Appendices for

Decision Trees for Decision-Making under the Predict-then-Optimize Framework

A. Proof of Theorem 1

Theorem 1. *Let $\bar{c}_l := \frac{1}{|R_l|}\sum_{i\in R_l} c_i$ denote the average cost of all observations within leaf l. If $\bar{c}_l$ has a unique minimizer in its corresponding decision problem, then $\bar{c}_l$ minimizes within-leaf SPO loss. More simply, if $|W^*(\bar{c}_l)| = 1$, then $\bar{c}_l = \arg\min_{\hat{c}_l} \sum_{i\in R_l} \ell_{SPO}(\hat{c}_l, c_i)$.*

Proof. Let $\bar{c}_l$ be defined as stated in the theorem. We will show that the within-leaf SPO loss associated with predicting $\bar{c}_l$ lower bounds that of predicting any other feasible cost vector $\hat{c}_l \in \mathbb{R}^d$. Let $N_l = |R_l|$ denote the number of observations within leaf l. The following holds for any $\hat{c}_l \in \mathbb{R}^d$:

$$\begin{aligned}
& \frac{1}{N_l}\sum_{i\in R_l} \ell_{SPO}(\bar{c}_l, c_i) - \frac{1}{N_l}\sum_{i\in R_l} \ell_{SPO}(\hat{c}_l, c_i) \\
=\ & \frac{1}{N_l}\sum_{i\in R_l} \max_{w\in W^*(\bar{c}_l)} \{c_i^T w\} - \frac{1}{N_l}\sum_{i\in R_l} \max_{w\in W^*(\hat{c}_l)} \{c_i^T w\} \\
=\ & \frac{1}{N_l}\sum_{i\in R_l} c_i^T w^*(\bar{c}_l) - \frac{1}{N_l}\sum_{i\in R_l} \max_{w\in W^*(\hat{c}_l)} \{c_i^T w\} \quad (W^*(\bar{c}_l) = \{w^*(\bar{c}_l)\} \text{ is a singleton}) \\
\leq\ & \frac{1}{N_l}\sum_{i\in R_l} c_i^T w^*(\bar{c}_l) - \max_{w\in W^*(\hat{c}_l)} \left\{\frac{1}{N_l}\sum_{i\in R_l} c_i^T w\right\} \\
=\ & \bar{c}_l^T w^*(\bar{c}_l) - \max_{w\in W^*(\hat{c}_l)} \{\bar{c}_l^T w\} \\
\leq\ & 0 \quad (\text{by definition of } w^*(\bar{c}_l))
\end{aligned}$$

We have thus demonstrated that $\bar{c}_l$ achieves a within-leaf SPO loss lower or equal to that of any other cost vector $\hat{c}_l \in \mathbb{R}^d$, thereby proving the theorem.

□

B. Proof of Theorem 2

Theorem 2. *Assume that the decision feasibility constraints $w \in S$ consist of only linear and integer constraints and that S is bounded. Then, optimization problem (4) may be equivalently expressed as the following MILP:*

$$\begin{aligned}
\min_{r,w,y}\quad & \frac{1}{n}\sum_{l=1}^{L}\sum_{i=1}^{n} y_{il} - \sum_{i=1}^{n} z^*(c_i) \\
\text{s.t.}\quad & y_{il} \geq c_i^T w_l - M_1(1 - r_{il}), && \forall i \in \{1...n\}, l \in \{1...L\}, \\
& y_{il} \geq -M_2 r_{il} && \forall i \in \{1...n\}, l \in \{1...L\}, \\
& w_l \in S && \forall l \in \{1...L\}, \\
& r_{il} \in \mathcal{T} && \forall i \in \{1...n\}, l \in \{1...L\}
\end{aligned} \tag{7}$$

Proof. Let $N_l = |R_l|$ denote the number of observations within leaf l. We first perform the following algebraic operations starting with optimization problem (4):

$$
\begin{aligned}
&\min_{R_{1:L}\in\mathcal{T}} \frac{1}{n}\sum_{l=1}^{L}\sum_{i\in R_l}\left(c_i^T w^*(\bar{c}_l) - z^*(c_i)\right)\\
=&\min_{R_{1:L}\in\mathcal{T}} \frac{1}{n}\sum_{l=1}^{L}\left(N_l \bar{c}_l^T w^*(\bar{c}_l) - \sum_{i\in R_l} z^*(c_i)\right)\\
=&\min_{R_{1:L}\in\mathcal{T}} \frac{1}{n}\sum_{l=1}^{L}\left(N_l \min_{w_l\in S}\{\bar{c}_l^T w_l\} - \sum_{i\in R_l} z^*(c_i)\right)\\
=&\min_{\substack{R_{1:L}\in\mathcal{T}\\ w_{1:L}\in S}} \frac{1}{n}\sum_{l=1}^{L}\left(N_l \bar{c}_l^T w_l - \sum_{i\in R_l} z^*(c_i)\right)\\
=&\min_{\substack{R_{1:L}\in\mathcal{T}\\ w_{1:L}\in S}} \frac{1}{n}\sum_{l=1}^{L}\sum_{i\in R_l}\left(c_i^T w_l - z^*(c_i)\right) .
\end{aligned}
$$

Let r_{il} denote a binary variable which indicates whether training observation i belongs to leaf R_l. Then,

$$
\begin{aligned}
&\min_{\substack{R_{1:L}\in\mathcal{T}\\ w_{1:L}\in S}} \frac{1}{n}\sum_{l=1}^{L}\sum_{i\in R_l}\left(c_i^T w_l - z^*(c_i)\right)\\
=&\min_{\substack{r_{1:L}\in\mathcal{T}\\ w_{1:L}\in S}} \frac{1}{n}\sum_{l=1}^{L}\left(\sum_{i=1}^{n} r_{il} c_i^T w_l\right) - \sum_{i=1}^{n} z^*(c_i)\\
=&\min_{\substack{r_{1:L}\in\mathcal{T}\\ w_{1:L}\in S\\ y_{1:L}}} \frac{1}{n}\sum_{l=1}^{L}\sum_{i=1}^{n} y_{il} - \sum_{i=1}^{n} z^*(c_i)\,,
\end{aligned}
$$

where in the last step we add the constraint that $y_{il} = r_{il}c_i^T w_l$ for every i and l. First, note that this constraint may be equivalently expressed as $y_{il} \geq r_{il}c_i^T w_l$, as y_{il} will always be set equal to its minimum feasible value ($r_{il}c_i^T w_l$) since it is being minimized in the objective function. However, this constraint is still not linear since it involves the multiplication of two decision variables r_{il} and w_l. We may rewrite it as the two linear constraints below:

$$
y_{il} \geq c_i^T w_l - M_1(1-r_{il}) \quad \text{and} \quad y_{il} \geq -M_2 r_{il}\,.
$$

Above, M_1 and M_2 are constants which upper bound $c_i^T w_l$ and $-c_i^T w_l$, respectively, for all $i \in \{1, 2, \ldots, n\}$ and $w_l \in S$. We therefore define $M_1 := \max\{\max_{i,w\in S} c_i^T w, 0\}$ and $M_2 := \max\{\max_{i,w\in S} -c_i^T w, 0\}$ which are finite due to S being bounded. Note that when the cost vectors are all nonnegative (nonpositive), then $M_2 = 0$ ($M_1 = 0$) assuming the decision variables w are nonnegative for all feasible $w \in S$. Thus, the optimization problem for training decision trees under SPO loss may be written as the following mixed integer linear program:

$$
\begin{aligned}
\min_{r,w,y}\quad & \frac{1}{n}\sum_{l=1}^{L}\sum_{i=1}^{n} y_{il} - \sum_{i=1}^{n} z^*(c_i)\\
\text{s.t.}\quad & y_{il} \geq c_i^T w_l - M_1(1-r_{il}), && \forall i \in \{1...n\}, l \in \{1...L\},\\
& y_{il} \geq -M_2 r_{il} && \forall i \in \{1...n\}, l \in \{1...L\},\\
& w_l \in S && \forall l \in \{1...L\},\\
& r_{il} \in \mathcal{T} && \forall i \in \{1...n\}, l \in \{1...L\}
\end{aligned}
$$

□

C. Encoding Decision Trees using Integer and Linear Constraints

Here we provide the complete formulation of $r_{il} \in \mathcal{T}$ as integer and linear constraints using the decision tree encoding proposed in Bertsimas & Dunn (2017). As it is only covered briefly here, we encourage the reader to examine Bertsimas & Dunn (2017) for a more thorough treatment of the materials below. We assume that the practitioner has specified the following parameters regulating the growth of the tree during the training procedure: (1) the depth H of the tree being trained, and (2) the minimum number of training observations N_{min} permitted to be in each leaf of the tree. We consider training a *complete* tree of depth H, define as a tree in which all leaves have a depth of H. Let L denote the number of leaves in the tree, and index each leaf by $l \in \mathcal{T}_L := \{1, 2, ..., L\}$. Further, let B denote the number of branch nodes (i.e., splitting nodes) within the tree, and index each branch node by $t \in \mathcal{T}_B := \{1, 2, ..., B\}$. Note that $L = 2^H$ and $B = 2^H - 1$. Not all leaves in the tree are required to be active (i.e., contain training observations), and not all branch nodes are required to be active splits (i.e., partition the training observations). Indeed, leaves may be pooled together if their parent splits do not contribute significantly to minimizing the objective function. To keep track of the active leaves and branch nodes, let k_l= $\mathbb{I}\{\text{leaf } l \text{ is not empty}\}$ and d_t= $\mathbb{I}\{\text{branch node } t \text{ is an active split}\}$. If a branch node is *not* an active split, then it effectively considered as a leaf with respect to the complete tree by (1) having all observations take the path corresponding to its left branch, and (2) constraining all child branch nodes to also not be active splits.

We assume without loss of generality that all feature components are numeric and belong to the interval $[0, 1]$. Note that categorical features can be easily transformed to fit this assumption through binarization. Each decision tree split is encoded through the variables $a_t \in \{0, 1\}^p$ and $b_t \in [0, 1]$. The variable a_t indicates which feature component is involved with the split, and b_t indicates the splitting point. For example, if there are three feature components, then the split "$x_2 < 0.4''$ is encoded by $a^T x < b$ where $a = [0, 1, 0]$ and $b = 0.4$. Since decision tree splits only consider one feature component at a time, only one entry of a_t is permitted to be nonzero. Note that the quantities a_t and b_t are treated as additional decision variables in the optimization problem (2) as well as k_l and d_t.

Let $p(t)$ denote the parent node of t. Further, let $A_L(t)$ be the set of left ancestor nodes of node t, defined as the set of ancestors of t whose left branch has been followed on the path from the root node to t. Define $A_R(t)$ similarly as the set of right ancestor nodes of t.

The constraint $r_{il} \in \mathcal{T}$ in optimization problem (2) may be replaced with the set of linear and integer constraints below developed by Bertsimas & Dunn (2017) to encode the splitting logic of decision trees:

$$\sum_{l=1}^{L} r_{il} = 1, \quad \forall i \in \{1, 2, ..., n\} \tag{8a}$$

$$r_{il} \leq k_l, \quad \forall i \in \{1, 2, ..., n\}, l \in \mathcal{T}_L \tag{8b}$$

$$\sum_{i=1}^{n} r_{il} \geq N_{min} k_l, \quad \forall l \in \mathcal{T}_L \tag{8c}$$

$$a_m^T x_i \geq b_m - (1 - r_{il}), \quad \forall l \in \mathcal{T}_L, i \in \{1, 2, ..., n\}, m \in A_R(l) \tag{8d}$$

$$a_m^T (x_i + \epsilon) \leq b_m + (1 + \epsilon_{max})(1 - r_{il}), \quad \forall l \in \mathcal{T}_L, i \in \{1, 2, ..., n\}, m \in A_L(l) \tag{8e}$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \quad \forall t \in T_B \tag{8f}$$

$$1 - d_t \leq b_t \leq 1, \quad \forall t \in T_B \tag{8g}$$

$$d_t \leq d_{p(t)}, \quad \forall t \in T_B / \{1\} \tag{8h}$$

$$a_{jt}, d_t \in \{0, 1\}, \quad \forall j \in \{1...p\}, t \in T_B \tag{8i}$$

$$r_{il}, k_l \in \{0, 1\}, \quad \forall i \in \{1...n\}, l \in \mathcal{T}_L \tag{8j}$$

Above, $\epsilon_j = \left\{ x_j^{(q+1)} - x_j^{(q)} | x_j^{(q+1)} \neq x_j^{(q)} \quad 1 = 1, 2, \ldots n-1 \right\}$ is the smallest nonzero difference between observed values of feature component j, where $x_j^{(q)}$ is the q^{th} largest value observed for feature x_j and $\epsilon_{max} = \max_j \epsilon_j$. We encourage the reader to consult Bertsimas & Dunn (2017) for intuition regarding ϵ and its role in the constraints.

In Bertsimas & Dunn (2017), if a branch node is considered to be inactive, then its associated split parameters a and b are set to the zero vector and zero, respectively. This design choice was intended by the authors to force all training observations down the right branch by making the left split direction constraint (8e) infeasible for all training observations. However, we believe that this logic was implemented incorrectly, as *both* constraints (8d) and (8e) are feasible for any training observations when a and b are both zero. We have corrected for this behavior by modifying constraint (8g) to set b equal to *one* when a branch node is inactive, therefore successfully making constraint (8d) infeasible when a is the zero vector and forcing observations down the left branch.

D. SPOT Integer Programming Approach: Additional Implementation Details

To prevent unnecessarily large trees and overfitting, Bertsimas & Dunn (2017) recommend adding the quantity "$\alpha \sum_{t \in T_B} d_t$" to the objective function to penalize trees with a large number of active splits. The parameter α is intended to be chosen by the practitioner to balance the trade-off between concise trees and low training set error, and this parameter can be tuned through applying methods such as cross-validation. However, cross-validation might not be feasible in situations where solving the optimization problem is too computationally expensive to be performed for multiple values of α across multiple folds. In our numerical experiments, we train the SPO Trees with no regularization and instead apply the well-known CART post-pruning algorithm (using SPO loss) proposed by Breiman et al. (1984) to regularize the tree. To avoid lengthy technical details, we refer the reader to Breiman et al. (1984) for more information about the pruning algorithm.

Finally, we detail a few strategies for improving the computational time associated with solving the mixed integer linear program. First, as noted in Section 4.2 of the main paper, we recommend warm starting the MILP with the solution recovered from the greedy algorithm. Second, we have observed that the computational time is influenced by the precision of the vector of constants ϵ. Since the magnitude of ϵ is tied to the smallest (nonzero) differences between feature values, we recommend rounding the features according to a certain precision (e.g., $1e^{-2}$) in settings where feature rounding would not affect the quality of the resulting decision tree. Finally, we have observed that the linear programming (LP) relaxation of the MILP often has large negative solutions, which can slow down MILP solvers which rely on LP relaxations to bound the objective function (e.g., branch and bound). We recommend including the following constraint to ensure that the LP relaxation associated with the MILP has at least a lower objective function bound of zero:

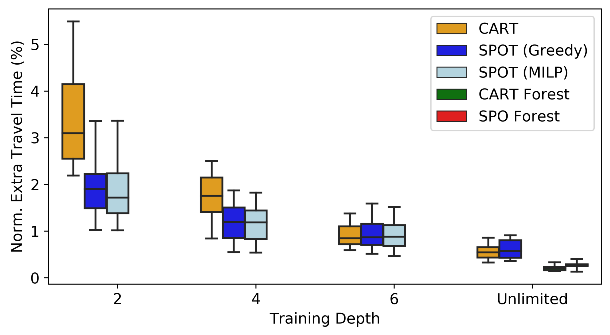
$$\left(\sum_{l=1}^{L} y_{il}\right) - z^*(c_i) \geq 0 \quad \forall i \in \{1, 2, ..., n\}$$

E. Additional Experimental Details: Noisy Shortest Path

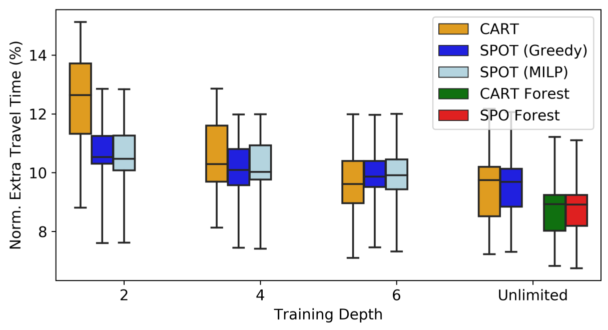
Here we investigate the decision performance of the algorithms on the shortest path problem when trained on larger datasets of $n = 10000$ observations. Since there are more training observations available, it is now feasible to train the decision tree algorithms to higher depths than in the previous experiment. Therefore, we train and evaluate the algorithms on depth sizes up to 6, and we also report the performance of SPOT and CART when trained without any depth restrictions. We also increase the level of noise from $\bar{\varepsilon} = 0.25$ to $\bar{\varepsilon} = 0.5$ to make the estimation problem more challenging for the algorithms given the increased amount of data.

The test set normalized extra travel times incurred by the algorithms for $n = 10000$ are given in Figure 5. As in the previous set of experiments, we observe that the SPO Trees achieve stronger empirical performance over CART when the training depths are restricted to small or modest values, with SPOTs attaining both better average performance and lower variance in performance across the 10 experimental trials. However, when the training depths increase to six or more, CART begins to achieve comparable performance to SPOT and even slightly outperforms SPOT in some cases. Although individual CART splits have little value for decision-making, in combination they finely partition the feature space to a sufficient degree that the predicted cost vectors are highly accurate within each of the resulting leaves. Therefore, CART is eventually able to achieve highly accurate predictions – and therefore near-optimal decisions – as its depth increases. However, its interpretabilty is sacrificed as a result, as the trees eventually grow to a size which is too large to be easily visualized and interpreted.

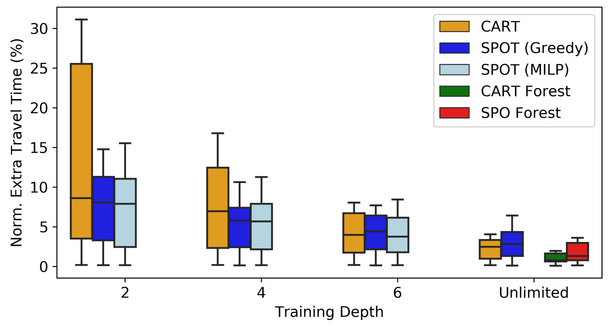
Figure 6 reports the number of leaves contained within the learned CART and SPOT trees as a function of their training depths. As the figure demonstrates, when the training depths of CART and SPOT are large or unrestricted, the SPO Trees contain less than half the number of leaf nodes as CART. Therefore, SPO Trees achieve comparable accuracy to CART in these settings *while also* being more concise and therefore more interpretable. We find that the random forest algorithms achieve similar performance, with CART random forests having a very slight edge over SPO Forests in the normalized



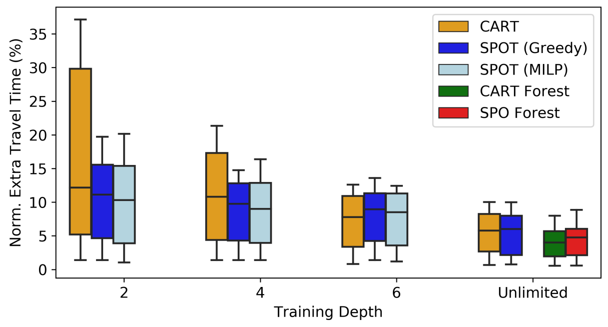
(a) $deg = 2, \bar{\varepsilon} = 0$



(b) $deg = 2, \bar{\varepsilon} = 0.5$

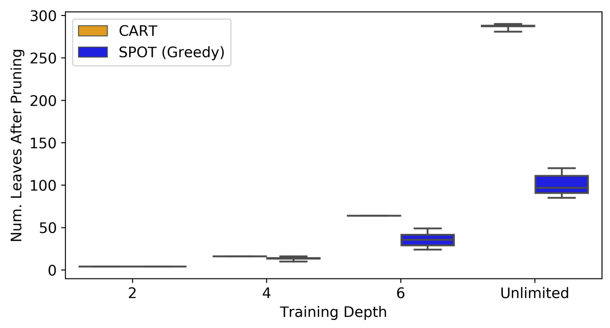


(c) $deg = 10, \bar{\varepsilon} = 0$

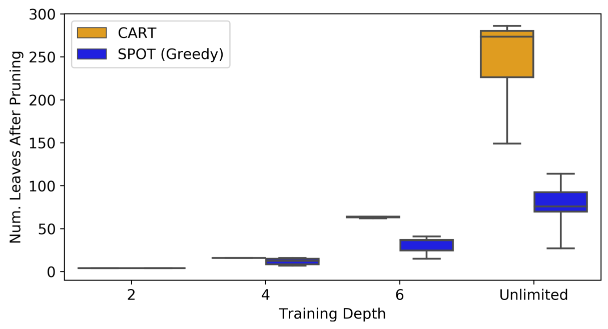


(d) $deg = 10, \bar{\varepsilon} = 0.5$

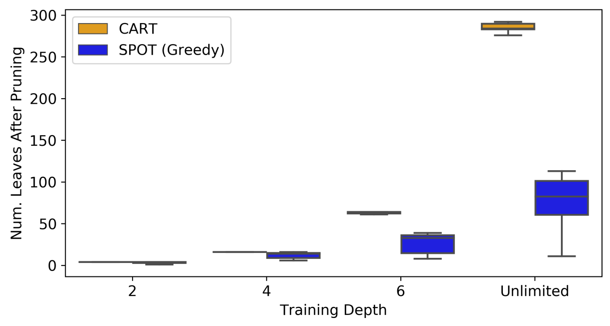
Figure 5. Test set normalized extra travel times on 10 different shortest path datasets of size $n = 10000$.



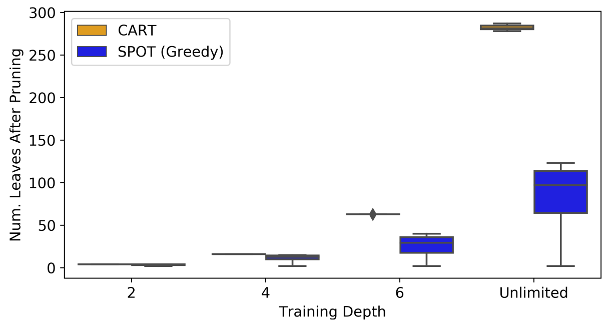
(a) $deg = 2, \bar{\varepsilon} = 0$



(b) $deg = 2, \bar{\varepsilon} = 0.5$



(c) $deg = 10, \bar{\varepsilon} = 0$



(d) $deg = 10, \bar{\varepsilon} = 0.5$

Figure 6. Number of leaves contained within the SPOT and CART trees from Figure 5. Each boxplot visualizes the number of leaves associated with the trained trees from 10 different shortest path datasets of size $n = 10000$.

extra travel times observed on the test set. The greedy SPOT approach also appears to perform similarly to the MILP approach.

F. Additional Experimental Details: News Article Recommendation

First, we provide a more thorough description of how we preprocessed the Yahoo! Front Page Today Module dataset. The dataset contains 45,811,883 interaction records between users and news articles from May 1, 2009 to May 10th, 2009. Each record entry consists of: a feature vector of dimension 5 that characterizes the visiting user, a feature vector of dimension 5 encoding the article displayed to the user, and finally a binary scalar representing whether the user clicked on the displayed article. The user and article features were constructed using a conjoint analysis with a bilinear model; see Chu et al. (2009) for more details. We preprocessed the dataset according to the following procedure in order to obtain training, validation, and test sets of feature-cost pairs for use in our predict-then-optimize problem.

1. Randomly sample without replacement 50% of the interaction records from May 1, 2009 to May 5, 2009 for training, and use the rest for validation. The test data consists of all records from May 6, 2009 to May 10, 2009.

2. Cluster users into 10,000 clusters ("user types") by applying the K-means algorithm to the user features observed in the training and validation data, and similarly cluster the displayed articles into 7 clusters ("article types") using the article features. For each user cluster, record the mean user feature vector associated with all training and validation set interaction records that map to that cluster.

3. Apply the following procedure separately to the training, validation, and test sets of interaction records. For each set of data, group the interaction records according to user type using the clustering obtained in the previous step. Each of these user types corresponds to a feature-cost pair (x, p) for the predict-then-optimize problem. The features x are derived by looking up the mean user feature vector associated with the given cluster computed in the previous step. The costs p are derived by computing the average click probability of each article type across the interaction records associated with the given cluster. Here, we note that we dropped one article type as well as a number of feature-cost pairs in the training, validation, and test sets to ensure the average click probabilities for each user and article type were calculated with at least 50 interaction records. We were left with 6 article types and 5130, 5105, and 8768 feature-cost pairs in the training, validation, and test sets, respectively.

We also note the empirical runtimes of our algorithms on this dataset. The greedy SPO Trees were trained on a Dell PowerEdge M915 Linux server using 1 processor core and 1 GB of memory per tree. The greedy SPOT training procedure (using unrestricted depth) terminated after at most 1.3 hours for each constraint set, yielding trees of depths between 28 and 38 before pruning (after pruning, the trees had an average depth of 7). SPO Forests were trained on the same server parallelizing fitting trees in the forest across 10 cores and using 40 GBs of memory. The SPO Forests training procedure terminated after at most 18.4 hours of computational time per constraint set.