ranging from 50 to over 100%.

Second, the integrality gap G_{LO} is quite small – on the order of a few percent in most cases. This suggests that the LO relaxation of problem (2) is quite tight. In contrast, the LO relaxation bound from the standard linearization problem (4) is weaker than that of problem (2), as predicted by Proposition 2, and strikingly so. The weakness of the relaxation explains why the corresponding integer problem cannot be solved to a high degree of optimality within the 30 minute time limit. These results, together with the results above on the MIO solution times and the final optimality gaps of problem (4), highlight the edge of our formulation (2) over the standard linearization formulation (4).

Third, although there are many cases where the local search solution performs quite well, there are many where it can be quite suboptimal, even when repeated with ten starting points. Moreover, while the local search time T_{LS} is generally smaller than the MIO time T_{MIO}, in some cases it is not substantially lower (for example, `solubility` for $T = 500$). The modest additional time required by the MIO formulation (2) may therefore be justified for the guarantee of provable optimality.

5.3. Depth d approximation experiments

In this section, we investigate the use of the depth d tree problem (formulation (5) with $\bar{\Omega}_d$) for approximating the full depth problem (2).

In this set of experiments, we focus on the same data sets as before with $T = 100$. We solve problem (5) with $\bar{\Omega}_d$ and vary the depth d of the approximation. We consider the upper bound $Z^*_{MIO,d}$ (denoted by "UB"), the actual value of the solution Z_d (denoted by "Actual") and the lower bound $Z^*_{MIO,d} - \sum_{t=1}^{T} \lambda_t \Delta_t$ (denoted by "LB").

Data set	T	G_{LO}	$G_{StdLin,LO}$	$G_{StdLin,MIO}$	G_{LS}
`solubility`	10	0.0	485.5	0.0	18.6
	50	0.0	498.0	50.1	9.5
	100	0.0	481.2	70.5	0.3
	200	0.0	477.5	77.7	0.2
	500	0.0	501.3	103.2	0.2
`permeability`	10	0.0	589.5	0.0	6.1
	50	0.0	619.4	71.9	3.5
	100	0.0	614.1	75.0	1.8
	200	0.0	613.0	80.0	0.1
	500	0.0	610.4	85.9	0.0
`winequalityred`	10	1.5	11581.3	89.8	1.2
	50	3.4	11873.6	98.3	2.3
	100	4.3	12014.9	98.8	0.6
	200	4.3	12000.6	99.0	1.2
	500	4.5	12031.8	99.2	1.4
`concrete`	10	0.0	6210.6	72.5	0.0
	50	1.8	6657.1	95.0	0.0
	100	2.6	6706.6	98.3	0.0
	200	1.6	6622.2	98.5	0.0
	500	2.2	6652.6	98.8	0.0

Table 3 **Gaps for tree ensemble optimization experiment.**

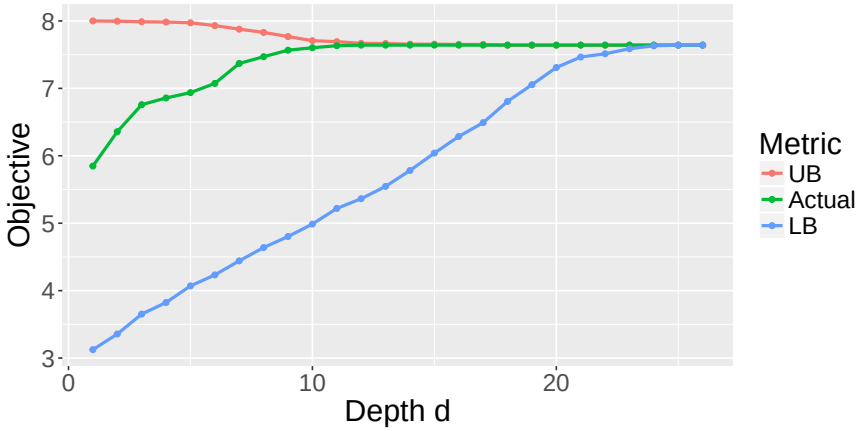

Figure 2 **Plot of UB, Actual and LB versus depth for `winequalityred` with $T = 100$.**

Figures 2 and 3 plot the above three metrics for the `winequalityred` and `concrete` data sets, respectively. From these plots, we can see that the upper bound is decreasing, while the lower bound and the actual objective are increasing. We can also see that the lower bound is quite loose, and the depth d needs to increase significantly in order for the lower bound to be close to the upper bound. However, even when the depth d is small and the lower bound is loose, the actual objective of the solution produced by the approximation is very good. In the case of `winequalityred`, the solution is essentially optimal after a depth of $d = 15$ (compared to a maximum depth of 26); for `concrete`, this occurs for a depth of $d = 9$ (compared to a maximum depth of 24).

To complement these results on the objective values of the formulations, Figures 4 and 5 show the computation time of the depth approximation formulation as d varies for the same two data

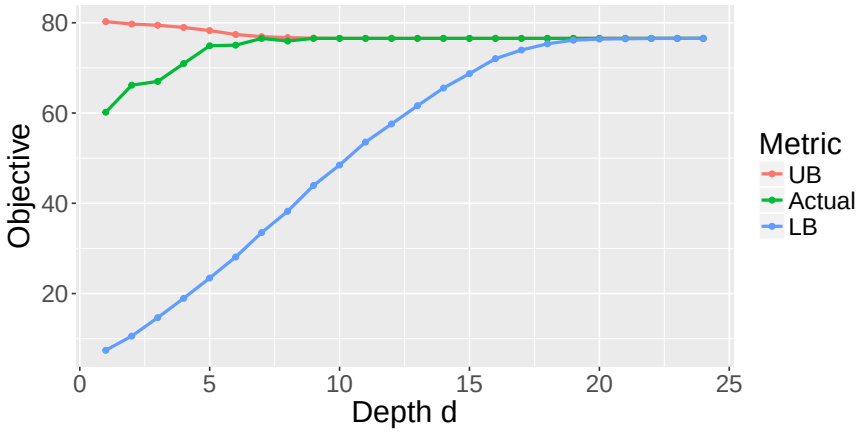

Figure 3 **Plot of UB, Actual and LB versus depth for `concrete` with $T = 100$.**

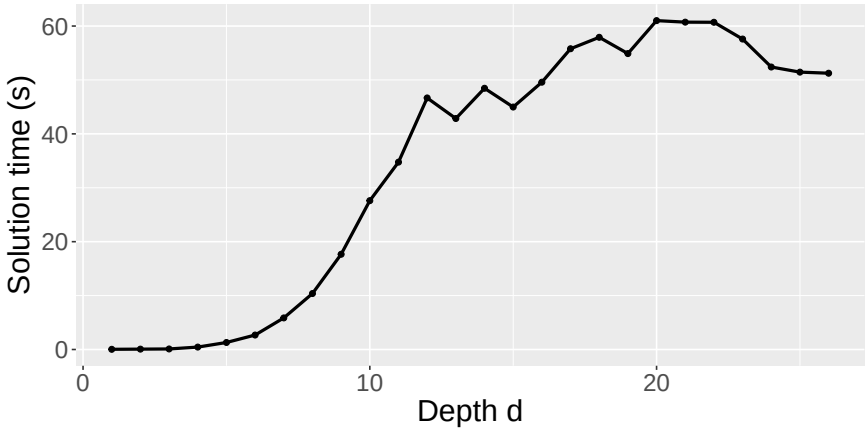

Figure 4 **Plot of solution time for problem (5) with $\bar{\Omega}_d$ versus depth d for `winequalityred` with $T = 100$.**

sets. Here we can see that the solution time required to solve the depth approximation formulation initially increases in an exponential fashion as d increases; this is to be expected, because with each additional layer of splits, the number of splits roughly doubles. Interestingly, though, the solution time seems to plateau after a certain depth, and no longer continues to increase. Together with Figures 2 and 3, these plots suggest the potential of the depth approximation approach to obtain near-optimal and optimal solutions with significantly reduced computation time relative to the full depth problem.

5.4. Solution method experiments

In this final set of experiments, we evaluate the effectiveness of the two solution methods from Section 4 – Benders decomposition and split constraint generation – on solving large instances of problem (2). We consider the same data sets as before with $T = 500$. For each instance, we consider: $\mathcal{T}_{Benders}$, $\mathcal{T}_{SplitGen}$ and $\mathcal{T}_{MIO}$, which are the times to solve problem (2) to full optimality using the Benders approach, using the split constraint generation approach and by directly solving problem (2), respectively.

Table 4 shows the results from this comparison. From this table, we can see that both approaches can lead to dramatic reductions in the solution time relative to solving problem (2) directly with all split constraints enforced at the start. In the most extreme case (`concrete`), we observe a reduction from about 800 seconds for the standard solution method to about 32 seconds for split constraint

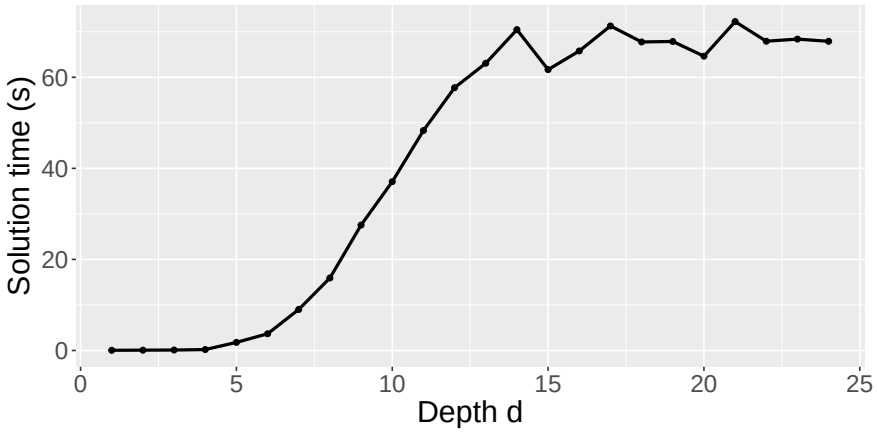

Figure 5 **Plot of solution time for problem** (5) **with** $\bar{\Omega}_d$ **versus depth** d **for** `concrete` **with** $T = 100$.

Data set	T	N_{Levels}	N_{Leaves}	$\mathcal{T}_{MIO}$	$\mathcal{T}_{Benders}$	$\mathcal{T}_{SplitGen}$
`solubility`	100	4129	31720	1.7	1.1	0.8
	200	6016	63704	4.5	2.6	0.8
	500	9646	159639	177.9	28.6	13.7
`permeability`	100	2138	6108	0.2	0.0	0.0
	200	2138	12214	0.5	0.2	0.1
	500	2138	30443	2.7	0.4	0.7
`winequalityred`	100	3000	32659	51.6	41.1	56.5
	200	3495	65199	216.0	152.3	57.2
	500	3981	162936	1159.7	641.8	787.8
`concrete`	100	7851	29120	67.8	8.8	8.6
	200	10459	58242	183.8	12.5	15.4
	500	13988	145262	846.9	37.5	32.3

Table 4 **Results of experiments comparing solution methods.**

generation and about 37 seconds for the Benders approach – a reduction in solution time of over 95%. In some cases, Benders decomposition is slightly faster than split generation; for example, for `winequalityred` with $T = 500$, the Benders approach requires just under 11 minutes whereas split generation requires just over 13 minutes. In other cases, split generation is faster (for example, `solubility` with $T = 500$).

6. Case study 1: drug design

In this section, we describe our case study in drug design. Section 6.1 provides the background on the problem and the data. Section 6.2 reports the unconstrained optimization results of the unconstrained optimization problem, while Section 6.3 shows the results of our optimization model when the similarity to existing compounds is constrained.

6.1. Background

For this set of experiments, we use the data sets from Ma et al. (2015). These data sets were created for a competition sponsored by Merck and hosted by Kaggle. There are fifteen different data sets. In each data set, each observation corresponds to a compound/molecule. Each data set has a single dependent variable, which is different in each data set and represents some measure of "activity" (a property of the molecule or the performance of the molecule for some task). The independent

Data set	Name	Num Obs.	Num. Variables
1	3A4	37241	9491
2	CB1	8716	5877
3	DPP4	6148	5203
4	HIVINT	1815	4306
5	HIVPROT	3212	6274
6	LOGD	37388	8921
7	METAB	1569	4505
8	NK1	9965	5803
9	OX1	5351	4730
10	OX2	11151	5790
11	PGP	6399	5135
12	PPB	8651	5470
13	RAT_F	6105	5698
14	TDI	4165	5945
15	THROMBIN	5059	5552

Table 5 Summary of drug design data sets (see Ma et al. 2015 for further details).

variables in each data set are the so-called "atom pair" and "donor-acceptor pair" features, which describe the substructure of each molecule (see Ma et al. 2015 for further details). The goal of the competition was to develop a model to predict activity using the molecular substructure; such models are known as quantitative structure-activity relationship (QSAR) models.

The optimization problem that we will consider is to find the molecule that maximizes activity as predicted by a random forest model. Our interest in this problem is two-fold. First, this is a problem of significant practical interest, as new drugs are extremely costly to develop. Moreover, these costs are rising: the number of drugs approved by the FDA per billion dollars of pharmaceutical R&D spending has been decreasing by about 50% every 10 years since 1950 (a phenomenon known as "Eroom's Law" – Moore's Law backwards; see Scannell et al. 2012). As a result, there has been growing interest in using analytics to identify promising drug candidates in academia as well as industry (see for example Atomwise Inc. 2017). With regard to random forests, we note that they are widely used in this domain: the QSAR community was one of the first to adopt them (Svetnik et al. 2003) and they have been considered a "gold standard" in QSAR modeling (Ma et al. 2015).

Second, the problem is of a very large scale. The number of independent variables ranges from about 4000 to just under 10,000, while the number of observations ranges from about 1500 to just over 37,000; in terms of file size, the smallest data set is approximately 15MB, while the largest weighs in at just over 700MB. Table 5 summarizes the data sets. Estimating a random forest model using the conventional `randomForest` package in R on any one of these data sets is a daunting task; to give an example, a single random forest tree on the largest data set required more than 5 minutes to estimate, which extrapolates to a total computation time of over 8 hours for a realistic ensemble of 100 trees.

6.2. Unconstrained optimization results

In our first set of experiments, we proceed as follows. For each data set, we estimate a random forest model to predict the activity variable using all available independent variables. To reduce the computational burden posed by estimating random forest models from such large data sets, we deviate from our previous experiments by using the `ranger` package in R (Wright and Ziegler 2017), which is a faster implementation of the random forest algorithm suited for high dimensional data sets. In addition, we follow Ma et al. (2015) in restricting the number of trees to 100. For each such random forest model, we solve the corresponding (unconstrained) tree ensemble optimization problem (2) using the Benders approach of Section 4.1 and the split generation approach of Section 4.2. We impose a time limit of two hours. We also solve each tree ensemble optimization problem using local search with ten repetitions. We consider the following metrics:

Data set	N_{Levels}	N_{Leaves}	$\mathcal{T}_{splitGen}$	$\mathcal{T}_{Benders}$	$\mathcal{T}_{LS}$	$G_{SplitGen}$	$G_{Benders}$	G_{LS}
1	27145	852533	97.3	7200.0	390.8	0.00	0.07	9.26
2	16480	289800	6533.8	7200.0	132.6	0.00	1.30	8.64
3	13697	201265	6252.2	7200.0	84.9	0.00	1.41	11.58
4	11790	59552	2.1	6.3	55.4	0.00	0.00	5.92
5	16426	109378	23.9	7200.0	108.9	0.00	0.07	6.99
6	26962	1307848	7219.8	7200.1	409.3	2.15	12.82	23.71
7	12523	53934	12.3	1743.0	60.8	0.00	0.00	11.17
8	17319	328705	7200.5	7200.1	146.9	0.27	2.67	5.22
9	12595	184841	55.0	7200.1	73.8	0.00	0.46	15.76
10	15780	379583	1339.5	7200.2	124.7	0.00	4.71	12.03
11	15111	217395	81.0	7200.1	94.0	0.00	0.55	12.87
12	15737	291709	40.0	7200.0	94.4	0.00	0.02	12.17
13	17841	212926	6731.5	7200.0	137.3	0.00	3.73	26.38
14	16272	145476	11.5	41.8	110.9	0.00	0.00	17.10
15	14863	169638	223.1	7200.0	111.7	0.00	0.89	13.10

Table 6 Comparison of split generation and Benders decomposition for drug design data sets.

- $G_{SplitGen}, G_{Benders}$: the optimality gap of the solution produced by the split generation and Benders methods, respectively. If $Z_{SplitGen,LB}$ and $Z_{Benders,LB}$ are lower bounds and $Z_{SplitGen,UB}$ and $Z_{Benders,UB}$ are lower bounds, then $G_{SplitGen}$ and $G_{Benders}$ are defined as

$$G_{SplitGen} = 100\% \times (Z_{SplitGen,UB} - Z_{SplitGen,LB})/Z_{SplitGen,UB},$$
$$G_{Benders} = 100\% \times (Z_{Benders,UB} - Z_{Benders,LB})/Z_{Benders,UB}.$$

- G_{LS}: the optimality gap of the local search solution, relative to the best split generation solution. If Z_{LS} is the local search objective, it is defined as

$$G_{LS} = 100\% \times (Z_{SplitGen,LB} - Z_{LS})/Z_{SplitGen,LB}.$$

- $\mathcal{T}_{SplitGen}, \mathcal{T}_{Benders}$: the time (in seconds) to solve problem (2) using the split generation and Benders methods, respectively. (A time that is below 7200 indicates that the problem was solved to full optimality.)
- $\mathcal{T}_{LS}$: the time (in seconds) to execute the local search procedure. (the time reported is the total of ten repetitions).
- N_{Levels} and N_{Leaves}: the number of levels and the number of leaves in the ensemble, respectively, defined as in Section 5.2.

Table 6 displays the results of this experiment. We first discuss the split generation results. With regard to the split generation approach, we can see that out of the fifteen data sets, ten were solved to full optimality within one hour, and another three were solved to full optimality within the next hour. For two data sets (6 and 8), the solver terminated after two hours with suboptimal solutions with a very low optimality gap (2.2% and 0.27% for sets 6 and 8, respectively). The high optimality gap for set 6 is to be expected, as this data set is among the two largest data sets in terms of the number of levels and the total number of leaves (which stems from the number of variables and the number of observations in that data set; see Table 5).

For the Benders approach, we can see that the performance is quite different. The optimality gap of the solution produced by the Benders approach is substantially worse than that achieved by split generation after two hours. The Benders approach is only able to solve two instances to full optimality within the two hour time limit and in both instances, the split generation approach is able to solve the same instance to full optimality more quickly. Overall, on this set of instances,

Data set	π_{avg}	π_{max}
1	0.00006	0.45
2	0.00033	0.37
3	0.00042	0.11
4	0.00157	0.71
5	0.00071	0.40
6	0.00005	0.15
7	0.00115	0.43
8	0.00029	0.42
9	0.00040	0.51
10	0.00027	0.48
11	0.00028	0.36
12	0.00032	0.54
13	0.00030	0.36
14	0.00053	0.41
15	0.00039	0.21

Table 7 Average and maximum proximity of split generation solutions for drug design data sets.

split generation dominates Benders decomposition in both the solution time and the optimality gap after a fixed computation time.

The last important insight from Table 6 concerns the performance of the local search procedure. With regard to solution times, we can see that in some cases the total time required for the ten repetitions of the local search *exceeds* the time required for split generation (see data set 1 for example). In addition, and more importantly, the best solution obtained by local search in each data set is highly suboptimal, as evidenced by the high values of G_{LS}. In the best case (data set 8), G_{LS} is about 5%, whereas in the worst case (data set 13), G_{LS} is as high as 26%. The main message of these results is that heuristic approaches are simply not enough for this problem: our approaches, which are provably optimal and based on mixed-integer optimization, deliver significantly better solutions.

6.3. Controlling proximity

In the random forest literature, one concept that is useful for analyzing random forest models is that of *proximity*. The proximity of two observations $\mathbf{X}, \mathbf{X}' \in \mathcal{X}$ is defined as the proportion of trees for which $\mathbf{X}$ and $\mathbf{X}'$ fall in the same leaf:

$$\pi(\mathbf{X}, \mathbf{X}') = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\{\ell_t(\mathbf{X}) = \ell_t(\mathbf{X}')\},$$

where $\ell_t(\mathbf{X})$ is the leaf to which tree t maps the observation $\mathbf{X}$. The proximity measure is useful to consider because it gives a sense of how similar two observations are.

In the context of our drug design case study, we can consider the proximity of the solution to the points in the training set. Table 7 displays, for the split generation solution of each data set, the average proximity, π_{avg}, and maximum proximity, π_{max}, taken over all of the training points in the data set. We can see that the maximum proximity – the highest proximity between the solution and any point in the training data – is in general relatively low. For example, for data set 3, π_{max} is 0.11; this means that the most that the solution and any point in the training data overlap in terms of how the trees in the forest classify them is 0.11 (i.e., at most only eleven trees out of the 100 trees in the forest will place both points in the same leaf). In addition, the average proximity of the solution to the training data is much lower than the maximum proximity, which suggests that for most points in the training data, the actual proximity to the solution is close to or exactly zero (i.e., there is no similarity between the solution and the training point). At the other extreme, the highest maximum proximity observed is for data set 4, for which the maximum proximity is 0.71 – in other words, the training set point closest to the solution is classified identically to the solution for 71 out of 100 trees.

The reason it is important to consider proximity is because, in the application at hand, the random forest model is being used to identify promising *new* molecules to test. As such, the proximity measure can be viewed as a measure of novelty: a solution whose maximum proximity

is 1 or close to 1 is a solution that is similar to one the molecules that already exist, whereas a proximity of zero or close to zero is a solution that is different from the extant molecules. Note that in Table 7, all solutions have a maximum proximity strictly lower than 1, indicating that all fifteen solutions are different from all of the molecules of their respective training data sets.

Motivated by the preceding discussion on proximity, we now present our second set of experiments. In this set of experiments, we solve problem (2) but with an added constraint on the maximum proximity of the solution to the training points. Such a constraint is defined as follows. Let $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(M)}$ be the set of observations in the training data, and define for each observation m the vector $\mathbf{y}^{(m)} \in \mathbb{R}^{\sum_{t=1}^{T} |\mathbf{leaves}(t)|}$ as

$$y_{t,\ell}^{(m)} = \mathbb{I}\{\ell_t(\mathbf{X}^{(m)}) = \ell\}.$$

The proximity between the fixed observation $\mathbf{X}^{(m)}$ and the solution encoded by $(\mathbf{x}, \mathbf{y})$ can then be written as an affine function of $\mathbf{y}$:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}^{(m)} \cdot y_{t,\ell}.$$

(Note that $\mathbf{y}^{(1)}, \ldots \mathbf{y}^{(M)}$ are data, and not decision variables.) We can thus enforce a constraint on the proximity of the solution encoded by $\mathbf{x}$ to each observation $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(M)}$ to be at most $c \in [0, 1]$ through the following family of linear constraints on $\mathbf{y}$:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell}^{(m)} \cdot y_{t,\ell} \leq c, \quad \forall\ m \in \{1, \ldots, M\}. \tag{11}$$

We solve problem (2) with constraint (11) and vary the parameter c to generate a Pareto efficient frontier of solutions that optimally trade-off their maximum proximity to the training data and their predicted value under the random forest model. We impose a time limit of one hour on each solve of problem (2).

To demonstrate, we focus on data sets 4 and 7. Figure 6 shows the proximity-objective value frontier (the points labeled "MIO") for data set 4, while Figure 7 does the same for data set 7. We also note that the objective value is expressed in terms of the maximum unconstrained objective value (i.e., the objective value attained when the proximity constraint is omitted). We note that the right-most point in each frontier (maximum proximities of 0.71 in Figure 6 and 0.43 in Figure 6) corresponds to the original unconstrained solution. As a comparison, we also solve the (unconstrained) problem for each data set using local search with 100 repetitions, and plot the proximity and objective value of each solution from each repetition (the points labeled "LS").

We remark that both of these frontiers are approximate because each point is not necessarily solved to full optimality, due to the one hour time constraint. However, upon closer examination, we found that only six points out of a total of 112 across both plots did not solve to full optimality, and the largest optimality gap among those six was only about 0.12%. Thus, although constraint (11) adds to the problem size, it does not drastically impact our ability to solve problem (2).

From both of these figures, we obtain several insights. First, in these two cases, we are able to push the proximity quite low; for both data sets, we can find solutions with maximum proximities of 0.01 (i.e., one tree out of the ensemble of 100). (For both data sets, we were unable to find a feasible solution with a maximum proximity of zero within the one hour time limit; it is possible that such a solution does not exist.) Second, we can can see that the price of novelty is low: as we decrease the maximum proximity, we can still obtain solutions with very good predicted performance. For example, for data set 4 (Figure 6), we can see that if we lower the proximity to 0.01, the relative objective value decreases by only about 7%. Third, although the solutions obtained by local search have smaller maximum proximities than the unconstrained MIO solution,

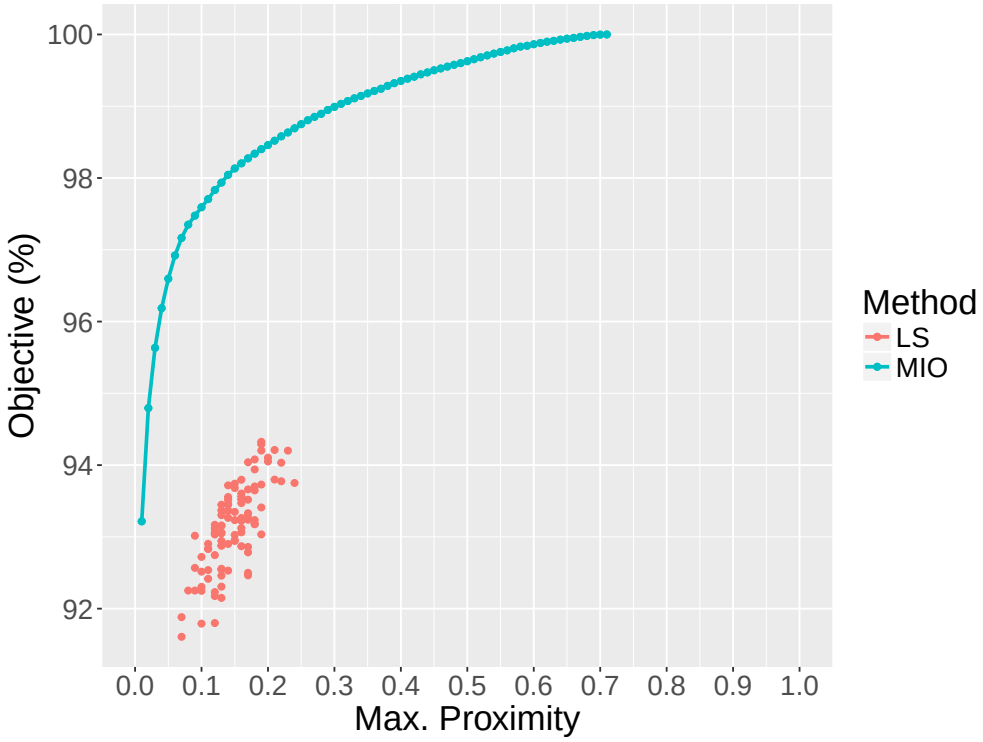

Figure 6 **Plot of objective-maximum proximity frontier for drug design data set 4.**

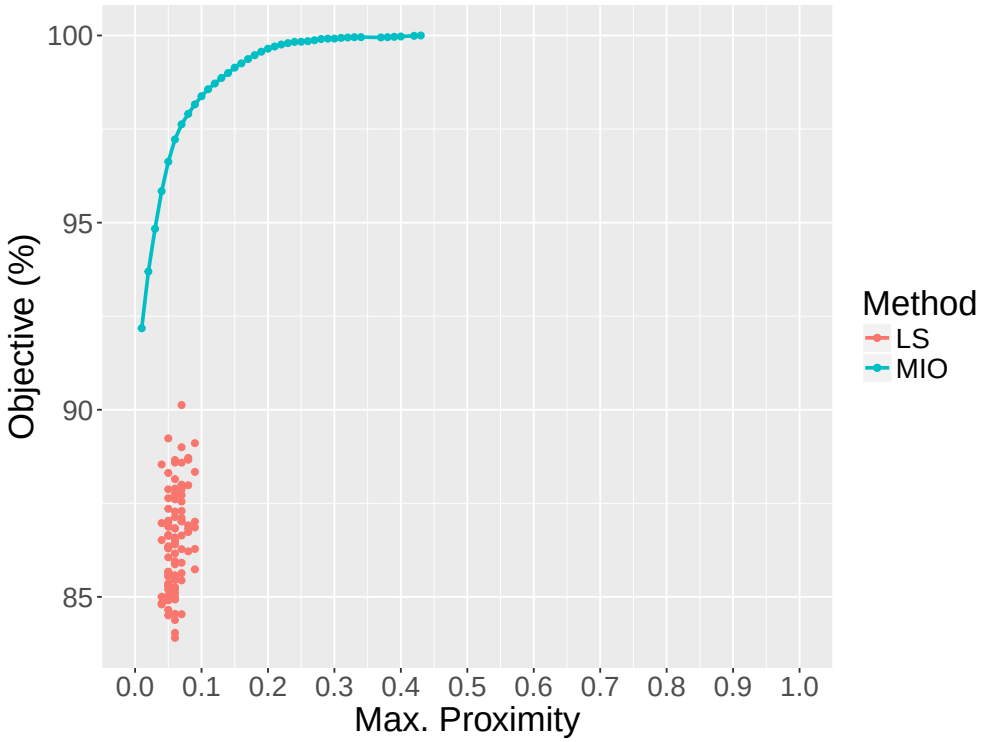

Figure 7 **Plot of objective-maximum proximity frontier for drug design data set 7.**

they are highly suboptimal with respect to objective value (best relative objectives for data sets 4 and 7 are roughly 94% and 90%, respectively) and are dominated in both maximum proximity and objective value by the MIO solutions. Overall, these results suggest that our MIO formulation can be used to systematically identify promising molecules that have good predicted performance and are sufficiently novel as compared to existing molecules in the training data.

7. Case study 2: customized pricing

In this section, we apply our approach to customized pricing. Section 7.1 provides the background on the data, while Section 7.2 describes our random forest model as well as two alternative models based on hierarchical Bayesian regression. Section 7.3 compares the models in terms of out-of-sample predictions of profit. Finally, Section 7.4 formulates the profit optimization problem and compares the three models.

7.1. Background

We consider the data set from Montgomery (1997), which was accessed via the `bayesm` package in R (Rossi 2012). This data set contains price and sales data for eleven different refrigerated orange juice brands for the Dominick's Finer Foods chain of grocery stores in the Chicago area.

In this data set, each observation corresponds to a given store in the chain at a given week. The data span 83 stores and a period of 121 weeks. Each observation consists of: the week t; the store s; the sales $q_{t,s,1},\ldots,q_{t,s,11}$ of the eleven orange juice brands; the prices $p_1,\ldots,p_{11}$ of the eleven orange juice brands; dummy variables $d_{t,s,1},\ldots,d_{t,s,11}$, where $d_{t,s,i}=1$ if orange juice brand i had any in-store displays (such as in-store coupons) at store s in week t through deal; and dummy variables $f_{t,s,1},\ldots,f_{t,s,11}$, where $f_{t,s,i}=1$ if brand i was featured/advertised in store s in week t. We use $\mathbf{p}$, $\mathbf{d}$ and $\mathbf{f}$ to denote vectors of prices, deal dummies and feature dummies, and subscripts to denote the observation; for example, $\mathbf{p}_{t,s}=(p_{t,s,1},\ldots,p_{t,s,11})$ is the vector of brand prices at store s in week t.

The data set also include 11 covariates for the 83 stores corresponding to demographic information of each store's neighborhood, such as the percentage of the population with a college degree and the percentage of households with more than five members; we denote these covariates as $z_{s,1},\ldots,z_{s,11}$. We denote the vector of covariates for store s as $\mathbf{z}_s$; for notational convenience later, we assume that $z_{s,0}=1$ in $\mathbf{z}_s$. For more details, we refer the reader to Montgomery (1997).

7.2. Models

The standard approach for modeling this type of data in marketing is to posit a regression model within a hierarchical Bayes (HB) framework (Rossi et al. 2005). We will consider two different HB specifications. In the first model, which we will refer to as HB-LogLog, we assume that the logarithm of sales of a given product is linear in the logarithm of prices of all eleven products, and linear in the deal and feature dummies. For a fixed focal brand i, the regression function is given below:

$$\log(q_{t,s,i})=\beta_{s,0}+\boldsymbol{\beta}_{s,i,\mathbf{p}}^T\log(\mathbf{p}_{t,s})+\boldsymbol{\beta}_{s,i,\mathbf{d}}^T\mathbf{d}_{t,s}+\boldsymbol{\beta}_{s,i,\mathbf{f}}^T\mathbf{f}_{t,s}+\epsilon_{t,s,i}, \tag{12}$$

where $\epsilon_{t,s,i}$ follows a univariate normal distribution with mean 0 and variance τ_s:

$$\epsilon_{t,s,i}\sim N(0,\tau_{s,i}), \tag{13}$$

and the vector of regression coefficients $\boldsymbol{\beta}_{s,i}=(\beta_{s,i,0},\boldsymbol{\beta}_{s,i,\mathbf{p}},\boldsymbol{\beta}_{s,i,\mathbf{d}},\boldsymbol{\beta}_{s,i,\mathbf{f}})$ follows a multivariate normal distribution with mean $\Delta_i^T\mathbf{z}_s$ and covariance matrix $V_{\boldsymbol{\beta},i}$:

$$\boldsymbol{\beta}_{s,i}\sim N(\Delta_i^T\mathbf{z}_s,V_{\boldsymbol{\beta},i}), \tag{14}$$

where Δ_i is a 12-by-34 matrix. (Each row corresponds to one of the store level covariates, and each column corresponds to one of the β coefficients.) This model assumes that the regression coefficients

of each store $\boldsymbol{\beta}_{s,i}$ follow a normal distribution whose mean depends on the store-specific covariates $\mathbf{z}_s$. Specifically, the mean of $\boldsymbol{\beta}_{s,i}$ is a linear function of $\mathbf{z}_s$; recall that $z_{s,0} = 1$, so that the first row of Δ specifies the intercept of the mean of $\boldsymbol{\beta}_{s,i}$. We note that log-log models like equation (12) are commonly used in demand modeling. The above distributions in equations (13) and (14) are sometimes referred to as first-stage priors.

For brand i's sales, we model Δ_i, $\tau_{1,i}, \ldots, \tau_{83,i}$ and $V_{\boldsymbol{\beta},i}$ as random variables with the following prior distributions:

$$\text{vec}(\Delta_i) \,|\, V_{\boldsymbol{\beta},i} \sim N(\text{vec}(\bar{\Delta}), V_{\boldsymbol{\beta},i} \otimes A^{-1}), \tag{15}$$

$$V_{\boldsymbol{\beta},i} \sim IW(\nu, V), \tag{16}$$

$$\tau_{s,i} \sim C/\chi^2_{\nu_\epsilon}, \quad \forall\ s \in \{1, \ldots, 83\}, \tag{17}$$

where $\text{vec}(\bar{\Delta})$ is the elements of the matrix $\bar{\Delta}$ (also 12-by-34) stacked into a column vector; $\otimes$ denotes the Kronecker product; $IW(\nu, V)$ is the inverse Wishart distribution with degrees of freedom ν and scale matrix V; and $\chi^2_{\nu_e}$ is a chi-squared distributed random variable with ν_e degrees of freedom. The matrices $\bar{\Delta}$, V and A and the scalars ν, ν_ϵ and C are the prior hyperparameters. The distributions (15), (16) and (17) are sometimes referred to as second-stage priors.

The second type of HB model that we will consider, which we will refer to as HB-SemiLog, we assume the same specification above except that instead of equation (12), where we assume the logarithm of sales is linear in the logarithm of prices, we assume it is linear in the actual prices themselves:

$$\log(q_{t,s,i}) = \beta_{s,0} + \boldsymbol{\beta}^T_{s,i,\mathbf{p}} \mathbf{p}_{t,s} + \boldsymbol{\beta}^T_{s,i,\mathbf{d}} \mathbf{d}_{t,s} + \boldsymbol{\beta}^T_{s,i,\mathbf{f}} \mathbf{f}_{t,s} + \epsilon_{t,s,i}, \tag{18}$$

This particular HB model is essentially the same model proposed in Montgomery (1997). One minor difference is that the above model allows for cross-brand promotion effects (e.g., brand 1 being on a deal or being featured can affect sales of brand 2). In our comparison of predictive accuracies in the next section, we will test both the above regression model, which allows for cross-brand promotion effects, and a simpler regression model with only own-brand promotion effects (i.e., equation (12) and (18) excludes $d_{t,s,i'}$ and $f_{t,s,i'}$ for brands i' different from i, as in Montgomery 1997).

In addition to these two HB models, we also consider a random forest model, which we denote by RF. The random forest model that we estimate will be different from the previous two HB models. Rather than predicting the sales $q_{t,s,i}$ or log sales $\log(q_{t,s,i})$, we will instead predict profit, that is,

$$(p_{t,s,i} - c_i) \cdot q_{t,s,i}, \tag{19}$$

where c_i is the unit cost of brand i. For each brand i, we estimate the random forest model using $p_1, \ldots, p_{11}, d_1, \ldots, d_{11}, f_1, \ldots, f_{11}, z_1, \ldots z_{11}$ as the predictor variables, where $z_1, \ldots, z_{11}$ are the store-level covariates of the store at which the prediction will be made. Letting F_i be the random forest prediction of profit for brand i, the overall profit that is predicted from the price and promotion decision $\mathbf{p}, \mathbf{d}, \mathbf{f}$ at a store with covariate vector $\mathbf{z}$ is simply the sum of these individual brand-level predictions:

$$F_{total}(\mathbf{p}, \mathbf{d}, \mathbf{f}, \mathbf{z}) = \sum_{i=1}^{11} F_i(\mathbf{p}, \mathbf{d}, \mathbf{f}, \mathbf{z}).$$

The choice to predict profit as opposed to sales or log of sales is motivated by tractability. By predicting the profit from each brand, the total profit function F_{total} can be directly used within the tree ensemble formulation. In contrast, if $F_i(\cdot)$ was a prediction of sales, then our objective function would be

$$\sum_{i=1}^{11} (p_i - c_i) \cdot F_i(\mathbf{p}, \mathbf{d}, \mathbf{f}, \mathbf{z}),$$

which would require modifications to our MIO formulation – one would need to model the tree ensemble behavior of each F_i, and then model the product of the price p_i and the predicted sales. Although our formulation can be suitably modified to do this, the resulting model is harder to solve than the basic problem (2).

7.3. Predictive accuracy results

Our first experiment does not consider optimization, but merely compares the out-of-sample predictive accuracy of the three models – HB-LogLog, HB-SemiLog and RF. To do so, we proceed as follows. We take the whole data set of store-week observations, and we split it randomly into a training set and a test set. We use the training set to estimate the HB-LogLog, HB-SemiLog and RF models for each brand. For each model, we predict the profit from brand i with the store-week observations in the test set, and compute the R^2 of the test set profit predictions. We repeat this procedure with ten random splits of the observations into train and test sets. In each split, 80% of the observations are randomly chosen for the training set, and the remaining 20% are used for the test set.

For HB-LogLog, we estimate it using Markov chain Monte Carlo (MCMC) using the R package `bayesm`. We use default values for the prior hyperparameters provided by `bayesm`. We run MCMC for 2000 iterations to obtain samples from the posterior distribution of $\boldsymbol{\beta}_s$ for each store s, and samples from the posterior distribution of τ_s for each store s. Due to mild autocorrelation in the draws of $\boldsymbol{\beta}_s$ and τ_s, we thin the draws by retaining every fifth draw. Of these thinned draws, we retain the last $J = 100$ samples. We index the draws/posterior samples by $j = 1, \ldots, J$. For an arbitrary price and promotion decision $\mathbf{p}, \mathbf{d}, \mathbf{f}$, we compute the predicted sales of brand i at store s by computing an approximation to the posterior expectation of sales $q_{s,i}$ under HB-LogLog as

$$\hat{q}_{s,i} = \frac{1}{J} \sum_{j=1}^{J} \exp\left(\beta_{s,i,0}^{(j)} + (\boldsymbol{\beta}_{s,i,\mathbf{p}}^{(j)})^T \log(\mathbf{p}) + (\boldsymbol{\beta}_{s,i,\mathbf{d}}^{(j)})^T \mathbf{d} + (\boldsymbol{\beta}_{s,i,\mathbf{f}}^{(j)})^T \mathbf{f} + \tau_{s,i}^{(j)}/2 \right),$$

where $\log(\mathbf{p})$ is the component-wise logarithm of $\mathbf{p}$, and quantities with the superscript (j) correspond to the jth posterior sample from the appropriate posterior distribution (that of either $\boldsymbol{\beta}_{s,i}$ or $\tau_{s,i}$). With this prediction of sales, we predict the profit of brand i as $(p_i - c_i) \cdot \hat{q}_{s,i}$.

For HB-SemiLog, we proceed in exactly the same way as for HB-LogLog, except that the predicted sales are computed as

$$\hat{q}_{s,i} = \frac{1}{J} \sum_{j=1}^{J} \exp\left(\beta_{s,i,0}^{(j)} + (\boldsymbol{\beta}_{s,i,\mathbf{p}}^{(j)})^T \mathbf{p} + (\boldsymbol{\beta}_{s,i,\mathbf{d}}^{(j)})^T \mathbf{d} + (\boldsymbol{\beta}_{s,i,\mathbf{f}}^{(j)})^T \mathbf{f} + \tau_{s,i}^{(j)}/2 \right).$$

For RF, we use the R package `ranger` to estimate one model for each brand's revenue. We use default parameters, with the exception of the number of trees which we vary in $\{20, 50, 100, 500\}$. For the RF model, we emphasize again that, unlike HB-LogLog and HB-SemiLog which first predict log sales and then translates this to revenue, the RF model directly predicts profits of each brand.

In addition to the above three models, we also consider slightly modified versions of the above three models where we do not allow for cross promotion effects (i.e., for each brand i's sales or profit prediction, we leave out $d_{i'}$ and $f_{i'}$ for $i' \neq i$ as independent variables). In our presentation below, we distinguish these models from the ones above by using the suffix "-Own" – thus, HB-LogLog-Own, HB-SemiLog-Own and RF-Own are the log-log, semi-log and random forest models with only own-brand promotion effects.

The prediction tasks we will consider involve predicting profit, which requires us to specify the unit cost c_i of each brand. Since the data set does not include this information, we will test two different sets of values for $\mathbf{c} = (c_1, \ldots, c_{11})$. In the first set, we set $c_i = 0$ for each brand i; we are thus effectively predicting the revenue from each brand. In the second set, we set $c_i = 0.9 \times p_{i,\min}$, where $p_{i,\min}$ is the lowest price at which brand i was offered in the whole data set.

Table 8 displays the test set/out-of-sample R^2 values for the profit predictions of each brand in the first prediction task ($c_i = 0$ for all brands i), averaged over the ten random splits of the data. From this table, we can see that both the HB-LogLog and HB-SemiLog models are very inaccurate for some brands, achieving R^2 values that are close to zero. In one case, namely brand 10, the out-of-sample R^2 is even negative, indicating that the model is worse than a naive model that

	Brand R^2										
Model	1	2	3	4	5	6	7	8	9	10	11
HB-SemiLog	0.34	0.74	0.41	0.07	0.38	0.53	0.47	0.34	0.25	–0.43	0.55
HB-SemiLog-Own	0.48	0.80	0.52	0.35	0.43	0.62	0.53	0.42	0.28	0.42	0.65
HB-LogLog	0.39	0.75	0.47	0.08	0.38	0.55	0.61	0.38	0.45	–0.17	0.58
HB-LogLog-Own	0.50	0.80	0.59	0.36	0.45	0.63	0.66	0.45	0.49	0.50	0.66
RF, $T = 20$	0.75	0.84	0.69	0.83	0.83	0.69	0.76	0.64	0.68	0.79	0.73
RF, $T = 50$	0.77	0.85	0.70	0.84	0.84	0.70	0.77	0.66	0.69	0.79	0.74
RF, $T = 100$	0.77	0.85	0.70	0.84	0.84	0.71	0.77	0.67	0.70	0.80	0.74
RF, $T = 500$	**0.78**	**0.85**	**0.70**	**0.84**	**0.84**	**0.71**	0.77	**0.67**	**0.70**	**0.80**	**0.74**
RF-Own, $T = 20$	0.73	0.84	0.68	0.81	0.79	0.70	0.75	0.64	0.68	0.74	0.72
RF-Own, $T = 50$	0.74	0.84	0.69	0.82	0.80	0.70	0.77	0.65	0.68	0.76	0.73
RF-Own, $T = 100$	0.74	0.84	0.69	0.82	0.80	0.71	0.77	0.66	0.68	0.76	0.73
RF-Own, $T = 500$	0.75	0.84	0.70	0.82	0.80	0.71	**0.77**	0.66	0.69	0.77	0.74

Table 8 Comparison of out-of-sample profit prediction R^2 for the different models and brands, averaged over ten random splits of the data, for the first prediction task ($c_i = 0$ for all brands i). The best R^2 value for each brand is indicated in bold.

just predicts the average training set profit. For the log-log model, when we remove cross-brand promotion effects and move from HB-LogLog to HB-LogLog-Own, the out-of-sample R^2 exhibits an absolute improvement ranging from 0.04 to 0.68, with an average over all brands of 0.15. For the semi-log model, when we remove cross-brand promotion effects and move from HB-SemiLog to HB-SemiLog-Own, the R^2 exhibits an absolute improvement ranging from 0.03 to 0.85, with an average of 0.17. With regard to the semi-log model, this finding is consistent with that of Montgomery (1997), where an own-brand promotion effect specification yielded a lower Schwartz information criterion value and higher out-of-sample accuracy than the cross-brand specification.

Comparing the HB models to the RF model with 500 trees, we can see that RF provides a significant improvement in predictive accuracy. For example, for brand 8, the highest R^2 attained by HB-LogLog, HB-LogLog-Own, HB-SemiLog and HB-SemiLog-Own, is 0.45. In contrast, the R^2 attained by RF with 500 trees is 0.67, which is an absolute improvement of 0.25. Over all of the brands, the improvement of RF over the best HB model for each brand ranges from 0.05 (brand 2) to as much as 0.48 (brand 4).

Within the family of RF models, Table 8 gives us a sense of how the number of trees affects the predictive accuracy. In particular, while the out-of-sample accuracy decreases as the number of trees is decreased, we can see that the loss in accuracy is very modest. For example, for brand 1, the R^2 is 0.7778 with 500 trees, which is reduced to 0.7663 when we use 50 trees; note that this is still higher than any of the HB models. This suggests that RF can still achieve an improvement over the HB models even with a smaller number of trees.

We can also determine the impact of cross promotion effects within RF. Note that unlike HB-LogLog and HB-SemiLog, where the out-of-sample R^2 improves once cross-promotion effects ($d_{i'}$ and $f_{i'}$ for i' different to the focal brand i) are removed, the opposite happens with the random forest models: RF-Own has slightly lower R^2 values than RF.

Table 9 presents the out-of-sample R^2 values for the profit predictions of each brand in the second prediction task ($c_i = 0.9 \times p_{i,\min}$ for all brands i), averaged over the ten random splits of the data. The same insights about the relative performance of the three different families of models derived from Table 8 hold for this case. Overall, these results provide evidence that a random forest model for profit predictions can outperform state-of-the-art models for this type of data. While a detailed comparison of random forests and hierarchical Bayesian models is beyond the scope of the present paper, we believe these result are encouraging and underscore the potential of tree ensemble models, such as random forests, to be used for profit/revenue prediction and for making marketing decisions.

	Brand R^2										
Model	1	2	3	4	5	6	7	8	9	10	11
HB-SemiLog	0.25	0.75	0.25	0.08	0.33	0.53	0.40	0.32	0.15	-0.62	0.55
HB-SemiLog-Own	0.39	0.81	0.39	0.31	0.38	0.63	0.53	0.44	0.30	0.32	0.66
HB-LogLog	0.32	0.76	0.30	0.13	0.36	0.55	0.49	0.34	0.31	-0.30	0.58
HB-LogLog-Own	0.44	0.81	0.43	0.32	0.42	0.64	0.59	0.44	0.43	0.42	0.67
RF, $T = 20$	0.72	0.84	0.62	0.84	0.82	0.71	0.75	0.65	0.65	0.77	0.74
RF, $T = 50$	0.74	0.84	0.63	0.84	0.83	0.72	0.76	0.66	0.67	0.79	0.75
RF, $T = 100$	0.75	0.85	0.63	0.84	0.83	0.72	0.76	0.67	0.67	0.79	0.76
RF, $T = 500$	**0.75**	**0.85**	**0.63**	**0.84**	**0.83**	**0.72**	0.76	**0.67**	**0.67**	**0.79**	**0.76**
RF-Own, $T = 20$	0.69	0.83	0.61	0.81	0.76	0.71	0.75	0.64	0.64	0.73	0.73
RF-Own, $T = 50$	0.70	0.84	0.62	0.81	0.77	0.72	0.76	0.66	0.66	0.74	0.74
RF-Own, $T = 100$	0.71	0.84	0.63	0.82	0.78	0.72	0.76	0.67	0.66	0.75	0.75
RF-Own, $T = 500$	0.72	0.84	0.63	0.82	0.78	0.72	**0.77**	0.67	0.66	0.75	0.75

Table 9 Comparison of out-of-sample profit prediction R^2 for the different models and brands, averaged over ten random splits of the data, for the second prediction task ($c_i = 0.9 p_{i,\min}$ for all brands i). The best R^2 value for each brand is indicated in bold.

7.4. Optimization results

We now turn our attention to optimization. Using the complete data set, we estimate the HB-LogLog-Own model, HB-SemiLog-Own model and RF model with 50 trees per brand. We fix the price of each brand c_i to $0.9 \times p_{i,\min}$. We restrict the price vector $\mathbf{p}$ to a set $\mathcal{P}$. We will specifically consider the following choice of $\mathcal{P}$:

$$\mathcal{P} = \prod_{i=1}^{11} \mathcal{P}_i \tag{20}$$

where

$$\mathcal{P}_i = \left\{ \delta \cdot \left\lceil \frac{p_{i,\min}}{\delta} \right\rceil, \delta \cdot \left\lceil \frac{p_{i,\min}}{\delta} \right\rceil + \delta, \delta \cdot \left\lceil \frac{p_{i,\min}}{\delta} \right\rceil + 2\delta, \ \ldots, \ \delta \cdot \left\lfloor \frac{p_{i,\max}}{\delta} \right\rfloor \right\}. \tag{21}$$

In the above definition of $\mathcal{P}_i$, δ is a discretization parameter – for example, $\delta = 0.05$ indicates that prices go up in increments of \$0.05 – and $p_{i,\min}$ and $p_{i,\max}$ are respectively the lowest and highest prices observed for brand i in the whole data set. In words, the above expression restricts brand i's price to go up in increments of δ, starting at the smallest multiple of δ above $p_{i,\min}$ and ending at the largest multiple of δ below $p_{i,\max}$; for example, if $p_{i,\min} = 1.44$ and $p_{i,\max} = 1.78$, then brand i's prices would be restricted to $\{1.45, 1.50, 1.55, 1.60, 1.65, 1.70, 1.75\}$. We will consider values of the discretization parameter $\delta \in \{0.05, 0.10, 0.20, 0.50\}$. For the purpose of this experiment, we do not consider optimization of $\mathbf{d}$ and $\mathbf{f}$, so we fix each d_i and f_i to zero.

For HB-LogLog-Own, our profit optimization problem for store s can thus be formulated as

$$\underset{\mathbf{p} \in \mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^{11} (p_i - c_i) \cdot \left[\frac{1}{J} \sum_{j=1}^{J} \exp\left(\beta_{s,i,0}^{(j)} + (\boldsymbol{\beta}_{s,i,\mathbf{p}}^{(j)})^T \log(\mathbf{p}) + \tau_{s,i}^{(j)}/2 \right) \right]. \tag{22}$$

For HB-SemiLog-Own, our profit optimization problem for store s is

$$\underset{\mathbf{p} \in \mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^{11} (p_i - c_i) \cdot \left[\frac{1}{J} \sum_{j=1}^{J} \exp\left(\beta_{s,i,0}^{(j)} + (\boldsymbol{\beta}_{s,i,\mathbf{p}}^{(j)})^T \mathbf{p} + \tau_{s,i}^{(j)}/2 \right) \right]. \tag{23}$$

We solve both problems (22) and (23) using local search from ten different randomly chosen starting points.

For RF, our profit optimization problem for store s is

$$\underset{\mathbf{p}\in\mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^{11} F_i(\mathbf{p}, \mathbf{0}, \mathbf{0}, \mathbf{z}_s), \tag{24}$$

where F_i is the random forest prediction function for the profit from brand i, $\mathbf{0}$ is a vector of zeros of the appropriate dimension (for the two arguments above, both are of length 11), and $\mathbf{z}_s$ is the vector of store-level covariates of store s. Regarding problem (24), we observe that:

1. The random forest defining each F_i may contain not only splits on $\mathbf{p}$, but also splits on $\mathbf{d}$, $\mathbf{f}$ and $\mathbf{z}$. However, $\mathbf{d}$, $\mathbf{f}$ and $\mathbf{z}$ are fixed to $\mathbf{0}$, $\mathbf{0}$ and $\mathbf{z}_s$, and are not decision variables.
2. While the actual split points on $p_1, \ldots, p_{11}$ could take any value, each p_i in the optimization problem (24) is restricted to values in $\mathcal{P}_i$.

These two observations are valuable because we can use them to simplify the tree model. In particular, the two observations can be used to identify splits/leaves that are unreachable in each tree, and to thus remove redundant splits. For example, if $z_1 = 10.5$ in $\mathbf{z}_s$ and a split in a tree has the query "Is $z_1 \leq 10.1$?", then all of the splits and leaves to the left of that split (the "yes" branch) can be removed because z_1 does not satisfy the query. As another example, suppose $\delta = 0.05$ in the definition of $\mathcal{P}$, and we take the "yes" branch for a split with query "Is $p_1 \leq 1.69$?"; if we then encounter the query "Is $p_1 \leq 1.66$?", we cannot reach the node on the "no" branch because this would imply $1.66 < p_1 \leq 1.69$, but p_1 is restricted to multiples of $\delta = 0.05$.

Once we have identified all such splits/leaves that cannot be reached, we can remove them and "collapse" the tree to obtain a much simpler, store-specific tree that is only in terms of $\mathbf{p}$. Using these collapsed trees, we formulate problem (24) using our MIO formulation (2) and solve it using the split generation approach. Due to the large number of MIO problems that need to be solved (83 stores by 4 discretization levels), we warm start the split generation approach by solving problem (24) using local search from ten random starting points, and using the best solution as the initial solution of problem (2). In addition, we also used slightly modified parameters for Gurobi, which we report in Section EC.3 of the electronic companion.

Table 10 displays the average and maximum time to solve problems (22), (23) and (24), for different values of δ, where the average and maximum are taken over the 83 stores in the chain. Note that for RF, the time includes both the local search warm start time as well as the split generation MIO time. From this table we can see that the two HB optimization problems are solved quite quickly; with the lowest level of discretization, the local search required no more than ten seconds to run. The RF problem (24) was solved in about 205 seconds – just over 3 minutes – on average per store for the finest discretization level $\delta = 0.05$. Although the RF problem is not solved quite as fast as the two HB-based problems, this is still a reasonable amount of time given the planning nature of the problem and the fact that the MIO approach provides a provably optimal solution, whereas the local search solution method needed for the two HB problems does not.

Price	RF		HB-SemiLog-Own		HB-LogLog-Own	
Increment δ	Avg.	Max.	Avg.	Max.	Avg.	Max.
0.05	205.5	550.5	7.9	9.9	6.7	9.1
0.10	140.2	345.7	4.0	5.4	3.4	4.5
0.20	56.5	160.9	2.1	3.2	1.9	3.2
0.50	12.1	23.9	1.0	1.4	0.9	1.2

Table 10 Average and maximum computation times (in seconds) of store-level optimal prices for the different models.

Aside from computation times, it is also interesting to compare both the optimal objective under each model for each of the stores. Figure 8 shows the smoothed density of optimal objectives of each

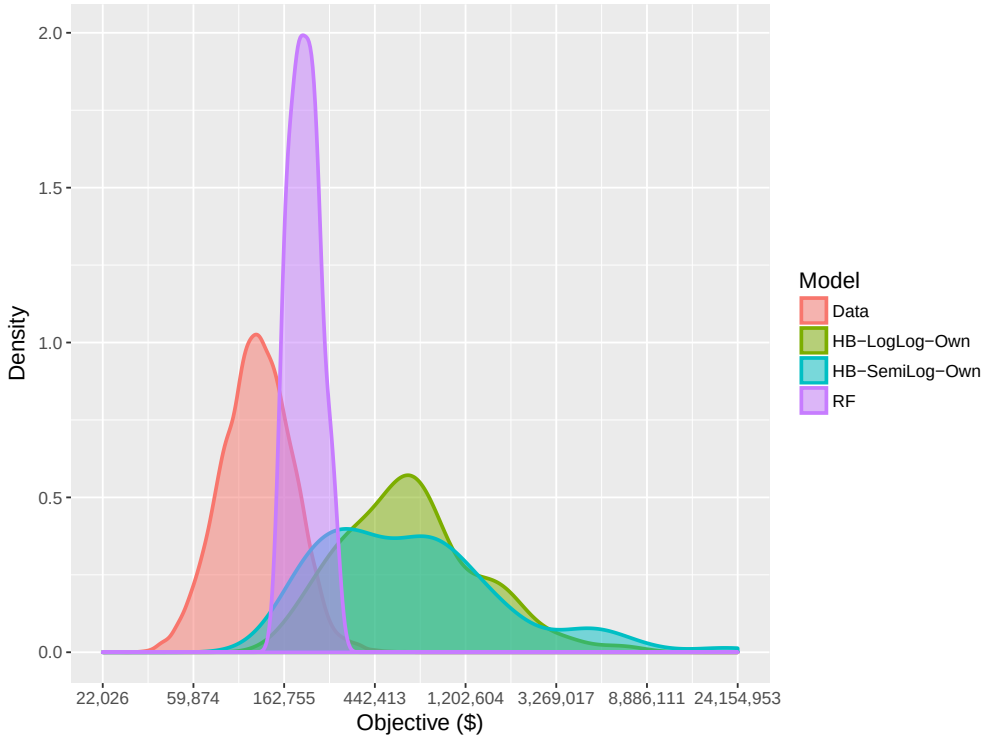

Figure 8 **Distribution of objective values (profits) under RF, HB-LogLog-Own and HB-SemiLog-Own, and the distribution of profits in the data (note that the x-axis is plotted on a log scale).**

of the three models that is induced by the 83 stores; note that the x-axis is shown on a log scale. In addition, we also plot the density of the actual revenues achieved in the whole data set, over all stores (including those where $\mathbf{f}$ and $\mathbf{d}$ are not necessarily $\mathbf{0}$). From this plot, we can see that there is a significant difference between the two HB models and the RF model with regard to the best attainable profit at each store. In particular, over all of the stores, the optimal profit with regard to RF is just under \$300,000. In contrast, the *average* optimal revenue for HB-LogLog is almost \$900,000, and for HB-SemiLog is over \$1,200,000, with the maximum optimal profit being over \$6,600,000 for HB-LogLog and over \$21,000,000 for HB-SemiLog. Note also that the distributions achieved by HB-LogLog and HB-SemiLog are also significantly different from the distribution of profits seen in the data.

This disparity occurs because of the forms of the log-log model (12) and the semi-log model (18). In particular, it is usually the case that for a focal brand i, the posterior samples of $\beta_{i,s,i}$ are negative, implying that lower prices increase log sales, while posterior samples of $\beta_{i,s,i'}$ for $i' \neq i$ tend to be positive, implying that high prices increase log sales. This matches our intuition (increasing the price of other brands will drive consumers to the focal brand, and decreasing the price of the focal brand will drive consumers to that brand). Due to the nature of the coefficients and the range of allowable prices, it turns out that usually, sales of one brand can be made very high (for example, exceeding 1,000,000 units) by setting that brand's price to the lowest possible value (i.e., $\delta \cdot \lceil p_{i,\min}/\delta \rceil$), and setting most or all of the brand's prices to their highest possible value (i.e., $\delta \cdot \lfloor p_{i,\max}/\delta \rfloor$).

To further emphasize this observation, Figure 9 shows the optimal prices under the three different models, over the 83 different stores, for each of the eleven brands; the solutions correspond to the increment $\delta = 0.05$. From this plot, we can see that in the overwhelming majority of cases, the optimal price vector $\mathbf{p}$ under HB-LogLog-Own and HB-SemiLog-Own involves setting each brand's price to the lowest or highest allowable price, which explains how the HB-LogLog and HB-SemiLog

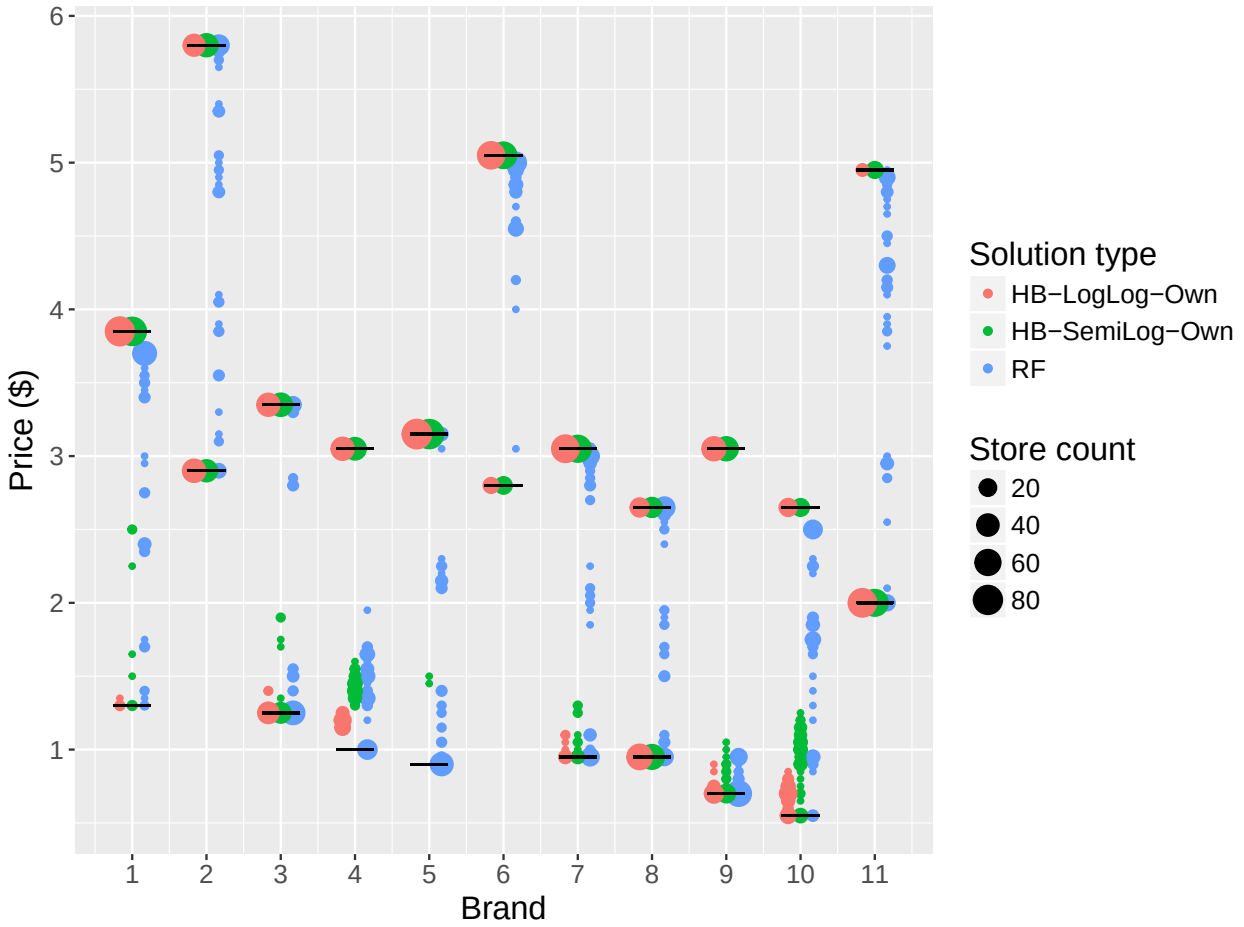

Figure 9 **Distribution of prices under RF, HB-LogLog-Own and HB-SemiLog-Own. The lowest and highest allowable price for each brand are indicated by black horizontal bars. The size of each circle indicates the number of stores for which the corresponding price was prescribed by the corresponding solution.**

model solutions achieve such high revenues under their respective models. In contrast, the price vectors obtained from RF are more interesting: for many stores, there are many brands whose prices are not set to the lowest or highest allowable price. Instead, the RF solutions cover a wider range of values within the price intervals of many brands.

Following on from Figure 9, we can also consider how close the prescribed price vectors are to the price vectors that have been realized in the data. For a store s and a method $m \in \{\text{RF}, \text{HB-LogLog-Own}, \text{HB-SemiLog-Own}\}$, let $\mathbf{p}_{m,s}$ be the optimal price vector, and let $\mathbf{p}_{t,s}$ be the (historical) vector of prices offered at store s in week t. We define the metric $D_{s,m}$ as the shortest distance in ℓ_1 norm between $\mathbf{p}_{m,s}$ and the collection $\{\mathbf{p}_{t,s}\}$:

$$D_{s,m} = \min_{t,s} \|\mathbf{p}_{m,s} - \mathbf{p}_{t,s}\|_1. \tag{25}$$

The metric $D_{s,m}$ measures how close the nearest historical price vector in the data is to $\mathbf{p}_{m,s}$, in terms of the total absolute difference in brand prices. Note that the minimization is taken over all observations, including observations from stores other than s and observations where $\mathbf{d}$ and $\mathbf{f}$ were not necessarily set to zero. Table 11 shows the minimum, mean and maximum values of $D_{s,m}$, taken over all 83 stores, for each of the three methods, with discretization increment $\delta = 0.05$. From this table, we can see that the RF approach and the two HB approaches all prescribe price vectors that are appreciably different from those that arise in the data. For example, the smallest value

	Model m		
Metric	RF	HB-LogLog-Own	HB-SemiLog-Own
$\min_s D_{s,m}$	3.64	3.61	4.90
$\text{mean}_s D_{s,m}$	6.82	7.85	7.51
$\max_s D_{s,m}$	9.00	10.29	9.81

Table 11 Distance of prescribed price vectors by RF, HB-LogLog-Own and HB-SemiLog-Own to historical price vectors.

of $D_{s,\text{RF}}$ over all of the stores is 3.64, which means that the price vector prescribed by RF for any store differs from any historical price vector in ℓ_1 norm by at least \$3.64.

Overall, from a prescriptive standpoint, we see that the RF profit optimization problem can be solved quickly to provable optimality (Table 10), leads to reasonable predicted profits (Figure 8) and recommends prices that are novel and different from previously adopted prices (Table 11). We believe that these results, combined with the strong out-of-sample predictive accuracy shown in Section 7.3, underscore the potential benefit of random forests and tree ensemble models in customized pricing, as well as the value of our optimization methodology in transforming such tree ensemble models into pricing decisions.

8. Conclusion

In this paper, we developed a modern optimization approach to the problem of finding the decision that optimizes the prediction of a tree ensemble model. At the heart of our approach is a compact mixed-integer optimization formulation that models the action of each tree in the ensemble. We showed that this formulation is better than a general alternate formulation, that one can construct an hierarchy of approximations to the formulation with bounded approximation quality through depth-based truncation and that one can exploit the structure of the formulation to derive efficient solution methods, based on Benders decomposition and based on split constraint generation. Using real data sets, we showed that our formulation can be solved quickly and outperforms heuristic approaches. We concluded with two case studies. In our drug design case study, we showed that our methodology scales to truly large instances, significantly outperforms a baseline heuristic approach and can be used to identify compounds that optimally trade off predicted performance and their similarity to existing compounds. In our customized pricing case study, we showed that random forests outperform hierarchical Bayesian models in out-of-sample profit prediction and that our optimization methodology can be used to rapidly obtain provably optimal price recommendations. Given the prevalence of tree models in predictive modeling, we believe that the methodology presented here will become an important asset in the modern business analytics toolbox, and is an exciting starting point for future research at the intersection of optimization and machine learning.

Acknowledgments

The author thanks Fernanda Bravo, Vishal Gupta and Auyon Siddiq for helpful conversations. The authors of Ma et al. (2015) and Merck Research Laboratories are gratefully acknowledged for the data set used in the drug design case study of Section 6. Dominick's Finer Foods and Peter Rossi are gratefully acknowledged for the data set used in the customized pricing case study of Section 7.

References

Atomwise Inc. Atomwise – Better Medicines Faster., 2017. Accessed May 27, 2017; available at `http://www.atomwise.com`.

P. Bajari, D. Nekipelov, S. P. Ryan, and M. Yang. Machine learning methods for demand estimation. *The American Economic Review*, 105(5):481–485, 2015.

D. Bertsimas and J. Dunn. Optimal classification trees. *Machine Learning*, to appear, 2017.

D. Bertsimas and N. Kallus. Pricing from observational data. *arXiv preprint arXiv:1605.02347*, 2016.

D. Bertsimas and A. King. OR Forum: An algorithmic approach to linear regression. *Operations Research*, 64(1):2–16, 2015.

D. Bertsimas and V. V. Mišić. Data-driven assortment optimization. *Working paper*, 2016.

D. Bertsimas, G. Lulli, and A. Odoni. An integer optimization approach to large-scale air traffic flow management. *Operations Research*, 59(1):211–227, 2011.

D. Bertsimas, A. King, and R. Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016a.

D. Bertsimas, A. O'Hair, S. Relyea, and J. Silberholz. An analytics approach to designing combination chemotherapy regimens for cancer. *Management Science*, 62(5):1511–1531, 2016b.

O. Besbes, R. Phillips, and A. Zeevi. Testing the validity of a demand model: An operations perspective. *Manufacturing & Service Operations Management*, 12(1):162–183, 2010.

J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

G. Biau and E. Scornet. A random forest guided tour. *TEST*, 25(2):197–227, 2016.

G. Biau, L. Devroye, and G. Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(Sep):2015–2033, 2008.

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

M. C. Cohen, N.-H. Z. Leung, K. Panchamgam, G. Perakis, and A. Smith. The impact of linear optimization on promotion planning. *Operations Research*, 65(2):446–468, 2017.

P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

Y. Crama. Concave extensions for nonlinear 0–1 maximization problems. *Mathematical Programming*, 61 (1-3):53–60, 1993.

D. R. Cutler, T. C. Edwards, K. H. Beard, A. Cutler, K. T. Hess, J. Gibson, and J. J. Lawler. Random forests for classification in ecology. *Ecology*, 88(11):2783–2792, 2007.

R. Díaz-Uriarte and S. A. De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

J. Elith, J. R. Leathwick, and T. Hastie. A working guide to boosted regression trees. *Journal of Animal Ecology*, 77(4):802–813, 2008.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

K. J. Ferreira, B. H. A. Lee, and D. Simchi-Levi. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, 18(1):69–88, 2015.

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

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

T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

J. Huuskonen. Estimation of aqueous solubility for a diverse set of organic compounds based on molecular topology. *Journal of Chemical Information and Computer Sciences*, 40(3):773–777, 2000.

N. Kallus. Recursive partitioning for personalization using observational data. *arXiv preprint arXiv:1608.08925*, 2016.

M. Kansy, F. Senner, and K. Gubernator. Physicochemical high throughput screening: parallel artificial membrane permeation assay in the description of passive absorption processes. *Journal of medicinal chemistry*, 41(7):1007–1010, 1998.

M. Kuhn and K. Johnson. *`AppliedPredictiveModeling`: Functions and Data Sets for 'Applied Predictive Modeling'*, 2014. URL `https://cran.r-project.org/web/packages/AppliedPredictiveModeling/index.html`. R package version 1.1-6.

A. Lemmens and C. Croux. Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, 43(2):276–286, 2006.

A. Liaw and M. Wiener. Classification and regression by `randomForest`. *R news*, 2(3):18–22, 2002.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

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

J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, 55(2):263–274, 2015.

A. L. Montgomery. Creating micro-marketing pricing strategies using supermarket scanner data. *Marketing Science*, 16(4):315–337, 1997.

J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

G. Ridgeway. `gbm`: Generalized boosted regression models. *R Package version 1.5-7*, 2006.

P. E. Rossi. *`bayesm`: Bayesian Inference for Marketing/Micro-econometrics*, 2012. URL `http://CRAN.R-project.org/package=bayesm`. R package version 2.2-5.

P. E. Rossi, G. M. Allenby, and R. E. McCulloch. *Bayesian statistics and marketing*. Wiley New York, 2005.

J. W. Scannell, A. Blanckley, H. Boldon, and B. Warrington. Diagnosing the decline in pharmaceutical r&d efficiency. *Nature Reviews Drug discovery*, 11(3):191–200, 2012.

R. E. Schapire and Y. Freund. *Boosting: Foundations and algorithms*. MIT press, 2012.

E. Scornet, G. Biau, and J.-P. Vert. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015.

V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of Chemical Information and Computer Sciences*, 43(6):1947–1958, 2003.

I. V. Tetko, V. Y. Tanchuk, T. N. Kasheva, and A. E. P. Villa. Estimation of aqueous solubility of chemical compounds using e-state indices. *Journal of chemical information and computer sciences*, 41(6):1488–1493, 2001.

H. R. Varian. Big data: New tricks for econometrics. *The Journal of Economic Perspectives*, 28(2):3–27, 2014.

J. P. Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57(1):3–57, 2015.

S. Wager and S. Athey. Estimation and inference of heterogeneous treatment effects using random forests. *arXiv preprint arXiv:1510.04342*, 2015.

S. Wager and G. Walther. Adaptive concentration of regression trees, with application to random forests. *arXiv preprint arXiv:1503.06388*, 2015.

M. N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017. doi: 10.18637/jss.v077.i01.

I.-C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

Electronic companion for "Optimization of Tree Ensembles"

EC.1. Proofs

EC.1.1. Auxiliary results

Before proceeding to the proofs, we first state two auxiliary results.

LEMMA EC.1. *For a given tree t, let $\ell \in \mathbf{leaves}(t)$. Then*

$$\{\ell' \in \mathbf{leaves}(t) \,|\, \ell' \neq \ell\} = \bigcup_{s \in \mathbf{LS}(\ell)} \mathbf{right}(s) \cup \bigcup_{s \in \mathbf{RS}(\ell)} \mathbf{left}(s). \tag{EC.1}$$

Furthermore, the collection of sets $\{\mathbf{right}(s) \,|\, s \in \mathbf{LS}(\ell)\} \cup \{\mathbf{left}(s) \,|\, s \in \mathbf{RS}(\ell)\}$ is pairwise disjoint.

To gain some intuition for the right hand set in equation (EC.1), recall that $\mathbf{LS}(\ell)$ is the set of splits for which we follow the left branch in order to reach ℓ, and $\mathbf{RS}(\ell)$ is the set of splits for which we follow the right branch to reach ℓ. For each $s \in \mathbf{LS}(\ell)$, $\mathbf{right}(s)$ is the set of leaves that is on the "wrong side" of split s (we take the left branch to reach ℓ, but each leaf in $\mathbf{right}(s)$ is only reachable by taking the right branch). Similarly, for each $s \in \mathbf{RS}(\ell)$, $\mathbf{left}(s)$ is the set of leaves that is on the wrong side of split s. The union of all leaves ℓ' on the wrong side of each split $s \in \mathbf{RS}(\ell) \cup \mathbf{LS}(\ell)$ covers all leaves except ℓ.

Proof of Lemma EC.1: We prove this by proving each inclusion. For the $\subseteq$ direction, let us fix $\ell' \in \mathbf{leaves}(t)$ such that $\ell' \neq \ell$. Then there must exist a split $\bar{s} \in \mathbf{splits}(t)$ such that either $\ell' \in \mathbf{left}(\bar{s})$, $\ell \in \mathbf{right}(\bar{s})$, or $\ell' \in \mathbf{right}(\bar{s})$, $\ell \in \mathbf{left}(\bar{s})$. In the former case, we have that $\bar{s} \in \mathbf{RS}(\ell)$, so that

$$\ell' \in \mathbf{left}(\bar{s}) \subseteq \bigcup_{s \in \mathbf{LS}(\ell)} \mathbf{right}(s) \cup \bigcup_{s \in \mathbf{RS}(\ell)} \mathbf{left}(s).$$

In the latter case, we have that $\bar{s} \in \mathbf{LS}(\ell)$, so that

$$\ell' \in \mathbf{right}(\bar{s}) \subseteq \bigcup_{s \in \mathbf{LS}(\ell)} \mathbf{right}(s) \cup \bigcup_{s \in \mathbf{RS}(\ell)} \mathbf{left}(s).$$

This proves the inclusion in the $\subseteq$ direction.

In the $\supseteq$ direction, we will argue the contrapositive. We have that

$$\begin{aligned}
\left[\bigcup_{s \in \mathbf{LS}(\ell)} \mathbf{right}(s) \cup \bigcup_{s \in \mathbf{RS}(\ell)} \mathbf{left}(s)\right]^C &= \bigcap_{s \in \mathbf{LS}(\ell)} (\mathbf{right}(s))^C \cap \bigcap_{s \in \mathbf{RS}(\ell)} (\mathbf{left}(s))^C \\
&\supseteq \bigcap_{s \in \mathbf{LS}(\ell)} \mathbf{left}(s) \cap \bigcap_{s \in \mathbf{RS}(\ell)} \mathbf{right}(s) \\
&\supseteq \{\ell\}
\end{aligned}$$

where the first step follows by De Morgan's law; the second follows by the fact that for any split $s \in \mathbf{splits}(t)$, $\mathbf{left}(s)$ and $\mathbf{right}(s)$ are disjoint; and the last by the definition of $\mathbf{LS}(\ell)$ and $\mathbf{RS}(\ell)$. This proves the inclusion in the $\supseteq$ direction, and thus proves the equivalence.

To show that the collection of sets is pairwise disjoint, order the splits in $\mathbf{LS}(\ell) \cup \mathbf{RS}(\ell)$ by their depth:

$$\mathbf{LS}(\ell) \cup \mathbf{RS}(\ell) = \{s_1, s_2, \ldots, s_K\},$$

where K is the total number of splits in $\mathbf{LS}(\ell) \cup \mathbf{RS}(\ell)$. Let us also define the sequence of sets $A_1, A_2, \ldots, A_K$ as

$$A_i = \begin{cases} \mathbf{left}(s_i) & \text{if } s_i \in \mathbf{LS}(\ell), \\ \mathbf{right}(s_i) & \text{if } s_i \in \mathbf{RS}(\ell), \end{cases}$$

and the sequence of sets $B_1, B_2, \ldots, B_K$ as

$$B_i = \begin{cases} \mathbf{right}(s_i) & \text{if } s_i \in \mathbf{LS}(\ell), \\ \mathbf{left}(s_i) & \text{if } s_i \in \mathbf{RS}(\ell). \end{cases}$$

We need to show that the collection $\{B_1, \ldots, B_K\}$ is disjoint. Observe that by the definition of $\mathbf{LS}(\ell)$ and $\mathbf{RS}(\ell)$, the sets $A_1, A_2, \ldots, A_K$ form a nested sequence, i.e.,

$$A_1 \supseteq A_2 \supseteq \cdots \supseteq A_K.$$

Notice also that for each $i \in \{2, \ldots, K\}$,

$$B_i \subseteq A_{i-1},$$

and for each $i \in \{1, \ldots, K\}$,

$$B_i \cap A_i = \emptyset.$$

It therefore follows that given $i, j \in \{1, \ldots, K\}$ with $i < j$, that

$$B_i \cap B_j \subseteq B_i \cap A_i = \emptyset,$$

which establishes that $\{B_1, \ldots, B_K\}$ are pairwise disjoint. □

In addition to Lemma EC.1, it will also be useful to state an analogous lemma for splits. With a slight abuse of notation, let us define $\mathbf{LS}(s)$ for a split $s \in \mathbf{splits}(t)$ as the sets of splits s' such that s is on the left of s'; similarly, we define $\mathbf{RS}(s)$ for a split $s \in \mathbf{splits}(t)$ as the set of splits s' such that s is on the right of s'. We then have the following lemma; the proof follows along similar lines to Lemma EC.1 and is omitted.

LEMMA EC.2. *For a given tree t, let $s \in \mathbf{splits}(t)$. We then have*

$$[\mathbf{left}(s) \cup \mathbf{right}(s)]^C = \bigcup_{s' \in \mathbf{RS}(s)} \mathbf{left}(s') \cup \bigcup_{s' \in \mathbf{LS}(s)} \mathbf{right}(s').$$

EC.1.2. Proof of Proposition 1

To prove that problem (1) is NP-Hard, we will show that it can be used to solve the minimum vertex cover problem. An instance of the minimum vertex cover problem is defined by a graph (V, E), where V is a a set of vertices and E is the set of edges between these vertices. The minimum vertex cover problem is to find the smallest set of vertices S from V such that each edge in E is incident to at least one vertex from the set S.

We now show how to cast this problem as a tree ensemble optimization problem. For convenience, let us index the vertices from 1 to $|V|$ and the edges from 1 to $|E|$. Also, let e_1 and e_2 be the nodes to which edge $e \in E$ is incident to. Suppose that our independent variable $\mathbf{X}$ is given by $\mathbf{X} = (X_1, \ldots, X_{|V|})$, where X_i is a numeric variable that is 1 or 0. The tree ensemble we will consider will consist of the following two types of trees:

1. **Type 1**: Trees 1 to $|V|$, where for $i \in \{1, \ldots, |V|\}$,

$$f_i(\mathbf{X}) = \begin{cases} 0 \text{ if } X_i \leq 0.5, \\ 1 \text{ if } X_i > 0.5. \end{cases}$$

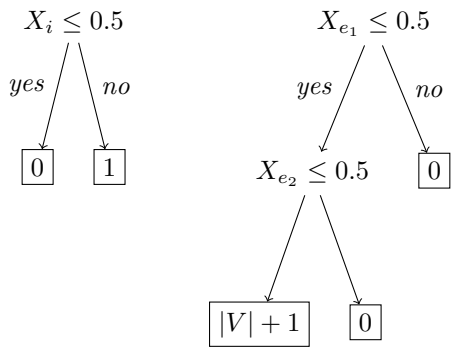

Figure EC.1 Type 1 tree (left) and type 2 tree (right) for vertex cover reduction.

2. **Type 2**: Trees $|V|+1$ to $|V|+|E|$, where for each edge $e \in E$,

$$f_{|V|+e}(\mathbf{X}) = \begin{cases} 0 & \text{if } X_{e_1} > 0.5, \\ 0 & \text{if } X_{e_1} \leq 0.5,\ X_{e_2} > 0.5, \\ +|V|+1 & \text{if } X_{e_1} \leq 0.5,\ X_{e_2} \leq 0.5. \end{cases}$$

These two types of trees are visualized in Figure EC.1.

We let the weight λ_t of each tree t be -1. The corresponding tree ensemble optimization problem is

$$\underset{\mathbf{X} \in \{0,1\}^{|V|}}{\text{maximize}} \ -\sum_{t=1}^{|V|} f_t(\mathbf{X}) - \sum_{t=|V|+1}^{|V|+|E|} f_t(\mathbf{X}). \tag{EC.2}$$

The above problem is identical to the minimum vertex cover problem. In particular, the independent variable $\mathbf{X}$ encodes the cover; $X_i = 1$ indicates that vertex i is part of the cover, and $X_i = 0$ indicates that vertex i is not in the cover. The type 1 trees count the size of the cover, while the type 2 trees penalize the solution if an edge is not covered by the set. More precisely, to understand the role of the type 2 trees, observe that:

- If the set of vertices encoded by $\mathbf{X}$ is a feasible cover, then $\sum_{t=|V|+1}^{|V|+|E|} f_t(\mathbf{X}) = 0$, and the objective only consists $-\sum_{t=1}^{|V|} f_t(\mathbf{X})$, which counts the number of vertices in the set encoded by $\mathbf{X}$.
- If the set of vertices encoded by $\mathbf{X}$ is not a feasible cover, then $f_t(\mathbf{X}) = |V|+1$ for at least one $t \in \{|V|+1, \ldots, |V|+|E|\}$, and therefore the objective satisfies the bound

$$-\sum_{t=1}^{|V|} f_t(\mathbf{X}) - \sum_{t=|V|+1}^{|V|+|E|} f_t(\mathbf{X}) \leq -(|V|+1). \tag{EC.3}$$

Observe that the bound in inequality (EC.3) is strictly worse than selecting all of the vertices, that is, setting $X_1 = X_2 = \cdots = X_{|V|} = 1$; using all of the vertices corresponds to an objective value of $-|V|$. Therefore, at optimality, the set of vertices encoded by $\mathbf{X}$ must be a feasible cover. As stated above, the objective value of $\mathbf{X}$ when it corresponds to a feasible cover reduces to

$$-\sum_{t=1}^{|V|} f_t(\mathbf{X}) = -|\{i \,|\, X_i = 1\}|,$$

which is (the negative of) the size of the set of vertices encoded by $\mathbf{X}$. Maximizing this quantity is equivalent to minimizing its negative, which is the same as minimizing the size of the set of vertices that covers E. Therefore, solving problem (EC.2) is equivalent to solving the minimum vertex cover problem for (V, E).

Since the minimum vertex cover problem is NP-Complete (Garey and Johnson 1979), it follows that the tree ensemble optimization problem (1) is NP-Hard. $\square$

EC.1.3. Proof of Proposition 2

To prove the proposition, we will show that the optimal solution $(\mathbf{x}, \mathbf{y})$ of the relaxation of problem (2) is a feasible solution of the relaxation of the standard linearization problem (4). Since the objective functions of problems (2) and (4) are the same, it will follow that the objective of $(\mathbf{x}, \mathbf{y})$, which is Z^*_{LO}, is less than or equal to $Z^*_{LO,StdLin}$, which is the optimal objective value of problem (4).

Let $(\mathbf{x}, \mathbf{y})$ be the optimal solution of the relaxation of problem (2). To show it is feasible for the relaxation of problem (4), we need to show that it satisfies the constraints of that formulation. We only need to show that constraints (4b) – (4d) are satisfied, since the other constraints of problem (4) are the same as in problem (2).

To verify constraint (4b), observe that $(\mathbf{x}, \mathbf{y})$ satisfies constraint (2c). For any $t \in \{1, \ldots, T\}$, $\ell \in \mathbf{leaves}(t)$ and $s \in \mathbf{LS}(\ell)$, we have

$$\begin{aligned}\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} &\geq \sum_{\ell' \in \mathbf{left}(s)} y_{t,\ell'} \\ &\geq y_{t,\ell}\end{aligned}$$

where the first inequality is exactly constraint (2c), and the second inequality follows because $\ell \in \mathbf{left}(s)$ (this is because $s \in \mathbf{LS}(\ell)$) and all $y_{t,\ell'}$'s are nonnegative (by constraint (2h)). Therefore, $(\mathbf{x}, \mathbf{y})$ satisfies constraint (4b). Similar reasoning can be used to establish constraint (4c).

To verify constraint (4d), observe that $(\mathbf{x}, \mathbf{y})$ satisfies

$$\sum_{\ell' \in \mathbf{leaves}(t)} y_{t,\ell'} \geq 1,$$

for any tree t, by virtue of constraint (2b). Fix a tree t and a leaf $\ell \in \mathbf{leaves}(t)$, and re-arrange the above to obtain

$$y_{t,\ell} \geq 1 - \sum_{\ell' \neq \ell} y_{t,\ell'}. \tag{EC.4}$$

We then have

$$\begin{aligned}y_{t,\ell} &\geq 1 - \sum_{\ell' \neq \ell} y_{t,\ell'} \\ &= 1 - \sum_{s \in \mathbf{LS}(\ell)} \sum_{\ell' \in \mathbf{right}(s)} y_{t,\ell'} - \sum_{s \in \mathbf{RS}(\ell)} \sum_{\ell' \in \mathbf{left}(s)} y_{t,\ell'} \\ &\geq 1 - \sum_{s \in \mathbf{LS}(\ell)} (1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}) - \sum_{s \in \mathbf{RS}(\ell)} (\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}) \\ &= 1 - \sum_{s \in \mathbf{LS}(\ell)} (1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}) - \sum_{s \in \mathbf{RS}(\ell)} (1 - (1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j})) \\ &= \sum_{s \in \mathbf{LS}(\ell)} \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} + \sum_{s \in \mathbf{RS}(\ell)} (1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}) - (|\mathbf{LS}(\ell)| + |\mathbf{RS}(\ell)| - 1)\end{aligned}$$

where the first inequality is just inequality (EC.4) from earlier, and the first equality follows from Lemma EC.1; the second inequality follows by constraints (2c) and (2d); and the last two equalities follow by simple algebra. This establishes that $(\mathbf{x}, \mathbf{y})$ satisfies constraint (4d). This establishes that $(\mathbf{x}, \mathbf{y})$ is feasible for problem (4), which concludes the proof. □

EC.1.4. Proof of Proposition 3

Observe that the sets of tree-split pairs are nested in the following way:

$$\bar{\Omega}_1 \subseteq \bar{\Omega}_2 \subseteq \cdots \subseteq \bar{\Omega}_{d_{\max}}.$$

As a consequence, the feasible region of problem (5) at depth d is a superset of the feasible region of problem (5) at depth $d+1$, and so we have

$$Z^*_{MIO,1} \geq Z^*_{MIO,2} \geq \cdots \geq Z^*_{MIO,d_{\max}}.$$

The equality $Z^*_{MIO,d_{\max}} = Z^*_{MIO}$ follows by the definition of $d_{\max}$ as the maximum depth of any tree in the ensemble. Combining this equality with the above sequence of inequalities establishes the result. $\square$

EC.1.5. Proof of Theorem 1

Let $(\mathbf{x}, \mathbf{y})$ be the solution of the depth d problem (i.e., problem (5) with $\bar{\Omega}_d$). Let $Z^*_{MIO,d}$ be the objective value of $(\mathbf{x}, \mathbf{y})$ within problem (5) with $\bar{\Omega}_d$.

Let $(\mathbf{x}, \tilde{\mathbf{y}})$ be the solution for problem (2) (the full-depth problem), obtained by finding the unique value of $\tilde{\mathbf{y}}$ such that $(\mathbf{x}, \tilde{\mathbf{y}})$ is feasible for problem (2). The existence and uniqueness of such a $\tilde{\mathbf{y}}$ is guaranteed by Proposition 4. Let Z_d be the objective value of $(\bar{\mathbf{x}}, \tilde{\mathbf{y}})$ within problem (2).

For a given tree $t \in \{1, \ldots, T\}$, let us consider the difference of the prediction of tree t for $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}, \tilde{\mathbf{y}})$:

$$\sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} y_{t,\ell} - \sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} \tilde{y}_{t,\ell}. \tag{EC.5}$$

In order to understand this quantity, we need to understand which part of the tree $\bar{\mathbf{x}}$ will get mapped to. We will do this through the following procedure:

1. Initialize ν to the root node of the tree.
2. If $\nu \in \mathbf{leaves}(t)$ or $\nu \in \mathbf{splits}(t, d)$, stop; otherwise:

- If $\sum_{j \in \mathbf{C}(\nu)} x_{\mathbf{V}(\nu),j} = 1$, set ν to its left child node;
- If $\sum_{j \in \mathbf{C}(\nu)} x_{\mathbf{V}(\nu),j} = 0$, set ν to its right child node;

and repeat step 2.

Upon termination, ν is some node in the tree – either a leaf or a depth d split node. Regardless of the type of node, we know that for any $s \in \mathbf{LS}(\nu)$, the depth of s is in $\{1, \ldots, d\}$, and so $(\mathbf{x}, \mathbf{y})$ satisfies the right split constraint (5c) of the depth d problem for s. By the definition of the procedure, it also must be that $\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} = 1$, which implies that

$$\begin{aligned}
&\sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}s,j} = 1 - 1 = 0, \\
&\Rightarrow \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} = 0, \\
&\Rightarrow y_{t,\ell} = 0, \quad \forall \ell \in \mathbf{right}(s).
\end{aligned}$$

For $s \in \mathbf{RS}(\nu)$, similar reasoning using the left split constraint (5b) allows us to assert that for any $s \in \mathbf{RS}(\nu)$,

$$\begin{aligned}
&\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}s,j} = 0, \\
&\Rightarrow \sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} = 0, \\
&\Rightarrow y_{t,\ell} = 0, \quad \forall \ell \in \mathbf{left}(s).
\end{aligned}$$

We thus know the following about $\mathbf{y}$:

$$y_{t,\ell} = 0, \quad \forall\ \ell \in \bigcup_{s \in \mathbf{LS}(\nu)} \mathbf{right}(s) \cup \bigcup_{s \in \mathbf{RS}(\nu)} \mathbf{left}(s).$$

We can also assert the same about $\tilde{\mathbf{y}}$, since $(\mathbf{x}, \tilde{\mathbf{y}})$ satisfies the left and right split constraints (2c) and (2d) at all depths:

$$\tilde{y}_{t,\ell} = 0, \quad \forall\ \ell \in \bigcup_{s \in \mathbf{LS}(\nu)} \mathbf{right}(s) \cup \bigcup_{s \in \mathbf{RS}(\nu)} \mathbf{left}(s).$$

We now consider three possible cases for the type of node ν is:

1. **Case 1**: $\nu \in \mathbf{leaves}(t)$. In this case, by Lemma EC.1, we can assert that

$$y_{t,\ell} = 0, \quad \forall\ \ell \neq \nu,$$

$$\tilde{y}_{t,\ell} = 0, \quad \forall\ \ell \neq \nu.$$

Since both $\mathbf{y}$ and $\tilde{\mathbf{y}}$ are nonnegative and sum to one, it follows that $y_{t,\nu} = \tilde{y}_{t,\nu} = 1$. We therefore have that the prediction difference (EC.5) is simply

$$\sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} y_{t,\ell} - \sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} \tilde{y}_{t,\ell} = p_{t,\nu} - p_{t,\nu} = 0. \tag{EC.6}$$

2. **Case 2**: $\nu \in \mathbf{splits}(t,d)$ and $\sum_{j \in \mathbf{C}(\nu)} x_{\mathbf{V}(s),j} = 1$. In this case, by Lemma EC.2, we have that

$$\begin{aligned} y_{t,\ell} &= 0, \quad \forall\ \ell \notin \mathbf{left}(\nu) \cup \mathbf{right}(\nu), \\ \tilde{y}_{t,\ell} &= 0, \quad \forall\ \ell \notin \mathbf{left}(\nu) \cup \mathbf{right}(\nu). \end{aligned}$$

In addition, since $\sum_{j \in \mathbf{C}(\nu)} x_{\mathbf{V}(s),j} = 1$, then by the right split constraints, we additionally have $y_{t,\ell} = 0$ and $\tilde{y}_{t,\ell} = 0$ for all $\ell \in \mathbf{right}(\nu)$, which implies that

$$\begin{aligned} y_{t,\ell} &= 0, \quad \forall\ \ell \notin \mathbf{left}(\nu), \\ \tilde{y}_{t,\ell} &= 0, \quad \forall\ \ell \notin \mathbf{left}(\nu). \end{aligned}$$

We can use the above properties of $\mathbf{y}_t$ and $\tilde{\mathbf{y}}_t$ to bound the prediction difference (EC.5) as follows:

$$\begin{aligned} \sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} y_{t,\ell} - \sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} \tilde{y}_{t,\ell} &\leq \max \left\{ \sum_{\ell} p_{t,\ell} y'_{t,\ell} \;\middle|\; \sum_{\ell} y'_{t,\ell} = 1;\ y'_{t,\ell} \geq 0,\ \forall \ell;\ y_{t,\ell} = 0,\ \forall \ell \notin \mathbf{left}(\nu) \right\} \\ &\quad - \min \left\{ \sum_{\ell} p_{t,\ell} y'_{t,\ell} \;\middle|\; \sum_{\ell} y'_{t,\ell} = 1;\ y'_{t,\ell} \geq 0,\ \forall \ell;\ y_{t,\ell} = 0,\ \forall \ell \notin \mathbf{left}(\nu) \right\} \\ &= \max_{\ell \in \mathbf{left}(\nu)} p_{t,\ell} - \min_{\ell \in \mathbf{left}(\nu)} p_{t,\ell}. \end{aligned}$$

3. **Case 3**: $\nu \in \mathbf{splits}(t,d)$ and $\sum_{j \in \mathbf{C}(\nu)} x_{\mathbf{V}(s),j} = 0$. Similar reasoning as in case 2 can be used to establish that

$$\begin{aligned} y_{t,\ell} &= 0, \quad \forall\ \ell \notin \mathbf{right}(\nu), \\ \tilde{y}_{t,\ell} &= 0, \quad \forall\ \ell \notin \mathbf{right}(\nu), \end{aligned}$$

and to bound the prediction difference (EC.5) as

$$\begin{aligned}\sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} y_{t,\ell} - \sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} \tilde{y}_{t,\ell} &\leq \max \left\{ \sum_{\ell} p_{t,\ell} y'_{t,\ell} \;\middle|\; \sum_{\ell} y'_{t,\ell} = 1;\ y'_{t,\ell} \geq 0,\ \forall \ell;\ y_{t,\ell} = 0,\ \forall \ell \notin \mathbf{right}(\nu) \right\} \\ &\quad - \min \left\{ \sum_{\ell} p_{t,\ell} y'_{t,\ell} \;\middle|\; \sum_{\ell} y'_{t,\ell} = 1;\ y'_{t,\ell} \geq 0,\ \forall \ell;\ y_{t,\ell} = 0,\ \forall \ell \notin \mathbf{right}(\nu) \right\} \\ &= \max_{\ell \in \mathbf{right}(\nu)} p_{t,\ell} - \min_{\ell \in \mathbf{right}(\nu)} p_{t,\ell}.\end{aligned}$$

Given cases 2 and 3, observe that if we know that $\nu \in \mathbf{splits}(t,d)$, then a valid upper bound on the prediction difference is simply

$$\begin{aligned}\sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} y_{t,\ell} - \sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} \tilde{y}_{t,\ell} &\leq \max \left\{ \max_{\ell \in \mathbf{left}(\nu)} p_{t,\ell} - \min_{\ell \in \mathbf{left}(\nu)} p_{t,\ell}, \max_{\ell \in \mathbf{right}(\nu)} p_{t,\ell} - \min_{\ell \in \mathbf{right}(\nu)} p_{t,\ell} \} \right\} \\ &= \delta_{t,\nu}.\end{aligned}$$

Now, if we do not know what type of node ν is – whether it is a leaf, or which split in $\mathbf{splits}(t,d)$ it is – then we can construct an upper bound, based on cases 1, 2 and 3 above, for the prediction difference as

$$\begin{aligned}\sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} y_{t,\ell} - \sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} \tilde{y}_{t,\ell} &\leq \max_{\nu \in \mathbf{splits}(t,d)} \delta_{t,\nu} \\ &= \Delta_t,\end{aligned}$$

where the maximum is defined to be zero if $\mathbf{splits}(t,d)$ is empty. (Note that in the case ν is a leaf, the above bound is valid, since all $\delta_{t,\nu}$ values are nonnegative by definition.)

Let us now unfix the tree t. Applying the above bound to bound the prediction difference of all trees t, and using the fact that $\lambda_t \geq 0$ for all t, it therefore follows that the difference between the objective $Z^*_{MIO,d}$ of $(\mathbf{x}, \mathbf{y})$ and the objective Z_d of $(\mathbf{x}, \tilde{\mathbf{y}})$ can be written as

$$\begin{aligned}Z^*_{MIO,d} - Z_d &= \sum_{t=1}^{T} \sum_{\ell \in \mathbf{leaves}(t)} \lambda_t \cdot p_{t,\ell} \cdot y_{t,\ell} - \sum_{t=1}^{T} \sum_{\ell \in \mathbf{leaves}(t)} \lambda_t \cdot p_{t,\ell} \cdot \tilde{y}_{t,\ell} \\ &= \sum_{t=1}^{T} \lambda_t \cdot \left(\sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} \cdot y_{t,\ell} - \sum_{\ell \in \mathbf{leaves}(t)} p_{t,\ell} \cdot \tilde{y}_{t,\ell} \right) \\ &\leq \sum_{t=1}^{T} \lambda_t \cdot \Delta_t.\end{aligned}$$

From here, it immediately follows that

$$Z^*_{MIO,d} - \sum_{t=1}^{T} \lambda_t \cdot \Delta_t \leq Z_d;$$

combining this with the fact that $(\mathbf{x}, \tilde{\mathbf{y}})$ is a feasible solution for problem (2) and Proposition 3 leads to the inequality,

$$Z^*_{MIO,d} - \sum_{t=1}^{T} \lambda_t \cdot \Delta_t \leq Z_d \leq Z^*_{MIO} \leq Z^*_{MIO,d},$$

as required. □

EC.1.6. Proof of Proposition 4

Feasibility. Let us first show that the proposed solution $\mathbf{y}_t$ is feasible. By construction, we have that $y_{t,\ell} \geq 0$ for all $\ell \in \mathbf{leaves}(t)$ and that $\sum_{\ell \in \mathbf{leaves}(t)} y_{t,\ell} = 1$, so constraints (8b) and (8e) are satisfied. This leaves the left and right split constraints (8c) and (8d).

For constraint (8c), let $s \in \mathbf{splits}(t)$. If $\ell^* \in \mathbf{left}(s)$, then it must be that $s \in \mathbf{LS}(\ell^*)$ and so by the definition of GetLeaf, it must be that $\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} = 1$. Therefore, we have:

$$\begin{aligned}\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} &= \sum_{\substack{\ell \in \mathbf{left}(s): \\ \ell \neq \ell^*}} y_{t,\ell} + y_{t,\ell^*} \\ &= 0 + 1 \\ &\leq \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \\ &= 1.\end{aligned}$$

Otherwise, if $\ell^* \notin \mathbf{left}(s)$, then

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

which is automatically less than or equal to $\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}$ (the latter can only be 0 or 1).

For constraint (8d), let $s \in \mathbf{splits}(t)$. If $\ell^* \in \mathbf{right}(s)$, then $s \in \mathbf{RS}(\ell^*)$ and by the definition of GetLeaf, it must be that $1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} = 1$. Therefore, applying similar reasoning as above, we get

$$\begin{aligned}\sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} &= \sum_{\substack{\ell \in \mathbf{right}(s): \\ \ell \neq \ell^*}} y_{t,\ell} + y_{t,\ell^*} \\ &= 0 + 1 \\ &\leq 1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \\ &= 1.\end{aligned}$$

Otherwise, if $\ell^* \notin \mathbf{right}(s)$, then again, we have $\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} = 0$, which is automatically less than or equal to $1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j}$ (again, it can only be 0 or 1). This establishes that $\mathbf{y}_t$ is a feasible solution to the subproblem (8).

Unique feasible solution. To establish that the proposed solution is the only feasible solution, we proceed as follows. We will show that if a solution $\mathbf{y}_t$ is a feasible solution of problem (8), then it must be equal to the solution of the statement of the proposition.

Let $\ell \in \mathbf{leaves}(t)$ such that $\ell \neq \ell^*$. Then by Lemma EC.1, we have that

$$\ell \in \bigcup_{s \in \mathbf{LS}(\ell^*)} \mathbf{right}(s) \cup \bigcup_{s \in \mathbf{RS}(\ell^*)} \mathbf{left}(s).$$

Moreover, the collection of sets in the union above is disjoint. Therefore, either $\ell \in \mathbf{right}(\bar{s})$ for some $\bar{s} \in \mathbf{LS}(t)$ or $\ell \in \mathbf{left}(\bar{s})$ for some $\bar{s} \in \mathbf{RS}(t)$.

In the former case – that is, $\ell \in \mathbf{right}(\bar{s})$ for some $\bar{s} \in \mathbf{LS}(\ell^*)$ – we have by constraint (8c) and constraint (8e) that

$$\begin{aligned}1 - \sum_{j \in \mathbf{C}(\bar{s})} x_{\mathbf{V}(\bar{s}),j} &\geq \sum_{\ell' \in \mathbf{right}(\bar{s})} y_{t,\ell'} \\ &\geq y_{t,\ell}.\end{aligned}$$

Therefore, $y_{t,\ell}$ is upper bounded by $1 - \sum_{j \in \mathbf{C}(\bar{s})} x_{\mathbf{V}(\bar{s}),j}$ and lower bounded by 0 (by constraint (8e)). Since $\bar{s} \in \mathbf{LS}(\ell^*)$ and from the definition of GetLeaf, it must be that $\sum_{j \in \mathbf{C}(\bar{s})} x_{\mathbf{V}(\bar{s}),j} = 1$, or equivalently, $1 - \sum_{j \in \mathbf{C}(\bar{s})} x_{\mathbf{V}(\bar{s}),j} = 0$. Therefore, $y_{t,\ell}$ must be equal to zero.

Similarly, if $\ell \in \mathbf{left}(\bar{s})$ for some $\bar{s} \in \mathbf{RS}(\ell^*)$, then by constraints (8d) and (8e) that

$$\begin{aligned}\sum_{j \in \mathbf{C}(\bar{s})} x_{\mathbf{V}(\bar{s}),j} &\geq \sum_{\ell' \in \mathbf{left}(\bar{s})} y_{t,\ell'} \\ &\geq y_{t,\ell}.\end{aligned}$$

Therefore, $y_{t,\ell}$ is upper bounded by $\sum_{j \in \mathbf{C}(\bar{s})} x_{\mathbf{V}(\bar{s}),j}$ and lower bounded by 0 (by (8e)). Since $\bar{s} \in \mathbf{LS}(\ell^*)$ and from the definition of GetLeaf, it must be that $\sum_{j \in \mathbf{C}(\bar{s})} x_{\mathbf{V}(\bar{s}),j} = 0$. Therefore, $y_{t,\ell}$ must be equal to zero.

From the above reasoning, we have shown that $y_{t,\ell} = 0$ for every $\ell \neq \ell^*$. By constraint (8b), it must be that $y_{t,\ell^*} = 1 - \sum_{\ell \neq \ell^*} y_{t,\ell} = 1 - 0 = 1$. The resulting solution is therefore *exactly the same* as the one proposed in the proposition; it follows that the proposed solution $\mathbf{y}_t$ is the only feasible solution to problem (8).

Optimality. Since $\mathbf{y}_t$ is the only feasible solution of problem (8), it must also be its optimal solution. This completes the proof. □

EC.1.7. Proof of Proposition 5

First, let us check the dual objective of the proposed solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$. We have

$$\begin{aligned}&\sum_{s \in \mathbf{splits}(t)} \alpha_{t,s} \left[\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right] + \sum_{s \in \mathbf{splits}(t)} \beta_{t,s} \left[1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right] + \gamma_t \\ &= \sum_{s \in \mathbf{RS}(\ell^*)} \alpha_{t,s} \left[\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right] + \sum_{s \in \mathbf{LS}(\ell^*)} \beta_{t,s} \left[1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \right] + p_{t,\ell^*} \\ &= 0 + 0 + p_{t,\ell^*} \\ &= p_{t,\ell^*},\end{aligned}$$

where the first step follows by the definition of $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$; the second step follows by the fact that $\sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} = 0$ for $s \in \mathbf{RS}(\ell^*)$ and $1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} = 0$ for $s \in \mathbf{LS}(\ell^*)$; and the last two steps by algebra. The final value is exactly equal to the objective of the primal optimal solution. If $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is feasible for the dual problem, then the proposition will be proven.

To verify feasibility, observe that by their definition, we have $\alpha_{t,s} \geq 0$ and $\boldsymbol{\beta}_{t,s} \geq 0$ for all $s \in \mathbf{splits}(t)$. Thus, we only need to check constraint (9b) for each $\ell \in \mathbf{leaves}(t)$. We consider two cases:

Case 1: $\ell = \ell^*$. In this case, proceeding from the left hand side of constraint (9b) for ℓ^*, we have

$$\begin{aligned}&\sum_{s \in \mathbf{LS}(\ell^*)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell^*)} \beta_{t,s} + \gamma_t \\ &= 0 + 0 + p_{t,\ell^*} \\ &\geq p_{t,\ell^*},\end{aligned}$$

where the first equality follows because $\alpha_{t,s} = 0$ for all $s \notin \mathbf{RS}(\ell^*)$ and $\beta_{t,s} = 0$ for all $s \notin \mathbf{LS}(\ell^*)$ (note that $\mathbf{LS}(\ell^*) \cap \mathbf{RS}(\ell^*) = \emptyset$; a leaf cannot be both to the left of and to the right of the same split), and also because $\gamma_t = p_{t,\ell^*}$ by definition.

Case 2: $\ell \neq \ell^*$. In this case, by Lemma (EC.1), we know that ℓ satisfies

$$\ell \in \bigcup_{s \in \mathbf{LS}(\ell^*)} \mathbf{right}(s) \cup \bigcup_{s \in \mathbf{RS}(\ell^*)} \mathbf{left}(s).$$

Recall also that by Lemma (EC.1) that each set in the above union is disjoint. Therefore, we have that $\ell \in \mathbf{right}(\bar{s})$ for some $\bar{s} \in \mathbf{LS}(\ell^*)$ or $\ell \in \mathbf{left}(\bar{s})$ for some $\bar{s} \in \mathbf{RS}(\ell^*)$. We now show that the inequality holds in either of these two scenarios.

If $\ell \in \mathbf{right}(\bar{s})$ for some $\bar{s} \in \mathbf{LS}(\ell^*)$, then we have

$$\begin{aligned}
\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t &\geq \beta_{t,\bar{s}} + \gamma_t \\
&= \max\{0, \max_{\ell' \in \mathbf{right}(\bar{s})} (p_{t,\ell'} - p_{t,\ell^*})\} + p_{t,\ell^*} \\
&\geq p_{t,\ell} - p_{t,\ell^*} + p_{t,\ell^*} \\
&= p_{t,\ell},
\end{aligned}$$

where the first step follows because $\bar{s}$ must belong to $\mathbf{RS}(\ell)$ and because by definition, $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ are nonnegative; the second step follows by the definition of $\beta_{t,s}$ for $s \in \mathbf{LS}(\ell^*)$; the third step by the definition of the maximum; and the last step by algebra.

If $\ell \in \mathbf{left}(\bar{s})$ for some $\bar{s} \in \mathbf{RS}(\ell^*)$, then we have

$$\begin{aligned}
\sum_{s \in \mathbf{LS}(\ell)} \alpha_{t,s} + \sum_{s \in \mathbf{RS}(\ell)} \beta_{t,s} + \gamma_t &\geq \alpha_{t,\bar{s}} + \gamma_t \\
&= \max\{0, \max_{\ell' \in \mathbf{left}(\bar{s})} (p_{t,\ell'} - p_{t,\ell^*})\} + p_{t,\ell^*} \\
&= \geq p_{t,\ell} - p_{t,\ell^*} + p_{t,\ell^*} \\
&= p_{t,\ell},
\end{aligned}$$

which follows by logic similar to the first case ($\ell \in \mathbf{right}(\bar{s})$ for some $\bar{s} \in \mathbf{LS}(\ell^*)$).

Thus, we have established that $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ is a feasible solution for the dual problem (9) and achieves the same objective as the primal optimal solution. By weak LO duality, the solution $(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t, \gamma_t)$ must therefore be optimal. □

EC.1.8. Proof of Proposition 6

We consider this by proving the equivalence in each direction. In the $\Rightarrow$ direction, the implication is immediate, because if $(\mathbf{x}, \mathbf{y})$ satisfies constraints (2c) and (2d) for all $s \in \mathbf{splits}(t)$, it will also satisfy them for arbitrary subsets of $\mathbf{splits}(t)$.

Thus, we only need to establish the $\Leftarrow$ direction. In the $\Leftarrow$ direction, let $s \in \mathbf{LS}(\ell^*)$. For any tree t and any $s \in \mathbf{LS}(\ell^*)$, we have that for any $\ell \in \mathbf{right}(s)$, that

$$\begin{aligned}
y_{t,\ell} &\leq \sum_{\ell \in \mathbf{right}(s)} y_{t,\ell'} \\
&\leq 1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j} \\
&= 1 - 1 \\
&= 0,
\end{aligned}$$

where the first step follows since $y_{t,\ell'}$ is nonnegative for any ℓ', the second step by the hypothesis of the implication, and the remaining two steps by algebra. Since $y_{t,\ell}$ is nonnegative, it must be that $y_{t,\ell} = 0$. Similarly, for any $s \in \mathbf{RS}(\ell^*)$, we can show that for each $\ell \in \mathbf{left}(s)$, $y_{t,\ell} = 0$.

It therefore follows that for any $\ell \in \bigcup_{s \in \mathbf{RS}(\ell^*)} \mathbf{left}(s) \cup \bigcup_{s \in \mathbf{LS}(\ell^*)} \mathbf{right}(s)$, $y_{t,\ell}$. Invoking Lemma EC.1, we have that $y_{t,\ell} = 0$ for all $\ell \in \mathbf{leaves}(t) \setminus \{\ell^*\}$. Since $\mathbf{y}$ is assumed to satisfy constraint (2b), it must be that $y_{t,\ell^*} = 1$.

From here, we can see that $\mathbf{y}_t$, the collection of $y_{t,\ell}$ values corresponding to tree t, is defined exactly as in the statement of Proposition 4. Thus, invoking Proposition 4, we can assert that $\mathbf{y}$ satisfies

$$\sum_{\ell \in \mathbf{left}(s)} y_{t,\ell} \leq \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j},$$
$$\sum_{\ell \in \mathbf{right}(s)} y_{t,\ell} \leq 1 - \sum_{j \in \mathbf{C}(s)} x_{\mathbf{V}(s),j},$$

for all $s \in \mathbf{splits}(t)$. This concludes the proof. □

EC.2. Local search procedure

We provide the pseudocode of our local search procedure for approximately solving problem (1) below as Algorithm 3. Before we define the procedure, we define the set $\bar{\mathcal{X}}_i$ as

$$\bar{\mathcal{X}}_i = \mathcal{X}_i$$

for categorical variables, and

$$\bar{\mathcal{X}}_i = \{a_{i,j} \,|\, j \in \{1, \ldots, K_i\}\} \cup \{a_{i,K_i} + 1\}$$

for numeric variables, where $a_{i,j}$ is the jth smallest split point of variable i in the tree ensemble model. The $\bar{\mathcal{X}}_i$ are simply the domains of each independent variable defined in a way that will be helpful in defining our local search. For numeric variables, $\bar{\mathcal{X}}_i$ consists of the K_i split points of variable i and one extra point, $a_{i,K_i} + 1$. The extra point $a_{i,K_i} + 1$ is arbitrary. We can use any value here, as long as it is strictly larger than the largest split point of variable i, as this will allow us to choose to be on the right-hand side of a split with query $X_i \leq a_{i,K_i}$.

Algorithm 3 Local search procedure.

Require: Tree ensemble model $f_1(\cdot), \ldots, f_T(\cdot)$, $\lambda_1, \ldots, \lambda_T$; finite domains $\bar{\mathcal{X}}_1, \ldots, \bar{\mathcal{X}}_T$.
  Select $\mathbf{X} = (X_1, \ldots, X_n)$ uniformly at random from $\prod_{i'=1}^{n} \bar{\mathcal{X}}_{i'}$.
  Initialize $Z \leftarrow \sum_{t=1}^{T} \lambda_t f_t(\mathbf{X})$.
  Initialize **untestedVariables** $= \{1, \ldots, n\}$.
  while $|$**untestedVariables**$| > 0$ **do**
    Select $i \in$ **untestedVariables**.
    Set $\mathcal{N} \leftarrow \{\mathbf{X}' \in \prod_{i'=1}^{n} \bar{\mathcal{X}}_{i'} \,|\, X'_j = X_j \text{ for } j \neq i\}$.
    Set $\mathbf{X}^* \leftarrow \arg\max_{\mathbf{X}' \in \mathcal{N}} \sum_{t=1}^{T} f_t \lambda_t(\mathbf{X}')$.
    Set $Z_c \leftarrow \sum_{t=1}^{T} f_t \lambda_t(\mathbf{X}^*)$.
    if $Z_c > Z$ **then**
      Set $Z \leftarrow Z_c$.
      Set $\mathbf{X} \leftarrow \mathbf{X}^*$.
      Set **untestedVariables** $\leftarrow \{1, \ldots, i-1, i+1, \ldots, n\}$.
    else
      Set **untestedVariables** $\leftarrow$ **untestedVariables** $\setminus \{i\}$.
    end if
  end while
  return Locally optimal solution $\mathbf{X}$ with objective value Z.

EC.3. Customized pricing MIO Gurobi parameters

Table EC.1 shows the parameters used for Gurobi when solving the RF profit optimization problem in Section 7.4.

Parameter	Value
`Heuristics`	0
`Cuts`	0
`VarBranch`	1
`InfUnbdInfo`	1
`PrePasses`	1

Table EC.1 Gurobi parameters for RF profit optimization problem in Section 7.4.