Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Stochastic Optimization Forests

Nathan Kallus, Xiaojie Mao

To cite this article:
Nathan Kallus, Xiaojie Mao (2022) Stochastic Optimization Forests. Management Science

Published online in Articles in Advance 28 Jun 2022

. https://doi.org/10.1287/mnsc.2022.4458

http://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Articles in Advance, pp. 1–20
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

Stochastic Optimization Forests

Nathan Kallus,[a] **Xiaojie Mao**[b,*]

[a] Cornell Tech, Cornell University, New York, New York 10044; [b] School of Economics and Management, Tsinghua University, Beijing 100084, China
*Corresponding author

Contact: kallus@cornell.edu, https://orcid.org/0000-0003-1672-0507 (NK); maoxj@sem.tsinghua.edu.cn, https://orcid.org/0000-0003-2985-1741 (XM)

Received: September 28, 2020
Revised: November 5, 2021
Accepted: January 18, 2022
Published Online in Articles in Advance: June 28, 2022

https://doi.org/10.1287/mnsc.2022.4458

Abstract. We study contextual stochastic optimization problems, where we leverage rich auxiliary observations (e.g., product characteristics) to improve decision making with uncertain variables (e.g., demand). We show how to train forest decision policies for this problem by growing trees that choose splits to directly optimize the downstream decision quality rather than split to improve prediction accuracy as in the standard random forest algorithm. We realize this seemingly computationally intractable problem by developing approximate splitting criteria that use optimization perturbation analysis to eschew burdensome reoptimization for every candidate split, so that our method scales to large-scale problems. We prove that our splitting criteria consistently approximate the true risk and that our method achieves asymptotic optimality. We extensively validate our method empirically, demonstrating the value of optimization-aware construction of forests and the success of our efficient approximations. We show that our approximate splitting criteria can reduce running time hundredfold while achieving performance close to forest algorithms that exactly reoptimize for every candidate split.

History: Accepted by Hamid Nazerzadeh, data science.
Funding: This work was supported by the National Science Foundation [Grant 1846210].
Supplemental Material: The data files and online appendices are available at https://doi.org/10.1287/mnsc.2022.4458.

Keywords: contextual stochastic optimization • decision making under uncertainty with side observations • random forests • perturbation analysis

1. Introduction

In this paper we consider the contextual stochastic optimization (CSO) problem:

$$z^*(x) \in \arg\min_{z\in\mathcal{Z}} \mathbb{E}[c(z;Y)|X=x], \tag{1}$$

$$\mathcal{Z} = \left\{ z \in \mathbb{R}^d \ : \ \begin{array}{ll} h_k(z) = 0, & k = 1,\dots,s, \\ h_k(z) \le 0, & k = s+1,\dots,m \end{array} \right\}, \tag{2}$$

wherein, having observed contextual features $X = x \in \mathcal{X} \subseteq \mathbb{R}^p$, we seek a decision $z \in \mathcal{Z}$ to minimize average costs, which are impacted by a yet-unrealized uncertain variable $Y \in \mathcal{Y}$. Equation (1) is essentially a stochastic optimization problem (Shapiro et al. 2014) where the distribution of the uncertain variable is given by the *conditional* distribution of $Y|X = x$. Crucially, this corresponds to using the observations of features $X = x$ to best possibly control total average costs over new realizations of pairs (X, Y); that is,

$$\mathbb{E}[c(z^*(X);Y)] = \min_{z(x):\mathbb{R}^p\to\mathcal{Z}} \mathbb{E}[c(z(X);Y)].$$

Stochastic optimization can model many managerial decision-making problems in inventory management (Simchi-Levi et al. 2005), revenue management (Talluri and Van Ryzin 2006), finance (Cornuejols and Tütüncü 2006), and other application domains (Kleywegt and Shapiro 2001, Shapiro et al. 2014). CSO in particular captures the interplay of such decision models with the availability of rich side observations of other variables (i.e., covariates X) often present in modern data sets, which can help significantly reduce uncertainty and improve performance compared with *un*conditional stochastic optimization (Bertsimas and Kallus 2020).

Because the exact joint distribution of (X, Y), which specifies the CSO in Equation (1), is generally unavailable, we are in particular interested in learning a well-performing policy $\hat{z}(x)$ based on n independent and identically distributed (i.i.d.) draws from the joint distribution of (X, Y):

$$\text{Data}: \ \mathcal{D} = \{(X_1,Y_1),\dots,(X_n,Y_n)\}, \ (X_i,Y_i) \sim (X,Y) \text{ i.i.d.}$$

The covariates X may be any that can help predict the value of the uncertain variable Y affecting costs so that we can reduce uncertainty and improve performance. A common approach is to first make predictions using models that are trained without consideration of the downstream decision-making problem and then solve optimization given their plugged-in predictions. However, this

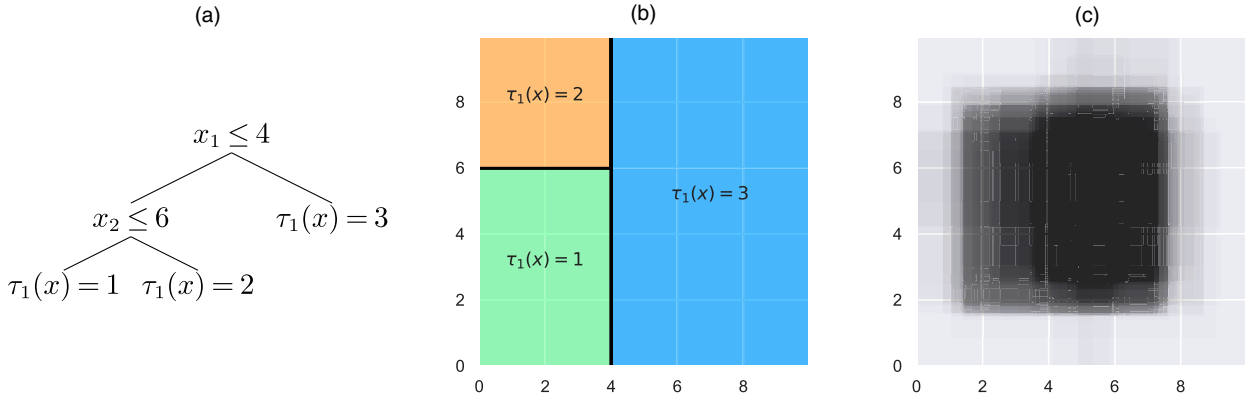
approach completely *separates* prediction and optimization. Because all predictive models make errors, especially when learning a complex object such as the conditional distribution of Y given X, the error tradeoffs of this approach may be undesirable for the end task of decision-making. In this paper, we aim to *learn effective forest-based CSO policies that integrate prediction and optimization*.To make a decision at a new query point x, a forest policy uses a *forest* $\mathcal{F} = \{\tau_1, \ldots, \tau_T\}$ of *trees* τ_j to reweight the sample to emphasize data points i with covariates X_i "close" to x. Each tree, $\tau_j : \mathbb{R}^p \to \{1, \ldots, L_j\}$, is a partition of $\mathbb{R}^p$ into L_j regions, where the function τ_j takes the form of a binary tree with internal nodes splitting on the value of a component of x (Figure 1, (a) and (b)). We then reweight each data point i in the sample by the frequency $w_i(x)$ with which X_i ends up in the same region (tree leaf) as x, over trees in the forest (Figure 1(c)). Using these weights, we solve a weighted sample analogue of Equation (1). That is, a forest policy has the following form, where the forest $\mathcal{F}$ constitutes the parameters of the policy $\hat{z}(x)$:

$$\hat{z}(x) \in \arg\min_{z \in \mathcal{Z}} \sum_{i=1}^{n} w_i(x) c(z; Y_i),$$
$$w_i(x) := \frac{1}{T} \sum_{j=1}^{T} \frac{\mathbb{I}\left[\tau_j(X_i) = \tau_j(x)\right]}{\sum_{i'=1}^{n} \mathbb{I}\left[\tau_j(X_{i'}) = \tau_j(x)\right]}. \tag{3}$$

Bertsimas and Kallus (2020) considered using a forest policy where the forest $\mathcal{F}$ is given by running the random forest (RandForest) algorithm (Breiman 2001). The RandForest algorithm, however, builds trees that target the prediction problem of learning $\mathbb{E}[Y \mid X = x]$, rather than the CSO problem in Equation (1). Namely, it builds each tree τ_j by, starting with all of $\mathbb{R}^p$, recursively subpartitioning each region $R_0 \subseteq \mathbb{R}^p$ into the two subregions $R_0 = R_1 \cup R_2$ that minimize the sum of squared distance to the mean of data in each subregion (i.e., $\sum_{j=1,2} \min_{z \in \mathbb{R}^d} \sum_{i: X_i \in R_j} \|z - Y_i\|_2^2$). For prediction, random forests are notable for adeptly handling high-dimensional feature data nonparametrically as they only split on variables relevant to prediction, especially compared with other methods for generating localized weights $w_i(x)$ like k-nearest neighbors and Nadaraya-Watson kernel regression. However, for CSO, they might miss signals more relevant to the particular optimization structure in Equation (1), deteriorating downstream policy performance in the actual decision-making problem. Athey et al. (2019) proposed a generalized random forest (GenRandForest) algorithm to estimate roots of conditional estimating equations, which can be repurposed for *unconstrained* CSO problems by solving their first order optimality conditions. Their splitting criteria are based on approximating the *mean squared errors* of equation root estimates, which again may fail to capture signals more important for the particular cost function in Equation (1) when optimization is one's aim.

In this paper, we design new algorithms to construct decision trees and forests that directly target the CSO problem in Equation (1). Specifically, we choose tree splits to optimize the cost of resulting decisions instead of standard impurity measures (e.g., sum of squared errors), thereby incorporating the general cost function $c(z; Y)$ and constraints $\mathcal{Z}$ into the tree construction. A similar idea was suggested in endnote 2 of Bertsimas and Kallus (2020) but is dismissed because it would be too computationally cumbersome to use this to evaluate many candidate splits in each node of each tree in a forest. In this paper, we solve this task in a computationally efficient manner by leveraging a second-order perturbation analysis of

Figure 1. (Color online) A Forest of Trees, $\mathcal{F} = \{\tau_1, \ldots, \tau_T\}$, Parameterizes a Forest Policy $\hat{z}(x)$ for CSO as in Equation (3)



Notes. (a) A depth-3 tree. When the condition in a branching node holds, we take the left branch. (b) Each tree gives a partition of $\mathbb{R}^d$, where each region corresponds to a leaf of the tree. (c) Darker regions fall into the same region as $x = (0, 0)$ for more trees in a forest.

stochastic optimization, resulting in efficient and effective large-scale forests tailored to the decision-making problem of interest that lead to strong performance gains in practice.

Our contributions are as follows. We formalize the oracle splitting criterion for recursively partitioning trees to target the CSO problem and then use second-order perturbation analysis to show how to approximate the intractable oracle splitting criterion by extrapolating from the given region, R_0, to the candidate subregions, R_1, R_2, provided that the CSO problem is sufficiently smooth. We do this in Section 2 for the unconstrained setting and in Section 3 for the constrained setting. Specifically, we consider both an approach that extrapolates the optimal value and an approach that extrapolates the optimal solution. Crucially, our perturbation approach means that we only must solve a stochastic optimization problem at the root region, R_0, and then we can efficiently extrapolate to what will happen to average costs for any candidate subpartition of the root, allowing us to efficiently consider many candidate splits. Using these new efficient approximate splitting criteria, we develop the stochastic optimization tree (StochOptTree) algorithm, which we then use to develop the stochastic optimization forest (StochOptForest) algorithm by running the former many times. The StochOptForest algorithm fits forests to *directly* target the downstream decision-making problem of interest and then uses these forests to construct effective forest policies for CSO. In Section 4, we empirically demonstrate the success of our StochOptForest algorithm and the value of forests constructed to directly consider the downstream decision-making problem. In Section 5, we provide asymptotic optimality results for StochOptForest. In Section 6, we offer a discussion of and comparison with related literature, and in Section 7, we offer some concluding remarks. We extend our results to *stochastically* constrained CSO problems in Online Appendix A, develop variable-importance measures in Online Appendix B, and provide additional empirical results in Online Appendix C. We defer all proofs to Online Appendix H.

1.1. Running Examples of CSOs

We will have a few running examples of CSOs.

Example 1 (Multi-Item Newsvendor). In the multi-item newsvendor problem, we must choose the order quantities for d products, $z = (z_1, \dots, z_d)$, before we observe the random demand for each of these, $Y = (Y_1, \dots, Y_d)$, to control holding and backorder costs. Whenever the order quantity for product l exceeds the demand for the product, we pay a holding cost of α_l per unit. Whenever the demand exceeds the order quantity, we pay a backorder cost of β_l per unit. The total cost is

$$c(z;y) = \sum_{l=1}^{d} \max\{\alpha_l(z_l - y_l), \beta_l(y_l - z_l)\}. \tag{4}$$

Negating and adding a constant, we can also consider this equivalently as the sale revenue up to the smaller of z_l and y_l, minus ordering costs for z_l units. The order quantities may be unrestricted (in which case the d problems decouple). They may be restricted by a capacity constraint,

$$\mathcal{Z} = \left\{ z \in \mathbb{R}^d : \sum_{l=1}^{d} z_l \le C,\ z_l \ge 0,\ l = 1, \dots, d \right\},$$

where C is a constant that stands for the inventory capacity limit.

Covariates X in this problem may be any that can help predict future demand. For example, for predicting demand for home video products, Bertsimas and Kallus (2020) use data from Google search trends, data from online ratings, and past sales data.

Example 2 (Variance-Based Portfolio Optimization). Consider d assets with random future returns $Y = (Y_1, \dots, Y_d)$, and decision variables $z = (z_1, \dots, z_d)$ that represent the fraction of investment in each asset in a portfolio of investments, constrained to be in the simplex $\Delta^d = \{z \in \mathbb{R}^d : \sum_{l=1}^{d} z_l = 1, z_l \ge 0, l = 1, \dots, d\}$. Then the return of the portfolio is $Y^\top z$. We want the portfolio $z(x)$ to minimize the variance of the return given $X = x$. This can be formulated as a CSO by introducing an additional unconstrained auxiliary optimization variable $z_{d+1} \in \mathbb{R}$ and letting

$$c(z;y) = \left(y^\top z_{1:d} - z_{d+1}\right)^2. \tag{5}$$

We can either let $\mathcal{Z} = \Delta^d \times \mathbb{R}$ or relax nonnegativity constraints to allow short selling.

More generally we may consider optimizing a linear combination of the conditional mean and variance of the return, which corresponds to a CSO with the following cost function:

$$c(z;y) = \left(y^\top z_{1:d} - z_{d+1}\right)^2 - \rho y^\top z_{1:d},\ \rho > 0. \tag{6}$$

Covariates X in this problem may be any that can help predict future returns. Examples include past returns, stock fundamentals, economic fundamentals, news stories, and so on.

Example 3 (CVaR-Based Portfolio Optimization). When the asset return distributions are not elliptically symmetric, conditional value-at-risk (CVaR) may be a more suitable risk measure than variance (Rockafellar and Uryasev 2000). We may therefore prefer to consider minimizing the CVaR at level α given $X = x$, defined as

$$\mathrm{CVaR}_\alpha(Y^\top z \mid X = x) = \min_{w \in \mathbb{R}} \mathbb{E}\left[\frac{1}{\alpha} \max\{w - Y^\top z, 0\} - w \,\middle|\, X = x \right].$$

This again can be formulated as a CSO by introducing an additional unconstrained auxiliary optimization

variable $z_{d+1} \in \mathbb{R}$ and letting

$$c(z;y) = \frac{1}{\alpha}\max\{z_{d+1} - y^\top z_{1:d}, 0\} - z_{d+1}. \tag{7}$$

We can analogously incorporate the simplex constraint or relax the nonnegativity constraint as in Example 2. We can also optimize a weighted combination of the different criteria (mean, variance, CVaR at any level); we need only introduce a separate auxiliary variable for variance and for CVaR at each level considered.

Example 4 (Prediction of Conditional Expectation). Although the previous examples provide actual decision-making problems, the problem of prediction also fits into the CSO framework as a special case. Namely, if $Y \in \mathbb{R}^d$, $c(z;y) = \frac{1}{2}\|z-y\|_2^2$, and $\mathcal{Z} = \mathbb{R}^d$ is unconstrained, then we can see that $z^*(x) = \mathbb{E}[Y|X=x]$. This can be understood as the best-possible (in squared error) prediction of Y in a draw of (X, Y) where only X is revealed. Fitting forest models to predict $\mathbb{E}[Y|X=x]$ is precisely the target task of random forests, which use squared error as a splitting criterion. We further compare with other literature on *estimation* using random forests in Section 6.1. A key aspect of handling general CSOs, as we do, is dealing with general cost functions and constraints and targeting the expected cost of our decision rather than the error in estimating $z^*(x)$.

2. Unconstrained Case

We begin by studying the unconstrained case as it is simpler and therefore more instructive. Throughout this section, we let $\mathcal{Z} = \mathbb{R}^d$. We extend to the more general constrained case in Section 3. To develop our StochOptForest algorithm, we start by considering the StochOptTree algorithm, which we will then run many times to create our forest. To motivate our StochOptTree algorithm, we will first consider an idealized splitting rule for an idealized policy, then consider approximating it using perturbation analysis, and then consider estimating the approximation using data. Each of these steps constitutes one of the next sections.

2.1. Oracle Splitting Rule

Given a partition, $\tau : \mathbb{R}^p \to \{1,\dots,L\}$, of $\mathbb{R}^p$ into L regions, consider the policy $z_\tau(x) \in \arg\min_{z\in\mathcal{Z}}\mathbb{E}[c(z;Y)\mathbb{I}[\tau(X)=\tau(x)]]$ that, for each x, optimizes costs only for (X, Y) where X falls in the same region as x. This policy is hypothetical and not implementable in practice given just the data as it involves the *true* joint distribution of (X, Y). We wish to learn a partition τ described by a binary decision tree with nodes of the form "$x_j \le \theta$?" such that it leads to a well-performing policy $z_\tau(x)$, that is, has small risk $\mathbb{E}[c(z_\tau(X);Y)]$. Finding the best τ over all trees of a given depth is generally a very hard problem, even if we knew the distributions involved. To simplify it, suppose we fix a partition τ and we wish only to refine it slightly by taking one of its regions, say $R_0 = \tau^{-1}(L)$, and choosing some $j \in \{1,\dots,p\}$, $\theta \in \mathbb{R}$ to construct a new partition τ' with $\tau'(x) = \tau(x)$ for $x \notin R_0$, $\tau'(x) = L$ for $x \in R_1 = R_0 \cap \{x \in \mathbb{R}^p : x_j \le \theta\}$, and $\tau'(x) = L+1$ for $x \in R_2 = R_0 \cap \{x \in \mathbb{R}^p : x_j > \theta\}$. That is, we further subpartition the region R_0 into the subregions R_1, R_2. We would then be interested in finding the choice of (j,θ), leading to minimal risk, $\mathbb{E}[c(z_{\tau'}(X);Y)] = \mathbb{E}[c(z_{\tau'}(X);Y)\mathbb{I}[X \notin R_0]] + \mathbb{E}[c(z_{\tau'}(X);Y)\mathbb{I}[X \in R_1]] + \mathbb{E}[c(z_{\tau'}(X);Y)\mathbb{I}[X \in R_2]]$. Notice that the first term is constant in the choice of the subpartition and only the second and third terms matter in choosing the subpartition. We should therefore seek the subpartition that leads to the minimal value of

$$\begin{aligned}\mathcal{C}^{\text{oracle}}(R_1,R_2) &= \sum_{j=1,2}\mathbb{E}\big[c(z_{\tau'}(X);Y)\mathbb{I}\big[X \in R_j\big]\big] \\ &= \sum_{j=1,2}\min_{z\in\mathcal{Z}}\mathbb{E}\big[c(z;Y)\mathbb{I}\big[X \in R_j\big]\big],\end{aligned} \tag{8}$$

where the last equality holds because the tree policy $z_{\tau'}$ makes the best decision within each region of the new partition. We call this the *oracle splitting criterion*. Searching over choices of (j,θ) in some given set of possible options, the best refinement of τ is given by the choice minimizing this criterion. If we start with the trivial partition, $\tau(x) = 1\ \forall x$, then we can recursively refine it using this procedure to grow a tree of any desired depth. When $c(z;y) = \frac{1}{2}\|z-y\|_2^2$ and the criterion is estimated by replacing expectations with empirical averages, this is precisely the regression tree algorithm of Breiman et al. (1984), in which case the estimated criterion is easy to compute as it is simply given by rescaled within-region variances of Y_i. For general $c(z;y)$, however, computing the criterion involves solving a general stochastic optimization problem that may have no easy analytical solution (even if we approximate expectations with empirical averages), and it is therefore hard to do quickly for many, many possible candidates for (j,θ), and correspondingly it would be hard to grow large forests of many of these trees.

2.2. Perturbation Analysis of the Oracle Splitting Criterion

Consider a region $R_0 \subseteq \mathbb{R}^p$ and its candidate subpartition $R_0 = R_1 \cup R_2$, $R_1 \cap R_2 = \emptyset$. Let

$$v_j(t) = \min_{z\in\mathcal{Z}} f_0(z) + t(f_j(z) - f_0(z)),$$
$$z_j(t) \in \arg\min_{z\in\mathcal{Z}} f_0(z) + t(f_j(z) - f_0(z)),$$
$$\text{where } f_j(z) = \mathbb{E}\big[c(z;Y)\,|\,X \in R_j\big], \quad j = 0,1,2,\ t \in [0,1]. \tag{9}$$

The optimization objective function in Equation (9) is obtained from perturbing the objective function $f_0(z)$ in the region R_0 toward the objective function $f_j(z)$ in a subregion R_j for $j = 1, 2$. The perturbation magnitude is quantified by the parameter $t \in [0,1]$. Note that the optimal values of fully perturbed problems (i.e., $t = 1$) in two subregions determine the oracle splitting criterion:

$$\mathcal{C}^{\text{oracle}}(R_1,R_2) = p_1 v_1(1) + p_2 v_2(1),$$
$$\text{where} \quad p_j = \mathbb{P}(X \in R_j), \tag{10}$$

and ideally, we would use these values to evaluate the quality of the subpartition. However, we would rather not have to solve the stochastic optimization problem involved in Equation (9) at $t = 1$ *repeatedly* for every candidate subpartition. Instead, we would rather solve the *single* problem $v_1(0) = v_2(0)$, that is, the problem corresponding to the region R_0, and try to extrapolate from there what happens as we take $t \to 1$, that is, the limiting problem corresponding to each subregion R_j for each candidate split. To solve this, we consider the perturbation of the problem $v_j(t)$ at $t = 0$ as we increase it infinitesimally and use this to approximate $v_j(1)$. As long as the distribution of $Y|X \in R_0$ is not too different from that of $Y|X \in R_j$, this would be a reasonable approximation.

First, we note that a first-order perturbation analysis would be insufficient. We can show that under appropriate continuity conditions and if $\arg\min_{z\in\mathcal{Z}} f_0(z) = \{z_0\}$ is a singleton, we would have $v_j(t) = (1-t)f_0(z_0) + tf_j(z_0) + o(t)$.[1] We could use this to approximate $v_j(1) \approx f_j(z_0)$ by plugging in $t = 1$ and ignoring the higher-order terms, which makes intuitive sense: if we only perturb the objective slightly, the optimal solution is approximately unchanged and we only need to evaluate its new objective value. This would lead to the approximate splitting criterion $p_1v_1(1) + p_2v_2(1) \approx p_1f_1(z_0) + p_2f_2(z_0)$. However, because $p_1f_1(z_0) + p_2f_2(z_0) = p_0f_0(z_0)$, this is ultimately unhelpful as it does not at all depend on the choice of subpartition.

Instead, we must conduct a finer, second-order perturbation analysis to understand the effect of the choice of subpartition on risk. The next result does this for the unconstrained case.

Theorem 1 (Second-Order Perturbation Analysis: Unconstrained). *Fix $j = 1, 2$. Suppose the following conditions hold:*

1. *The functions $f_0(z)$ and $f_j(z)$ are twice continuously differentiable;*
2. *The inf-compactness condition: there exist constants α and $t_0 \in (0,1]$ such that the sublevel sets $\{z \in \mathbb{R}^d : f_0(z) + t(f_j(z) - f_0(z)) \le \alpha\}$ are nonempty and uniformly bounded for $t \in [0, t_0)$;*
3. *The function $f_0(z)$ has a unique minimizer z_0 over $\mathbb{R}^d$, and $\nabla^2 f_0(z_0)$ is positive definite;*

Then

$$v_j(t) = (1-t)f_0(z_0) + tf_j(z_0) - \frac{1}{2}t^2\nabla f_j(z_0)^\top\big(\nabla^2 f_0(z_0)\big)^{-1}\nabla f_j(z_0) + o(t^2), \tag{11}$$
$$z_j(t) = z_0 - t\big(\nabla^2 f_0(z_0)\big)^{-1}\nabla f_j(z_0) + o(t). \tag{12}$$

Theorem 1 gives the second-order expansion of the optimal value $v_j(t)$ and the first-order expansion of *any* choice of $z_j(t)$ that attains $v_j(t)$ around $t = 0$. These expansions quantify the impact on the optimal value and optimal solution when infinitesimally perturbing the objective function in region R_0 toward that in a subregion R_j. One crucial condition of Theorem 1 is that the objective functions are sufficiently smooth (condition 1). This condition holds for any subpartition if we assume that $\mathbb{E}[c(z;Y)|X]$ is almost surely twice continuously differentiable, which is trivially satisfied if the cost function $c(z;Y)$ is almost surely twice continuously differentiable function of z. However, even if $c(z;Y)$ is nonsmooth (e.g., Examples 1 and 2), $\mathbb{E}[c(z;Y)|X]$ may still be sufficiently smooth if the distribution of $Y|X$ is continuous (see examples in Section 2.3). In particular, one reason we defined the oracle splitting criterion using the population expectation rather than empirical averages is that for many relevant examples such as newsvendor and CVaR only the population objective may be smooth while the sample objective may be nonsmooth and therefore not amenable to perturbation analysis.

Condition 2 ensures that if we only slightly perturb the objective function f_0, optimal solutions of the resulting perturbed problem are always bounded and never escape to infinity. This means that without loss of generality we can restrict our attention to a compact subset of $\mathbb{R}^d$. This compactness condition and the smoothness condition (condition 1) together ensure the existence of optimal solutions for any optimization problem corresponding to $t \in [0, t_0)$. In addition, this condition is crucial for ensuring $z(t) \to z_0$ as $t \to 0$ (Bonnans and Shapiro 2000, proposition 4.4). One sufficient condition for this is that any optimal solution $z^*(X)$ in Equation (1) is almost surely bounded, for example, when conditional quantiles of all item demands in Example 1 are almost surely bounded. Finally, the regularity condition (condition 3) is obviously satisfied if $f_0(z)$ is strictly convex,

which is implied if either $\mathbb{E}[c(z;Y)|X]$ or $c(z;Y)$ is almost surely strictly convex. Condition 3 may be satisfied even if the cost function $c(z;Y)$ is not strictly convex: for example, it holds for the newsvendor problem (Example 1) when the density of $Y_l|X \in R_0$ is positive at z_0 for all $l = 1,\dots,d$.

2.3. Approximate Splitting Criteria

Theorem 1 suggests two possible approximations of the oracle splitting criterion.

2.3.1. Approximate Risk Criterion. If we use Equation (11) to extrapolate to $t = 1$, ignoring the higher-order terms, we arrive at

$$v_j(1) \approx f_j(z_0) - \frac{1}{2}\nabla f_j(z_0)^\top (\nabla^2 f_0(z_0))^{-1}\nabla f_j(z_0).$$

Taking a weighted average of this over $j = 1, 2$, we arrive at an approximation of the oracle splitting criterion $\mathcal{C}^{\text{oracle}}$ in Equation (8). Because $p_1 f_1(z_0) + p_2 f_2(z_0) = p_0 f_0(z_0)$ is constant in the subpartition, we may ignore these terms, leading to the following criterion:

$$\mathcal{C}^{\text{apx-risk}}(R_1,R_2) = -\frac{1}{2}\sum_{j=1,2} p_j \nabla f_j(z_0)^\top (\nabla^2 f_0(z_0))^{-1}\nabla f_j(z_0). \tag{13}$$

By strengthening the conditions in Theorem 1, we can in fact show that this approximation becomes arbitrarily accurate as the partition becomes finer.

Theorem 2. *Suppose the following conditions hold for both* $j = 1, 2$:

1. *Condition* 1 *of Theorem* 1.
2. *Condition* 2 *of Theorem* 1 *holds for all* $t \in [0,1]$.
3. *The function* $f_0(z) + t(f_j(z) - f_0(z))$ *has a unique minimizer* z_0 *and* $\nabla^2(f_0(z) + t(f_j(z) - f_0(z)))$ *is positive definite at this unique minimizer for all* $t \in [0,1]$.
4. *The function* $\mathbb{E}[c(z;Y)|X = x]$ *is twice Lipschitz-continuously differentiable in* x.

Then

$$|\mathcal{C}^{\text{oracle}}(R_1,R_2) - p_0 f_0(z_0) - \mathcal{C}^{\text{apx-risk}}(R_1,R_2)| = o(\mathcal{D}_0^2),$$

where $\mathcal{D}_0 = \sup_{x,x' \in R_0}\|x - x'\|_2$ *is the diameter of* R_0.

Again, note that $p_0 f_0(z_0)$ is constant in the choice of subpartition.

2.3.2. Approximate Solution Criterion. Because $v_j(1) = f_j(z_j(1))$, we can also approximate $v_j(1)$ by approximating $z_j(1)$ and plugging it in. Using Equation (12) to extrapolate $z_j(t)$ to $t = 1$ and ignoring the higher-order terms, we arrive at the following approximate criterion:

$$\mathcal{C}^{\text{apx-soln}}(R_1,R_2) = \sum_{j=1,2} p_j f_j(z_0 - (\nabla^2 f_0(z_0))^{-1}\nabla f_j(z_0)). \tag{14}$$

This almost looks like applying a Newton update to z_0 in the $\min_z f_j(z)$ problem, namely, the solution that optimizes the second-order expansion of $f_j(z)$ at z_0. However, a naive Newton update will require to invert the Hessian for f_j, which varies across different candidate splits. In contrast, the criterion $\mathcal{C}^{\text{apx-soln}}$ requires the Hessian for f_0, meaning we only have to invert a Hessian once for all candidate subpartitions.

For unconstrained CSO problems in this section, we may also apply the GenRandForest algorithm in Athey et al. (2019) to solve their first-order optimality condition. The GenRandForest algorithm uses a similar way to approximate optimal solutions in split subregions. It chooses splits to maximize the difference between approximate solutions in two subregions induced by each candidate split, as their proposition 1 shows that this approximately minimizes the total mean squared errors of the resulting estimated optimal solutions. In contrast, by using $\mathcal{C}^{\text{apx-soln}}$, we choose splits to minimize the expected cost of the approximate optimal solutions, thereby directly targeting the ultimate objective in CSO problems. More importantly, we tackle the constrained case (Section 3), whereas the GenRandForest algorithm cannot. In Section 4 and Online Appendix C.1, we show the impact of both of these differences can be significant in practice when optimization is the aim.

In the following theorem, we show that the approximate solution criterion also becomes arbitrarily accurate as the partition becomes finer.

Theorem 3. *Suppose the assumptions of Theorem* 2 *hold. Then*

$$|\mathcal{C}^{\text{oracle}}(R_1,R_2) - \mathcal{C}^{\text{apx-soln}}(R_1,R_2)| = o(\mathcal{D}_0^2).$$

2.3.3. Revisiting the Running Examples. The previous approximate criteria crucially depend on the gradients $\nabla f_1(z_0), \nabla f_2(z_0)$ and Hessian $\nabla^2 f_0(z_0)$. We next study these quantities for some examples.

Example 5 (Derivatives with Smooth Cost). If $c(z;y)$ is itself twice continuously differentiable for every y, then under regularity conditions that enable the exchange of derivative and expectation (e.g., $|(\nabla c(z;Y))_\ell| \le W$ for all z in a neighborhood of z_0 with $\mathbb{E}[W \mid X_i \in R_j] < \infty$), we have $\nabla f_j(z_0) = \mathbb{E}\big[\nabla c(z_0;Y) \mid X_i \in R_j\big]$ and $\nabla^2 f_0(z_0) = \mathbb{E}\big[\nabla^2 c(z_0;Y) \mid X_i \in R_0\big]$.

Example 1, Continued (Derivatives in Multi-Item Newsvendor). In many cases, $c(z;y)$ is not smooth, as in the example of the multi-item newsvendor cost in Equation (4). In this case, it suffices that the distribution of $Y \mid X \in R_j$ is continuous for gradients and Hessians

to exist. Then, we can show that $(\nabla f_j(z_0))_l = (\alpha_l + \beta_l)\,\mathbb{P}(Y_l \le z_{0,l} \mid X \in R_j) - \beta_l$ and $(\nabla^2 f_0(z_0))_{ll} = (\alpha_l + \beta_l)\mu_{0,l}(z_0)$ for $l = 1, \ldots, d$, $j = 1, 2$, and $(\nabla^2 f_0(z_0))_{ll'} = 0$ for $l \neq l'$, where $\mu_{0,l}$ is the density function of $Y_l | X \in R_0$. Therefore, $\nabla^2 f_0(z_0)$ is invertible as long as $\mu_{0,l}(z_0) > 0$ for $l = 1, \ldots, d$.

Example 2, Continued (Derivatives in Variance-Based Portfolio Optimization). The cost function in Equation (5) is an instance of Example 5 (smooth costs). Using block notation to separate the first d decision variables from the final single auxiliary variable, we verify in Proposition EC.8 that

$$\nabla f_j(z_0) = 2\begin{bmatrix} \mathbb{E}[YY^\top | X \in R_j] z_{0,1:d} - \mathbb{E}[Y \mid X \in R_j] z_{0,d+1} \\ z_{0,1:d}^\top (\mathbb{E}[Y \mid X \in R_0] - \mathbb{E}[Y \mid X \in R_j]) \end{bmatrix}, \tag{15}$$

$$\nabla^2 f_0(z_0) = 2\begin{bmatrix} \mathbb{E}[YY^\top | X \in R_0] & -\mathbb{E}[Y \mid X \in R_0] \\ -\mathbb{E}[Y^\top | X \in R_0] & 1 \end{bmatrix}. \tag{16}$$

Notice $\nabla^2 f_0(z_0)$ is invertible if and only if the covariance matrix $\mathrm{Var}(Y \mid X \in R_0)$ is invertible.

Example 3, Continued (Derivatives in CVaR-Based Portfolio Optimization). Like the newsvendor cost in Equation (4), the CVaR cost in Equation (7) is not smooth either. Again, we assume that the distribution of $Y \mid X \in R_j$ is continuous. Then, when $z_0 \neq 0$ (Proposition EC.9 in Online Appendix G),

$$\nabla f_j(z_0) = \frac{1}{\alpha}\begin{bmatrix} -\mathbb{E}\left[Y \mathbb{I}\left[\overline{Y}_0 \le q_0^\alpha(\overline{Y}_0)\right] | X \in R_j\right] \\ \mathbb{P}\left(q_0^\alpha(\overline{Y}_0) - \overline{Y}_0 \ge 0 | X \in R_j\right) - \alpha \end{bmatrix},$$
$$\overline{Y}_0 := Y^\top z_{0,1:d}, \tag{17}$$

$$\nabla^2 f_0(z_0) = \frac{\mu_0\left(q_0^\alpha(\overline{Y}_0)\right)}{\alpha} \begin{bmatrix} \mathbb{E}\left[YY^\top | \overline{Y}_0 = q_0^\alpha(\overline{Y}_0), X \in R_0\right] & -\mathbb{E}\left[Y | \overline{Y}_0 = q_0^\alpha(\overline{Y}_0), X \in R_0\right] \\ -\mathbb{E}\left[Y^\top | \overline{Y}_0 = q_0^\alpha(\overline{Y}_0), X \in R_0\right] & 1 \end{bmatrix}, \tag{18}$$

where μ_0 is the density function of $\overline{Y}_0$ given $X \in R_0$, and $q_0^\alpha(\overline{Y}_0)$ is the α-level quantile of $\overline{Y}_0$ given $X \in R_0$. The Hessian matrix $\nabla^2 f_0(z_0)$ may not necessarily be invertible. This arises because of the homogeneity of returns in scaling the portfolio, so that second derivatives in this direction may vanish. This issue is corrected when we consider the constrained case where we fix the scale of the portfolio (see Section 3).[2]

2.3.4. Reoptimizing Auxiliary Variables.

In Examples 2 and 3, z contains both auxiliary variables and decision variables, and we construct the approximate criteria based on gradients and Hessian matrix with respect to both sets of variables. A natural alternative is to reoptimize the auxiliary variables first so that the objective only depends on decision variables and then evaluate the corresponding gradients and Hessian matrix. That is, if we partition $z = (z^{\mathrm{dec}}, z^{\mathrm{aux}})$, then we can redefine $f_j(z^{\mathrm{dec}}) = \min_{z^{\mathrm{aux}}} \mathbb{E}\left[c((z^{\mathrm{dec}}, z^{\mathrm{aux}}); Y) | X \in R_j\right]$ and $z_0^{\mathrm{dec}} = \arg\min_{z^{\mathrm{dec}}} f_0(z^{\mathrm{dec}})$. The perturbation analysis remains largely the same by simply using the gradients and Hessian matrix for the redefined $f_j(z^{\mathrm{dec}})$ at z_0^{dec}. This leads to an alternative approximate splitting criterion. However, evaluating the gradients $\nabla f_j(z_0^{\mathrm{dec}})$ for $j = 1, 2$ would now involve repeatedly finding the optimal solution $\arg\min_{z^{\mathrm{aux}}} \mathbb{E}\left[c((z_0^{\mathrm{dec}}, z^{\mathrm{aux}}); Y) | X \in R_j\right]$ for all candidate splits. See Online Appendix E for details.

Because the point of our approximate criteria is to avoid reoptimization for every candidate split, this alternative is practically relevant only when reoptimizing the auxiliary variables is very computationally easy. For example, in Example 2, z^{dec} corresponds to the first d variables and reoptimizing the auxiliary $(d+1)$ th variable amounts to computing the mean of $Y^\top z_0^{\mathrm{dec}}$ in each subregion, which can be done quite efficiently as we vary the candidate splits.

2.4. Estimating the Approximate Splitting Criteria

The benefit of our approximate splitting criteria, $\mathcal{C}^{\text{apx-soln}}(R_1, R_2)$, $\mathcal{C}^{\text{apx-risk}}(R_1, R_2)$ in Equations (13) and (14), is that they only involve the solution of z_0. Thus, if we want to evaluate many different subpartitions of R_0, we need only solve for z_0 once, compute $(\nabla^2 f_0(z_0))^{-1}$ once, and then only recompute $\nabla f_j(z_0)$ for each new subpartition. However, this involves quantities we do not actually know because all of these depend on the joint distribution of (X, Y). We therefore next consider the estimation of these approximate splitting criteria from data.

Given estimators $\hat{H}_0, \hat{h}_1, \hat{h}_0$ of $\nabla^2 f_0(z_0)$, $\nabla f_1(z_0)$, $\nabla f_2(z_0)$, respectively (see following examples), we can construct the estimated approximate splitting criteria as

$$\hat{\mathcal{C}}^{\text{apx-risk}}(R_1, R_2) = -\sum_{j=1,2} \frac{n_j}{n} \hat{h}_j^\top \hat{H}_0^{-1} \hat{h}_j, \tag{19}$$

$$\hat{\mathcal{C}}^{\text{apx-soln}}(R_1, R_2) = \sum_{j=1,2} \frac{1}{n} \sum_{i=1}^n \mathbb{I}[X_i \in R_j] c\left(\hat{z}_0 - \hat{H}_0^{-1} \hat{h}_j; Y_i\right), \tag{20}$$

where $n_j = \sum_{i=1}^n \mathbb{I}[X_i \in R_j]$.

Under appropriate convergence of $\hat{H}_0, \hat{h}_1, \hat{h}_2$, these estimated criteria, respectively, converge to the population approximate criteria $\mathcal{C}^{\text{apx-risk}}(R_1, R_2)$ and $\mathcal{C}^{\text{apx-soln}}(R_1, R_2)$ in Section 2.3, as summarized by the following self-evident proposition.

Proposition 1. *If* $\|\hat{H}_0^{-1} - (\nabla^2 f_0(z_0))^{-1}\|_{\mathrm{F}} = o_p(1)$, $\|\hat{h}_j - \nabla f_j(z_0)\|_2 = O_p(n^{-1/2})$ *for* $j = 1, 2$, *then*

$$\hat{\mathcal{C}}^{apx\text{-}risk}(R_1,R_2)=\mathcal{C}^{apx\text{-}risk}(R_1,R_2)+O_p(n^{-1/2}).$$

If also $\left|\frac{1}{n}\sum_{i=1}^n \mathbb{I}[X_i\in R_j]c(\hat{z}_0-\hat{H}_0^{-1}\hat{h}_j;Y_i)-p_jf_j(z_0-(\nabla^2 f_0(z_0))^{-1}\nabla f_j(z_0))\right|=O_p(n^{-1/2})$ *for* $j=1,2$, *then*

$$\hat{\mathcal{C}}^{apx\text{-}soln}(R_1,R_2)=\mathcal{C}^{apx\text{-}soln}(R_1,R_2)+O_p(n^{-1/2}).$$

If we can find estimators that satisfy the conditions of Proposition 1, then together with Theorems 2 and 3, we will have shown that the estimated approximate splitting criteria can well approximate the oracle splitting criterion when samples are large, and the partition is fine. It remains to find appropriate estimators.

2.4.1. General Estimation Strategy. Because the gradients and Hessian to be estimated are evaluated at a point z_0 that is itself unknown, a general strategy is to first estimate z_0 and then estimate the gradients and Hessian at this estimate. This is the strategy we follow in the examples here.

Specifically, we can first estimate z_0 by its sample analogue:

$$\hat{z}_0\in\arg\min_{z\in\mathcal{Z}}\widehat{p_0f_0}(z),$$

$$\text{where } \widehat{p_0f_0}(z):=\frac{1}{n}\sum_{i=1}^n \mathbb{I}[X_i\in R_0]c(z;Y_i). \tag{21}$$

Under standard regularity conditions, the previously estimated optimal solution $\hat{z}_0$ is consistent (see Lemma EC.1 in Online Appendix G). Then, given generic estimators $\hat{H}_0(z)$ of $\nabla^2 f_0(z)$ at any one z and similarly estimators $\hat{h}_j(z)$ of $\nabla f_j(z)$ for $j=1,2$, we let $\hat{H}_0=\hat{H}_0(\hat{z}_0)$ and $\hat{h}_j=\hat{h}_j(\hat{z}_0)$. Examples of this follow.

2.4.2. Revisiting the Running Examples. We next discuss examples of possible estimates $\hat{H}_0$, $\hat{h}_j$ that can be proved to satisfy the conditions in Proposition 1 (see Propositions EC.10 to EC.13 in Online Appendix G for details). All our examples use the previous general estimation strategy.

Example 5, Continued (Estimation with Smooth Cost). If $c(z;y)$ is itself twice continuously differentiable in z for every y, we can simply use $\hat{H}_0(\hat{z}_0)=\frac{1}{n_0}\sum_{i=1}^n \mathbb{I}[X_i\in R_0]\nabla^2 c(\hat{z}_0;Y_i)$ and $\hat{h}_j(\hat{z}_0)=\frac{1}{n_j}\sum_{i=1}^n \mathbb{I}[X_i\in R_j]\nabla c(\hat{z}_0;Y_i)$. In Proposition EC.13 in Online Appendix G, we show that these satisfy the conditions of Theorem 1 because of the smoothness of $c(z;y)$. Example 2 is one example of this case, which we discuss later. Example 4 is another example.

In particular, for the squared error cost function in Example 4 ($c(z;y)=\frac{1}{2}\|z-y\|_2^2$), we show in Proposition EC.7 in Online Appendix G that, using the previous $\hat{H}_0,\hat{h}_j$, we have

$$\frac{1}{n}\sum_{i=1}^n \mathbb{I}[X_i\in R_0]c(\hat{z}_0;Y_i)+\hat{\mathcal{C}}^{\text{apx-risk}}(R_1,R_2)$$

$$=\hat{\mathcal{C}}^{\text{apx-soln}}(R_1,R_2)=\sum_{j=1,2}\frac{n_j}{2n}\sum_{l=1}^d \text{Var}(\{Y_{i,l}:X_i\in R_j\}),$$

which is exactly the splitting criterion used for regression by random forests, namely the sum of squared errors to the mean within each subregion. Notice the very first term is constant in R_1, R_2.

Example 1, Continued (Estimation in Multi-Item Newsvendor). In the previous section, we saw that the gradient and Hessian depend on the cumulative distribution and density functions, respectively. We can therefore estimate the gradients by $\hat{h}_{j,\ell}(\hat{z}_0)=\frac{\alpha_l+\beta_l}{n_j}\sum_{i=1}^n \mathbb{I}[X_i\in R_j, Y_l\le\hat{z}_{0,l}]-\beta_l$, and the Hessian using, for example, kernel density estimation: $\hat{H}_{0,ll}(\hat{z}_0)=\frac{\alpha_l+\beta_l}{n_jb}\sum_{i=1}^n \mathbb{I}[X_i\in R_0]\mathcal{K}((Y_{i,l}-\hat{z}_{0,l})/b)$, where $\mathcal{K}$ is a kernel such as $\mathcal{K}(u)=\mathbb{I}\left[|u|\le\frac{1}{2}\right]$ and b is the bandwidth, and $\hat{H}_{0,ll'}(\hat{z}_0)=0$ for $l\neq l'$. We show the validity of these estimates in Proposition EC.10 in Online Appendix G.

Example 2, Continued (Estimation in Variance-Based Portfolio Optimization). With $\hat{z}_0=\{\hat{z}_{0,1},\dots,\hat{z}_{0,d},\hat{z}_{0,d+1}\}$ given by solving the problem in Equation (21), the gradient and Hessian in Equations (15) and (16) can be estimated by their sample analogues:

$$\hat{h}_j(\hat{z}_0)=2\begin{bmatrix}\frac{1}{n_j}\sum_{i=1}^n \mathbb{I}[X_i\in R_j]Y_iY_i^\top\hat{z}_{0,1:d}-\frac{1}{n_j}\sum_{i=1}^n \mathbb{I}[X_i\in R_j]Y_i\hat{z}_{0,d+1}\\ \hat{z}_{0,1:d}^\top\left(\frac{1}{n_0}\sum_{i=1}^n \mathbb{I}[X_i\in R_0]Y_i-\frac{1}{n_j}\sum_{i=1}^n \mathbb{I}[X_i\in R_j]Y_i\right)\end{bmatrix},$$

$$\hat{H}_0(\hat{z}_0)=2\begin{bmatrix}\frac{1}{n_0}\sum_{i=1}^n \mathbb{I}[X_i\in R_0]Y_iY_i^\top & -\frac{1}{n_0}\sum_{i=1}^n \mathbb{I}[X_i\in R_0]Y_i\\ -\frac{1}{n_0}\sum_{i=1}^n \mathbb{I}[X_i\in R_0]Y_i^\top & 1\end{bmatrix}.$$

These estimators are in fact specific examples of the general smooth case in Example 5, so they too can be analyzed by Proposition EC.13 in Online Appendix G.

Example 3, Continued (Estimation in CVaR-Based Portfolio Optimization). It is straightforward to estimate the gradient in Equation (17):

$$\hat{h}_j(\hat{z}_0)=\frac{1}{\alpha}\begin{bmatrix}-\frac{1}{n_j}\sum_{i=1}^n \mathbb{I}[Y_i^\top\hat{z}_{0,1:d}\le\hat{q}_0^\alpha(Y^\top\hat{z}_{0,1:d}),X_i\in R_j]Y_i\\ \frac{1}{n_j}\sum_{i=1}^n \mathbb{I}\left[Y_i^\top\hat{z}_{0,1:d}\le\hat{q}_0^\alpha(Y^\top\hat{z}_{0,1:d}),X_i\in R_j\right]-\alpha\end{bmatrix}, \tag{22}$$

where $\hat{q}_0^\alpha(Y^\top\hat{z}_{0,1:d})$ is the empirical α-level quantile of $Y^\top\hat{z}_{0,1:d}$ based on data in R_0. The Hessian matrix in Equation (18) is more challenging to estimate because

it involves many conditional expectations given the event $Y^\top z_{0,1:d} = q_0^\alpha(Y^\top z_{0,1:d})$. In principle, we could estimate these nonparametrically by, for example, kernel smoothing estimators (Fan and Yao 1998, Yin et al. 2010, Chen and Leng 2015, Loubes et al. 2020). For simplicity and because this is only used as an approximate splitting criterion anyway, in our empirics we can consider a parametric approach instead, which we will use in our empirics in Section 4.1: if $Y \mid X \in R_0$ has a Gaussian distribution $\mathcal{N}(m_0, \Sigma_0)$, then

$$\begin{aligned}&\mathbb{E}\left[Y \mid Y^\top z_{0,1:d} = q_0^\alpha(Y^\top z_{0,1:d}), X \in R_0\right]\\ &\quad = m_0 + \Sigma_0 z_{0,1:d}\left(z_{0,1:d}^\top \Sigma_0 z_{0,1:d}\right)^{-1}(q_0^\alpha(Y^\top z_{0,1:d}) - m_0^\top z_{0,1:d}),\end{aligned} \tag{23}$$

$$\begin{aligned}&\mathrm{Var}\left(Y \mid Y^\top z_{0,1:d} = q_0^\alpha(Y^\top z_{0,1:d}), X \in R_0\right)\\ &\quad = \Sigma_0 - \Sigma_0 z_{0,1:d}\left(z_{0,1:d}^\top \Sigma_0 z_{0,1:d}\right)^{-1} z_{0,1:d}^\top \Sigma_0,\end{aligned} \tag{24}$$

and $\mathbb{E}\left[YY^\top \mid Y^\top z_{0,1:d} = q_0^\alpha(Y^\top z_{0,1:d}), X \in R_0\right]$ can be directly derived from these two quantities. We can then estimate these quantities by plugging in $\hat{z}_0$ for z_0, the empirical mean estimator of Y for m_0, the empirical variance estimator of Y for Σ_0, and the empirical α-level quantile of $Y^\top \hat{z}_{0,1:d}$ for $q_0^\alpha(Y^\top z_{0,1:d})$, all based only on the data in R_0. Finally, we can estimate $\mu_0(q_0^\alpha(Y^\top z_{0,1:d}))$ by a kernel density estimator $\frac{1}{n_0 b}\sum_{i=1}^n \mathbb{I}[X_i \in R_0]\mathcal{K}\left(\left(Y_i^\top \hat{z}_{0,1:d} - \hat{q}_0^\alpha(Y^\top \hat{z}_{0,1:d})\right)/b\right)$. Although the Gaussian distribution may be misspecified, the resulting estimator is more stable than and easier to implement than nonparametric estimators (especially considering that it will be used repeatedly in tree construction), and it can still approximate the relative scale of entries in the Hessian matrix reasonably well. In Section 4.1, we empirically show that our method based on these approximate estimates works well even if the Gaussian model is misspecified. If it happens to be correctly specified, we can also theoretically validate that the estimator satisfies the conditions of Theorem 1 (see Proposition EC.12 in Online Appendix G).

2.5. Stochastic Optimization Tree and Forest Algorithms

With the estimated approximate splitting criteria in hand, we can now describe our StochOptTree and StochOptForest algorithms. Specifically, we will first describe how we use our estimate approximate splitting criteria to build trees, which we will then combine to make a forest that leads to a CSO decision policy $\hat{z}(x)$ as in Equation (3).

Algorithm 1 (Recursive Procedure to Grow a StochOptTree (Unconstrained Case))

procedure StochOptTree.Fit(region R_0, data $\mathcal{D}$, `depth`, `id`)
  $\hat{z}_0 \leftarrow$ Minimize$(\sum_{(X_i,Y_i)\in\mathcal{D}} \mathbb{I}[X_i \in R_0]c(z;Y_i), z \in \mathcal{Z})$ ▷ Solve Equation (21)
  $\hat{H}_0 \leftarrow$ Estimate $\nabla^2 f_0(z)$ at $z = \hat{z}_0$
  `CandSplit` $\leftarrow$ GenerateCandidateSplits(R_0, $\mathcal{D}$) ▷ Create the set of possible splits
  $\hat{C} \leftarrow \infty$
  for $(j, \theta) \in$ `CandSplit` **do** ▷ Optimize the estimated approximate criterion
    $(R_1, R_2) \leftarrow (R_0 \cap \{x \in \mathbb{R}^p : x_j \le \theta\}, R_0 \cap \{x \in \mathbb{R}^p : x_j > \theta\})$
    $(\hat{h}_1, \hat{h}_2) \leftarrow$ Estimate $\nabla f_1(z)$, $\nabla f_2(z)$ at $z = \hat{z}_0$
    $C \leftarrow \hat{\mathcal{C}}^{\text{apx-risk/apx-soln}}(R_1, R_2)$ ▷ Compute the criterion using $\hat{H}_0, \hat{h}_1, \hat{h}_2, \mathcal{D}$
    if $C < \hat{C}$ **then** $(\hat{C}, \hat{j}, \hat{\theta}) \leftarrow (C, j, \theta)$
  if Stop?$((\hat{j}, \hat{\theta}), R_0, \mathcal{D},$ `depth`$)$ **then**
    return $(x \longmapsto$ `id`$)$
  else
    `LeftSubtree`$\leftarrow$StochOptTree.Fit $(R_0 \cap \{x \in \mathbb{R}^p : x_j \le \theta\}, \mathcal{D},$ `depth` $+ 1,$ `2id`$)$
    `RightSubtree` $\leftarrow$StochOptTree.Fit$(R_0 \cap \{x \in \mathbb{R}^p : x_j > \theta\}, \mathcal{D},$ `depth` $+ 1,$ `2id` $+ 1)$
    return $(x \longmapsto x_j \le \theta$? `LeftSubtree`(x) : `RightSubtree`$(x))$

2.5.1. StochOptTree Algorithm.

We summarize the tree construction procedure in Algorithm 1. We will extend Algorithm 1 to the constrained case in Section 3. This procedure partitions a generic region, R_0, into two children subregions, R_1 and R_2, by an axis-aligned cut along a certain coordinate of covariates. It starts with solving the optimization problem within R_0 according to Equation (21) and then finds the best split coordinate $\hat{j}$ and cutoff value $\hat{\theta}$ over a set of candidate splits by minimizing[3] the estimated approximate risk criterion in Equation (19) or the estimated approximate solution criterion in Equation (20). Once the best split $(\hat{j}, \hat{\theta})$ is found, R_0 is partitioned into the two subregions accordingly, and the whole procedure continues on recursively until a stopping criterion.

There are a few subroutines to be specified. First, there is the optimization of $\hat{z}_0$. Depending on the structure of the problem, different algorithms may be appropriate. For example, if $c(z;y)$ is the maximum of several linear functions, a linear programming solver may be used. More generally, because the objective has the form of a sum of functions, methods such as stochastic gradient descent (aka stochastic approximation; Nemirovski et al. 2009) may be used. Second, there is the estimation of $\hat{H}_0, \hat{h}_1, \hat{h}_2$, which was discussed in Section 2.4. Third, we need to generate a set of candidate splits, which can be done in different ways. The original RandForest algorithm (Breiman 2001) randomly selects a prespecified number of distinct coordinates j from $\{1, \ldots, p\}$ without replacement, and considers θ to be all midpoints in the $X_{i,j}$ data, which exhausts all possible subpartitions along each selected coordinate. Another option is to consider a random subset of cutoff values, possibly enforcing that the sample sizes of the corresponding

two children nodes are balanced, as in Denil et al. (2014). This approach not only enforces balanced splits, which is important for statistical guarantees (see Theorem 5), but it also reduces the computation time. Finally, we need to decide when to stop the tree construction. A typical stopping criterion is when each child region reaches a prespecified number of data points (Breiman 2001). Depth may also additionally be restricted. In an actual implementation, if the stopping criterion would have stopped regardless of the split chosen, we can short circuit the call and skip the split optimization.

Notice that $\hat{z}_0$ and $\hat{H}_0$ need only be computed once for each recursive call to StochOptTree.Fit, whereas $\hat{h}_1, \hat{h}_2$ need to be computed for each candidate split. All estimators $\hat{h}_j$ discussed in Section 2.4 take the form of a sample average over $i \in R_j$, for $j = 1, 2$, and therefore can be easily and quickly computed for each candidate split. Moreover, when candidate cutoff values consist of all midpoints of the sample values in the j th coordinate, such sample averages can be efficiently updated by proceeding in sorted order, where only one datapoint changes from one side of the split to the other at a time, similarly to how the original random forest algorithm maintains within-subpartition averages of outcomes and their squares for each candidate split.

Notably, the tree construction computation is typically dominated by the step of searching best splits. This step can be implemented very efficiently with our approximate criteria because they only involve estimation of gradients and simple linear algebra operations (Section 2.3). Only one optimization and Hessian computation is needed at the beginning of each recursive call. In particular, we do not need to solve optimization problems repeatedly for each candidate split, which is the central aspect of our approach and which enables the construction of large-scale forests.

Algorithm 2 (Procedure to Fit a StochOptForest)

procedure StochOptForest.Fit(data $\mathcal{D}$, number of trees T)
  for $j = 1$ to T **do**
    $\mathcal{I}_j^{\text{tree}}, \mathcal{I}_j^{\text{dec}} \leftarrow$ Subsample($\{1, \dots, |\mathcal{D}|\}$)
    $\tau_j \leftarrow$StochOptTree.Fit($\mathcal{X}$, $\{(X_i, Y_i) \in \mathcal{D} : i \in \mathcal{I}_j^{\text{tree}}\}$, 1, 1)
      ▷ Fit tree using the subdataset $\mathcal{I}_j^{\text{tree}}$
  return $\{(\tau_j, \mathcal{I}_j^{\text{dec}}) : j = 1, \dots, T\}$

Algorithm 3 (Procedure to Make a Decision Using StochOptForest)

procedure StochOptForest.Decide(data $\mathcal{D}$, forest $\{(\tau_j, \mathcal{I}_j^{\text{dec}}) : j = 1, \dots, T\}$, target x)
  $w(x) \leftarrow$ Zeros($|\mathcal{D}|$)
    ▷ Create an all-zero vector of length $|\mathcal{D}|$
  for $j = 1, \dots, T$ **do**
    $\mathcal{N}(x) \leftarrow \{i \in \mathcal{I}_j^{\text{dec}} : \tau_j(X_i) = \tau_j(x)\}$
      ▷ Find the τ_j-neighbors of x among the data in $\mathcal{I}_j^{\text{dec}}$
    for $i \in \mathcal{N}(x)$ **do** $w_i(x) \leftarrow w_i(x) + \frac{1}{|\mathcal{N}(x)|\,T}$
      ▷ Update the sample weights
  return Minimize($\sum_{(X_i,Y_i)\in\mathcal{D}} w_i(x) c(z; Y_i)$, $z \in \mathcal{Z}$)
    ▷ Compute the forest policy Equation (3)

2.5.2. StochOptForest Algorithm. In Algorithm 2, we summarize the algorithm of building forests using trees constructed by Algorithm 1. It involves an unspecified subsampling subroutine. For each $j = 1, \dots, T$, we consider possibly subsampling the data on which we will fit the j th tree ($\mathcal{I}^{\text{tree}}$) as well as the data that we will later use to generate localized weights for decision making ($\mathcal{I}^{\text{dec}}$). There are different possible ways to construct these subsamples. Following the original random forest algorithm, we may set $\mathcal{I}_j^{\text{tree}} = \mathcal{I}_j^{\text{dec}}$ equal to a bootstrap sample (a sample of size n with replacement). Alternatively, we may set $\mathcal{I}_j^{\text{tree}} = \mathcal{I}_j^{\text{dec}}$ to be sampled as a fraction of n *without* replacement, which is an approach adopted in more recent random forest literature as it is more amenable to theoretical analysis and has similar empirical performance (Scornet et al. 2015, Mentch and Hooker 2016). Alternatively, we may also *sequentially* sample $\mathcal{I}_j^{\text{tree}}, \mathcal{I}_j^{\text{dec}}$ without replacement so the two are disjoint (e.g., take a random half of the data and then further split it at random into two). The property that the two sets are disjoint, $\mathcal{I}_j^{\text{tree}} \cap \mathcal{I}_j^{\text{dec}} = \emptyset$, is known as *honesty*, and it is helpful in proving statistical consistency of random forests (Denil et al. 2014, Wager and Athey 2018, Athey et al. 2019).[4]

2.5.3. Final Decision. In Algorithm 3, we summarize the algorithm of making a decision at new query points x once we have fit a forest, that is, compute the forest policy, Equation (3). Although the tree algorithm we developed thus far, Algorithm 1, is for the unconstrained case, we present Algorithm 3 in the general constrained case. In a slight generalization of Equation (3), we actually allow the data weighted by each tree to be a subset of the whole data set (i.e., $\mathcal{I}_j^{\text{dec}}$), as described previously. Namely, the weights $w_i(x)$ computed by Algorithm 3 are given by

$$w_i(x) = \frac{1}{T}\sum_{j=1}^{T} \frac{\mathbb{I}\left[i \in \mathcal{I}_j^{\text{dec}}, \tau_j(X_i) = \tau_j(x)\right]}{\sum_{i'=1}^{n} \mathbb{I}\left[i \in \mathcal{I}_j^{\text{dec}}, \tau_j(X_{i'}) = \tau_j(x)\right]}, \tag{25}$$

which is slightly more general than Equation (3). Algorithm 3 then optimizes the average cost over the data with sample weights given by $w_i(x)$. Under honest splitting, for each single tree, each data point is used in either placing splits or constructing weights but not both. However, because each tree uses an independent random subsample, every data point will participate in the construction of some trees and also the computation of weights by other trees. Therefore, all observations contribute to both forest

construction and the weights in the final decision making. In this sense, despite appearances, honest splitting is not "wasting" data.

The weights $\{w_i(x)\}_{i=1}^n$ generated by Algorithm 3 represent the average frequency with which each data point falls into the same terminal node as x. The measure given by the sum over i of $w_i(x)$ times the Dirac measure at Y_i can be understood as an estimate for the conditional distribution of $Y \mid X = x$. However, in contrast to nonadaptive weights such as given by k-nearest neighbors or Nadaraya-Watson kernel regression (Bertsimas and Kallus 2020), which nonparametrically estimate this conditional distributional generically, our weights *directly* target the optimization problem of interest, focusing on the aspect of the data that is relevant to the optimization problem, which makes our weights much more efficient. Moreover, in contrast to using weights given by standard random forests, which targets prediction with minimal squared error, our weights target the *right* downstream optimization problem.

3. Constrained Case

In this section, we develop approximate splitting criteria for training forests for general CSO problems with constraints as described at the onset in Equation (1). Namely, in this section, we let $\mathcal{Z} = \{z \in \mathbb{R}^d : h_k(z) = 0, k = 1, \ldots, s,\ h_k(z) \le 0, k = s+1, \ldots, m\}$ be as in Equation (2). The oracle criterion we target remains $C^{\text{oracle}}(R_1, R_2)$ as in Equation (8) with the crucial difference that now $\mathcal{Z}$ need not be $\mathbb{R}^d$ and may be constrained as earlier. We then proceed as in Section 2: We approximate the oracle criterion in two ways, then we estimate the approximations, and then we use these estimated splitting criteria to construct trees. Because the perturbation analysis in the presence of constraints is somewhat more cumbersome, this section will be more technical. However, the high-level idea remains the same as the simpler unconstrained case in Section 2.

3.1. Perturbation Analysis of the Oracle Splitting Criterion

Again, consider a region $R_0 \subseteq \mathbb{R}^d$ and its candidate subpartition $R_0 = R_1 \cup R_2$, $R_1 \cap R_2 = \varnothing$. We define $v_j(t)$, $z_j(t)$, $f_j(t)$ as in Equation (9) with the crucial difference that now $\mathcal{Z}$ is constrained. The oracle criterion is given by $C^{\text{oracle}}(R_1, R_2) = p_1 v_1(1) + p_2 v_2(1)$, as before. We again approximate $v_1(1), v_2(1)$ by computing $v_1(t), v_2(t)$ at $t = 0$ (where they are equal and do not depend on the subpartition) and then extrapolating from there by leveraging second order perturbation analysis. We present our key perturbation result for this here.

Theorem 4 (Second-Order Perturbation Analysis: Constrained). *Fix $j = 1, 2$. Suppose the following conditions hold:*

1. *The functions $f_0(z), f_j(z)$ are twice continuously differentiable.*
2. *The problem corresponding to $f_0(z)$ has a unique minimizer z_0 over $\mathcal{Z}$.*
3. *The inf-compactness condition: there exist constants α and $t_0 \in (0, 1]$ such that the sublevel set $\{z \in \mathcal{Z} : f_0(z) + t(f_j(z) - f_0(z)) \le \alpha\}$ is nonempty and uniformly bounded over $t \in [0, t_0)$.*
4. *The minimizer z_0 is associated with a unique Lagrangian multiplier ν_0 that also satisfies the strict complementarity condition: $\nu_{0,k} > 0$ if $k \in K_h(z_0)$, where $K_h(z_0) = \{k : h_k(z_0) = 0, k = s+1, \cdots, m\}$ is the index set of active at z_0 inequality constraints.*
5. *The Mangasarian-Fromovitz constraint qualification condition at z_0:*

$$\begin{aligned} &\nabla h_1(z_0), \ldots, \nabla h_s(z_0) \text{ are linearly independent, and} \\ &\quad \exists d_z \text{ s.t. } \nabla h_k(z_0) d_z = 0,\ k = 1, \ldots, s, \\ &\nabla h_k(z_0) d_z < 0,\ k \in K_h(z_0). \end{aligned}$$

6. *Second-order sufficient condition:*

$$d_z^\top \left(\nabla^2 f_0(z_0) + \sum_{k=1}^m \nu_{0,k} \nabla^2 h_k(z_0) \right) d_z > 0 \qquad \forall d_z \in C(z_0) \setminus \{0\},$$

where $C(z_0)$ is the critical cone defined as follows:

$$C(z_0) = \{d_z : d_z^\top \nabla h_k(z_0) = 0, \text{ for } k \in \{1, \ldots, s\} \cup K_h(z_0)\}.$$

Let d_z^{j*} be the first part of the (unique) solution of the following linear system of equations:

$$\begin{bmatrix} \left(\nabla^2 f_0(z_0) + \sum_{k=1}^m \nu_{0,k} \nabla^2 h_k(z_0) \right) & \nabla \mathcal{H}^{K_h \top}(z_0) \\ \nabla^\top \mathcal{H}^{K_h}(z_0) & 0 \end{bmatrix} \begin{bmatrix} d_z^j \\ \xi \end{bmatrix} = \begin{bmatrix} -(\nabla f_j(z_0) - \nabla f_0(z_0)) \\ 0 \end{bmatrix}, \tag{26}$$

where $\nabla^\top \mathcal{H}(z_0) \in \mathbb{R}^{m \times d}$ is the matrix whose kth row is $(\nabla h_k(z_0))^\top$, and $\nabla^\top \mathcal{H}^{K_h}(z_0) \in \mathbb{R}^{s + |K_h(z_0)| \times d}$ consists only of the rows corresponding to equality and active inequality constraints.

Then

$$\begin{aligned} v_j(t) = {} & (1-t) f_0(z_0) + t f_j(z_0) \\ & + \frac{1}{2} t^2 \left\{ d_z^{j* \top} \left(\nabla^2 f_0(z_0) + \sum_{k=1}^m \nu_{0,k} \nabla^2 h_k(z_0) \right) d_z^{j*} \right. \\ & \left. + 2 d_z^{j* \top} \left(\nabla f_j(z_0) - \nabla f_0(z_0) \right) \right\} + o(t^2), \end{aligned} \tag{27}$$

$$z_j(t) = z_0 + t d_z^{j*} + o(t). \tag{28}$$

Because of the presence of constraints, the approximations of optimal value $v_j(t)$ and optimal solution $z_j(t)$ in Theorem 4 require more complicated conditions

than those in Theorem 1. In particular, we need to incorporate constraints in the inf-compactness conditions (condition 3) and second-order sufficient condition (condition 6), impose uniqueness and strict complementarity regularity conditions for the Lagrangian multiplier (condition 4), and assume a constraint qualification condition (condition 5). The coefficient matrix on the left-hand side of the linear system of equations in Equation (26) is invertible because of the second-order sufficient condition in condition 6 (Bertsekas 1995, proposition 4.2.2), which ensures that d_z^{j*} uniquely exists. These regularity conditions guarantee that the optimal value $v_j(t)$ and optimal solution $z_j(t)$ vary smoothly with perturbations to the optimization objective, and they rule out problems whose optimal solution may change nonsmoothly. For example, optimal solutions to linear programming problems may change to completely different vertices under even tiny perturbations to linear objectives.[5] Nevertheless, Theorem 4 may still apply to some problems with linear costs and nonlinear constraints such as the quadratically constrained problems in section 6.2 of Elmachtoub and Grigas (2022).

Concretely, the constraints in Examples 1 to 3 all ensure that the decision variables are bounded. Therefore, the inf-compactness condition (condition 3) is satisfied when there is no additional auxiliary variable (e.g., the newsvendor problem) or when the auxiliary variables at CSO optimal solutions are almost surely bounded. (e.g., conditional expectation or conditional quantiles of optimal portfolio returns in Example 2 or 3, respectively) Moreover, because these constraints are all simple affine constraints, in Online Appendix G, Proposition EC.14, we verify that they satisfy a stronger linear independence constraint qualification condition than condition 5, which ensures the unique existence of Lagrangian multiplier v_0 for the solution z_0 (condition 4). Because our problems in Examples 1 to 3 are all convex, the second-order sufficient condition (condition 6) can ensure z_0 to be the unique optimal solution (condition 2). This second-order sufficient condition trivially holds when the Hessian matrix is positive definite and the constraints are affine, for example, under the conditions we discuss in Section 2.3 for Examples 1 and 2. In contrast to these conditions, the strict complementary slackness in condition 4 is generally more difficult to verify exactly. However, even if it does not hold exactly, splitting criteria based on the approximations in Equations (27) and (28) may still capture signals relevant to CSO problems, especially compared with RandForest, which completely ignores the optimization problem structure.

Theorem 1 is a special case of Theorem 4 without constraints ($m = 0$). Indeed, without the constraints, the regularity conditions for the Lagrangian multiplier and constraint qualification condition are vacuous, and conditions 3 and 6 reduce to the inf-compactness and positive definite Hessian matrix conditions in Theorem 1, respectively. Without constraints, the linear equation system in Equation (26) consists only of the part corresponding to d_z^j, and the solution exactly coincides with the linear term in Equation (14). Theorem 4 can itself be viewed as a special case of our Theorem EC.1 in Online Appendix A, where we tackle CSO problems with both deterministic and stochastic constraints.

3.2. Approximate Splitting Criteria

Analogous to Theorem 1 for unconstrained problems, Theorem 4 for constrained problems also motivates two different approximations of the oracle splitting criterion $\mathcal{C}^{\text{oracle}}(R_1, R_2) = p_1 v_1(1) + p_2 v_2(1)$. Extrapolating Equation (27) and Equation (28) to $t = 1$ and ignoring the high-order terms gives an approximate risk and approximate solution criterion, respectively:

$$\mathcal{C}^{\text{apx-risk}}(R_1,R_2) = \frac{1}{2}\sum_{j=1,2} p_j d_z^{j*\top}\left(\nabla^2 f_0(z_0) + \sum_{k=1}^{m} v_{0,k}\nabla^2 h_k(z_0)\right) d_z^{j*} + \sum_{j=1,2} p_j d_z^{j*\top}(\nabla f_j(z_0) - \nabla f_0(z_0)), \tag{29}$$

$$\mathcal{C}^{\text{apx-soln}}(R_1,R_2) = \sum_{j=1,2} p_j f_j(z_0 + d_z^{j*}), \tag{30}$$

where in the approximate risk criterion, $\mathcal{C}^{\text{apx-risk}}(R_1, R_2)$, we again omit from the extrapolation the constant term $\sum_{j=1,2} p_j(f_j(z_0)) = p_0 f_0(z_0)$, as it does not depend on the choice of subpartition.

3.2.1. Estimating the Approximate Splitting Criteria.

We next discuss a general strategy to estimate our more general approximate splitting criteria in Equations (29) and (30) that handle constraints. First, we start by estimating z_0 by its sample analogue as in Equation (21), where crucially now $\mathcal{Z}$ is constrained. Then we can estimate the gradients of f_j at z_0 for $j = 0$, 1, 2 and the Hessians of f_0 at z_0 in the very same way that gradients of f_j and Hessians of f_0 were estimated in Section 2.3, namely, estimating them at $\hat{z}_0$, which is now simply solved *with* constraints. Gradients and Hessians of h_k at z_0 can be estimated by simply plugging in $\hat{z}_0$, because the functions h_k are known deterministic functions. We can estimate $K_h(z_0)$ by $K_h(\hat{z}_0)$, that is, the index set of the inequality constraints that are active at $\hat{z}_0$. Next, we can estimate v_0 by solving $\widehat{\nabla f}_0(\hat{z}_0) + \sum_{k=1}^{m} v_k \nabla h_k(\hat{z}_0) = 0$ subject to $v_k \geq 0$ for $k \in K_h(\hat{z}_0)$ and $v_k = 0$ for $k \in \{s+1, \ldots, m\} \backslash K_h(\hat{z}_0)$, or alternatively using a solver for Equation (21) that provides associated dual solutions. Finally, we can estimate d_z^{j*} by solving Equation (26) with estimates plugged in for

unknowns. With all these pieces in hand, we can estimate our approximate criteria in Equations (29) and (30).

3.2.2. Revisiting the Running Examples. In Section 2.4, we discussed how to estimate gradients and Hessians of the objectives of our running examples. Now we revisit the examples and discuss their constraints. The nonnegativity and capacity constraints in Example 1 can be written as $h_1(z') = \sum_{l=1}^d z_l \le C,\ h_{l+1}\ (z') = -z_l \le 0,\ \ l = 1,\dots,d$, and the simplex constraint in Examples 2 and 3 as $h_1(z) = \sum_{l=1}^d z_l = 1,\ h_{l+1}(z) = -z_l \le 0, l = 1,\dots,d$. These are all deterministic linear constraints: Their gradients are known constants and their Hessians are zero.

3.3. Construction of Trees and Forests

It is straightforward to now extend the tree fitting algorithm, Algorithm 1, to the constrained case. First, we note that in line 2 that solves for $\hat{z}_0$, we use a constrained feasible set $\mathcal{Z}$. Then, we update line 3 to estimate $\nabla f_0(z_0), \nabla^2 f_0(z_0), \nabla h_k(z_0), \nabla^2 h_k(z_0), K_h(z_0), \nu_0$. Next, we update line 8 to estimate $\nabla f_j(z_0), d_z^{j*}$. Finally, we update line 9 to use the general splitting criteria in Equations (29) and (30) where we plug in these estimates for the unknowns.

A crucial point that is key to the tractability of our method even in the presence of constraints is that the only step that requires any recomputation for each candidate split is the estimation of $\nabla f_j(z_0), d_z^{j*}$. As in the unconstrained case, estimators for $\nabla f_j(z_0)$ usually consist of very simple sample averages over the data in the region R_j so they can also be very quickly computed. Moreover, only the right-hand side defining d_z^{j*} in Equation (26) varies with each candidate split, so the equation can be presolved using an *LU* decomposition or a similar approach. Therefore, we can easily and quickly consider many candidate splits, and correspondingly grow large-scale forests.

Algorithm 2 for fitting the forest remains the same, because the only change in fitting is in the consideration of tree splits. Algorithm 3 was already written in the general constrained setting and therefore also remains the same. In particular, after growing a forest where tree splits take the constraints into consideration and given this forest, we impose the constraints in $\mathcal{Z}$ when computing the final forest-policy decision, $\hat{z}(x)$.

4. Empirical Study

In this section, we study our algorithm and baselines empirically to investigate the value of optimization-aware construction of forest policies and the success of our algorithm in doing so. We focus on constrained CSO problems with CVaR objectives, including one simulated portfolio optimization problem and one real-data shortest path problem. In Online Appendices C.1, C.4, and C.5, we show additional experimental results for unconstrained multi-item newsvendor problems (Example 1) and constrained variance-based portfolio optimization problems (Example 2).

Implementation and replication code are available at https://github.com/CausalML/StochOptForest.

4.1. CVaR Portfolio Optimization

We first apply our method to the CVaR portfolio optimization problem (see Example 3). We consider $d = 3$ assets and $p = 10$ covariates. The covariates X are drawn from a standard Gaussian distribution, and the asset returns are independent and are drawn from the conditional distributions $Y_1 \mid X \sim 1 + 0.2\exp(X_1) - \text{LogNormal}(0, 1 - 0.5\mathbb{I}[-3 \le X_2 \le -1])$, $Y_2 \mid X \sim 1 - 0.2\, X_1 - \text{LogNormal}(0, 1 - 0.5\mathbb{I}[-1 \le X_2 \le 1])$, and $Y_3 \mid X \sim 1 + 0.2 \mid X_1 \mid - \text{LogNormal}(0, 1 - 0.5\mathbb{I}[1 \le X_2 \le 3])$. We seek an investment policy $z(\cdot) \in \mathbb{R}^d$ that for each x aims to achieve smallest risk $\text{CVaR}_{0.2}(Y^\top z(x) \mid X = x)$, or equivalently the 0.8-CVaR of the portfolio loss $-Y^\top z(x)$ while satisfying the simplex constraint, *i.e.*, $\mathcal{Z} = \{z \in \mathbb{R}^d : \sum_{l=1}^d z_l = 1, z_l \ge 0\}$.

We compare our StochOptForest algorithm using either the apx-risk or apx-soln approximate criterion for constrained problems (Equations (29) and (30)) to five benchmarks, where all algorithms are identical except for their *splitting criterion*. The first two benchmarks are our StochOptForest algorithm using apx-risk and apx-soln criteria that (mistakenly) *ignore* the constraints (i.e., Equations (13) and (14)). The third benchmark is a modified[6] GenRandForest algorithm (Athey et al. 2019) applied to the first-order optimality condition for the CVaR optimization problem *without* the simplex constraint, as GenRandForest is designed for *unconstrained* problems. The fourth benchmark is the regular RandForest, which uses the squared error splitting criterion in Example 4 and targets the predictions of asset mean returns, and the fifth is the RandSplitForest algorithm, which chooses splits uniformly at random (without using the portfolio return data). For our approximate criteria (both constrained and unconstrained) and the GenRanForest criterion, we use the parametric Hessian estimator in Equations (23) and (24) (which is *misspecified* in this example). We do not compare with StochOptForest with the oracle splitting criterion because it is too computationally intensive as we investigate further (Table 1). In all forest algorithms, we use an ensemble of 500 trees. To compute $\hat{z}_0$ in our StochOptTree algorithm (Algorithm 1, line 2) and to compute the final forest policy for any forest, we formulate the constrained CVaR optimization problem as a linear programming problem (Rockafellar and Uryasev 2000) and solve it using Gurobi 9.0.2. We evaluate each forest policy $\hat{z}(\cdot)$ by its

Table 1. Average Running Time in Seconds over 10 Repetitions (and Standard Deviations) of Constructing One Tree for Different Algorithms in the CVaR Optimization Problem

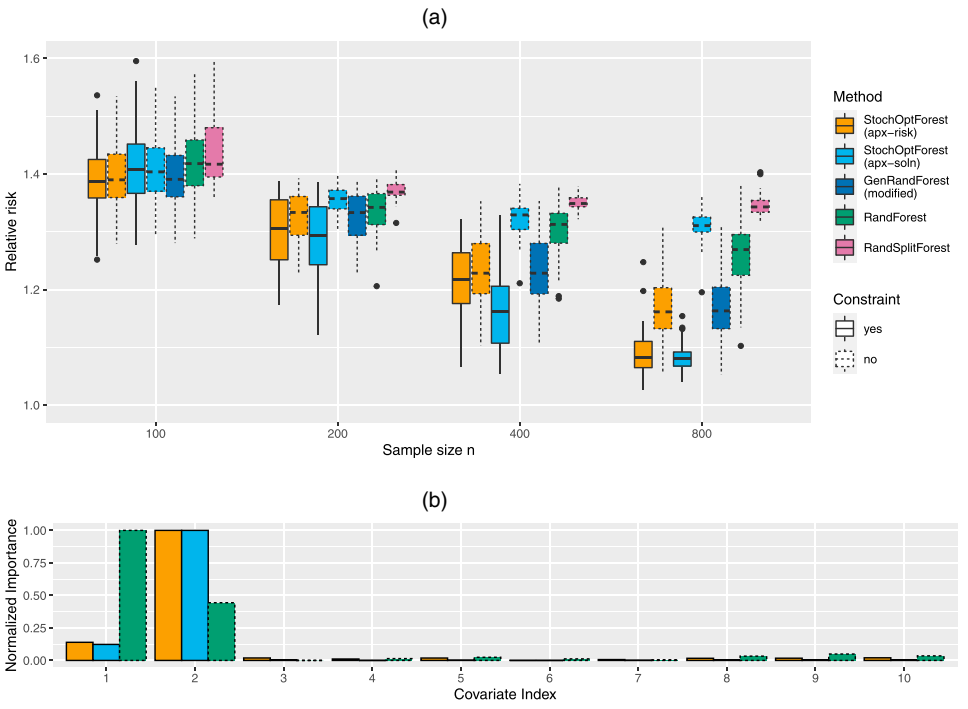
Method	$n = 100$	$n = 200$	$n = 400$
StochOptTree (oracle)	41.41 (5.43)	165.03 (15.77)	695.88 (83.91)
StochOptTree (apx-risk)	0.26 (0.08)	0.68 (0.36)	1.68 (0.54)
StochOptTree (apx-soln)	0.22 (0.05)	0.70 (0.20)	2.24 (0.33)

relative risk compared with the optimal $z^*(\cdot)$, namely the ratio of $\mathbb{E}[\ \mathrm{CVaR}_{0.2}(Y^\top \hat{z}(X)\,|\,X)|\mathcal{D}]$ over $\mathbb{E}[\mathrm{CVaR}_{0.2}(Y^\top z^*(X)\,|\,X)]$, which we approximate using a very large testing data set. See Online Appendix C.2 for more details.

Figure 2(a) shows the distribution of the relative risk over 50 replications for each forest algorithm across different training set size $n \in \{100, 200, 400, 800\}$. The dashed boxes corresponding to "Constraint = no" indicate that the associated method does not take constraints into account when choosing the splits, which applies to all four benchmarks. (*All* methods consider constraints in computing the final forest-policy decision, $\hat{z}(x)$.) We can observe that our StochOptForest algorithms with approximate criteria that incorporate constraints achieve the best relative risk over all sample sizes, and their relative risks decrease considerably when the training set size n increases. In contrast, the relative risks of all benchmark methods decrease much more slowly. Therefore, both failing to target the cost function structure (GenRandForest,[7] RandForest, and RandSplitForest) *and* failing to take constraints into account (all five benchmark methods) can significantly undermine the ultimate decision-making quality. In contrast, our StochOptForest algorithms based on the approximate criteria effectively account for both, so they perform much better. Moreover, our results show that, although the normal distribution assumption used to derive our Hessian estimator (Equations (23) and (24)) is wrong in our experiment, our proposed forest policies still achieve superior performance, which illustrates the robustness of our methods.

To further understand these results, we also consider feature importance measures based on each forest algorithm. In Online Appendix B, we extend the impurity-based feature importance measures (Hastie et al. 2009) to our StochOptForest method. Recall there are $p = 10$ covariates, and the first two determine the distributions of asset returns. The first covariate influences the conditional mean of return distributions more, whereas the second one influences more the distribution tails. In Figure 2(b), we visualize the feature importance measures for our proposed method and RandForest when $n = 800$. The importance measures are normalized for each method so that the most

Figure 2. (Color online) Results for the CVaR Portfolio Optimization Problem



Notes. (a) Relative risks of different forest policies (lower relative risk means better performance). (b) Feature Importance.

important feature has an importance value equal to one. We can observe that our StochOptForest methods (incorporating constraints) value the second covariate more than the first one, which shows the importance of signals in the return distribution tails for CVaR optimization. In contrast, the RandForest algorithm puts more importance on the first covariate, validating that it is designed to target the prediction of asset mean returns. There do not exist feature importance measures for the GenRandForest algorithm. Instead, we show its average frequency of splitting on each covariate in Online Appendix C.2, Figure EC.4. We observe that the GenRandForest method splits on noise covariates (i.e., the 3rd to 10th covariate) more frequently than our proposals, which may partly explain its inferior performance.

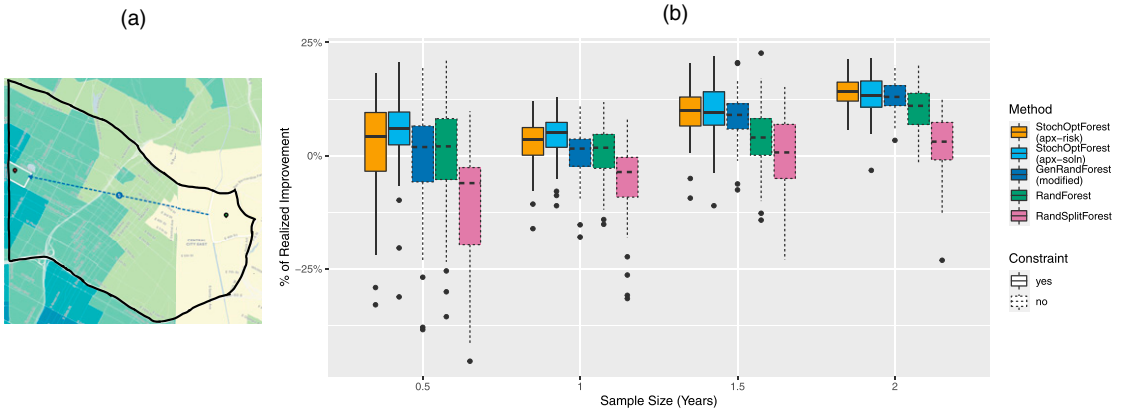
We also consider the average running time of our proposed algorithm in Table 1. We compare our StochOptTree algorithm with approximate criteria *incorporating* constraints to the oracle splitting criterion (using empirical expectations). We consider 10 repetitions, in each of which we apply each tree algorithm with the same specifications to construct a single tree on the same training data with varying size $n \in \{100, 200, 400\}$. We run this experiment on a MacBook with 2.7-GHz Intel Core i5 processor. We can see that the running time of our StochOptTree algorithm with apx-risk criterion is hundreds of times faster than the StochOptTree algorithm with the oracle criterion that must solve the constrained CVaR optimization problems for each candidate split. The computational gains of our approximate criteria relative to the oracle criterion also grow with larger sample size n (from around 200 times faster at $n = 100$ to more than 400 times faster at $n = 400$), as the CVaR optimization problem becomes slower to solve.

Because the StochOptForest algorithm with the oracle criterion is extremely slow, we can only evaluate its performance in a small-scale experiment in Figure EC.5 in Online Appendix C.2. Focusing on constructing small forests of only 50 trees with n up to 400, we find the performance of the oracle criterion is marginally better than our approximate criteria. However, our approximate criteria are much more computationally efficient, which enables us to leverage larger data sets for better performance. In Online Appendix C.2, we also show that similar results hold for portfolio optimization with a linear combination of CVaR and mean return as the objective (Figure EC.6) and for CVaR optimization with asset returns drawn from normal distributions (Figure EC.7). We include additional empirical results on minimizing the variance of investment portfolios (Example 2) in Online Appendix C.4 and show that the performance of our approximate criteria is close to the oracle criterion.

4.2. CVaR Shortest Path Problem Using Uber Movement Data

We next demonstrate our methods in a shortest path problem, using traveling times data in Los Angeles (LA) collected from Uber Movement (https://movement.uber.com). We focus on 45 census tracts in downtown LA (Figure 3(a)), collecting historical data of average traveling times from each of these census tracts to its neighbors during five periods in each day (AM Peak, Midday, PM Peak, Evening, Early Morning) in 2018 and 2019. This results in 3,650 observations of traveling times Y_j for $j = 1, \ldots, 93$ edges on a graph with 45 nodes. We consider $p = 197$ covariates X including weather, period of day and other calendar features, and lagged traveling times. We aim to go from an eastmost census tract (Aliso Village) to a westmost census tract (MacArthur Park) in this region (the endpoints of the arrow in Figure 3(a)), through a path between them, encoded by $z \in \{0,1\}^d$ with $d = 93$, where z_j indicates whether we travel on edge j. In particular, we consider the CSO problem $z^*(x) \in \arg\min_{z(\cdot)\in\mathcal{Z}} \mathrm{CVaR}_{0.8}(Y^\top z(x) \mid X = x)$, where $\mathcal{Z}$ is given by standard flow preservations constraints,

Figure 3. (Color online) Setup and Results for the CVaR Shortest Path Problem Using Uber Movement Data



Notes. (a) The downtown Los Angeles region (enclosed by the black lines) for the shortest path problem. (b) Percentages of realized improvement by different forest policies for the CVaR shortest-path problem. A higher percentage is better.

with a source of +1 at Aliso Village and a sink of –1 at MacArthur Park. See Online Appendix C.3 for more details about data collection, optimization formulation, and other experiment specifications.

We again compare different forest algorithms as we do in Section 4.1, but to reduce computation, we only train them up to 100 trees. We consider four different sample sizes ranging from 0.5-year to the whole 2-year data. For each sample size, we randomly split the corresponding data set into two halves as training data $\mathcal{D}_{\text{train}}$ and testing data $\mathcal{D}_{\text{test}}$, respectively. The distribution of $Y \mid X$ is unknown, so we can no longer benchmark the performance of each forest policy $\hat{z}(\cdot)$ trained on $\mathcal{D}_{\text{train}}$ against the CSO optimal policy $z^*(\cdot)$ as in Section 4.1. Instead, we compare their percentages of realized improvement, termed the coefficient of prescriptiveness in Bertsimas and Kallus (2020). Namely, we consider the ratio between each method's improvement over the context-free sample average approximation (SAA), which finds a single solution $\hat{z}_{\text{SAA}}$ to optimize the average cost on the whole training data, over the improvement over SAA of the (infeasible) perfect-information shortest path, which in each test sample computes the shortest path for the observed travel time Y. More effective forest policies have higher percentages of realized improvement, but even known distributions optimal policy $z^*(\cdot)$ to Equation (1) cannot generally achieve 100% realized improvement as the covariates do not perfectly predict travel times.

In Figure 3(b), we show the results across 50 realizations of random train-test splits. We observe that as the sample sizes increase, all methods tend to perform better. In particular, our StochOptForest algorithm with either the apx-risk or apx-soln criterion (incorporating constraints) outperforms all benchmarks across all sample sizes, with the clearest improvement seen using the apx-risk criterion and in smaller data sets. Overall, the results show that incorporating the optimization problem structure in the tree construction can lead to improvements when optimization is the aim.

5. Asymptotic Optimality

In this section, we prove that under some regularity conditions, our forest policy asymptotically attains the optimal risk, namely, $\mathbb{E}[c(\hat{z}_n(x); Y) \mid X = x]$ converges in probability to $\min_{z \in \mathcal{Z}} \mathbb{E}[c(z; Y) \mid X = x]$ as $n \to \infty$ for any $x \in \mathcal{X}$.

It is well known that forests algorithms with adaptively constructed trees are extremely difficult to analyze, so some simplifying regularity conditions are often needed to make the theoretical analysis tractable (Biau and Scornet 2016). In this section, we assume the tree regularity conditions introduced by Athey et al. (2019) and Wager and Athey (2018).

Assumption 1 (Regular Trees). *The trees constructed satisfy the following regularity conditions for constants $\omega \in (0, 0.2]$, $\pi \in [0, 1)$, and an integer $k_n > 0$:*

1. *Every tree split puts at least a fraction ω of observations in the parent node into each child node. Every leaf node in every tree contains between k_n and $2k_n - 1$ observations.*

2. *For an index set $\mathcal{J} \subseteq \{1, \dots, p\}$ such that $\mathbb{E}[c(z; Y) \mid X] = \mathbb{E}[c(z; Y) \mid X_{\mathcal{J}}]$ for all $z \in \mathcal{Z}$, for each leaf of each tree and for each $j \in \mathcal{J}$, the average probability of splitting along feature x_j is bounded below by π/p, averaging over nodes on the path from the root to the leaf and marginalizing over any randomization of candidate splits (and conditioning on the data).*

3. *Each tree grows on a subsample of size s_n drawn randomly without replacement from the whole training data, and it is honest, that is, $\mathcal{I}_j^{\text{tree}} \cap \mathcal{I}_j^{\text{dec}} \neq \emptyset$ with $|\mathcal{I}_j^{\text{tree}}| + |\mathcal{I}_j^{\text{dec}}| = s_n$ for $j = 1, \dots, T$.*

Condition 1 in Assumption 1 specifies that the stopping criterion must ensure a minimal leaf size and that all candidate splits be balanced in that they put at least a constant fraction of observations in each child node. Without this condition, even when sample size n is large, some imbalanced splits may run out of data so quickly that some leaves are not sufficiently partitioned and thus too large. As a result, the estimation bias of the objective function may fail to vanish even when $n \to \infty$. Condition 2 requires the trees to split along every relevant direction at sufficient frequency, which ensures that the leaves of the trees become small in all relevant dimensions of the feature space as n gets large. Relevant features are described by those such that the random cost of any decision is mean-independent of X given only these relevant features, which is trivially satisfied for $\mathcal{J} = \{1, \dots, p\}$. Condition 3 specifies that we use subsample splitting, that is, the data used to construct each tree ($\mathcal{I}_j^{\text{tree}}$) and the data used to construct localized weights from this tree for final decision making ($\mathcal{I}_j^{\text{dec}}$) are disjoint. This so-called honesty property plays a critical role in the theoretical analysis of forest algorithms, but it may be largely technical. In Section 4, we empirically show that our forest policies appear to achieve asymptotic optimality even without using honest subsample splitting. In Online Appendix C.6, we further illustrate in Figure EC.11 that StochOptForest with no subsample splitting (i.e., $\mathcal{I}_j^{\text{dec}} = \mathcal{I}_j^{\text{tree}}$) performs better than the honest version with splitting, which can be explained as honest trees using fewer data for tree construction and decision making.

In the following assumption, we further impose some regularity conditions on the cost function $c(z; y)$ and the distribution of $Y \mid X$.

Assumption 2 (Distribution Regularity). *Fix $x \in \mathcal{X}$ and assume the following conditions:*

1. *The marginal distribution of X has a density, its support $\mathcal{X}$ is compact, and the density is bounded away from zero and ∞ on $\mathcal{X}$.*
2. *There exist a constant α and a compact set $\mathcal{C} \subseteq \mathcal{Z}$ such that, $\{z \in \mathcal{Z} : \mathbb{E}[c(z;Y) \mid X = x] \le \alpha\} \subseteq \mathcal{C}$ and $\{z \in \mathcal{Z} : \sum_{i=1}^n w_i(x) c(z;Y_i) \le \alpha\} \subseteq \mathcal{C}$ for $w_i(x)$ in Equation (25) almost surely eventually.*
3. *There exists a function $b(y)$ such that for any $z, z' \in \mathcal{C}, y \in \mathcal{Y}$, $|c(z;y) - c(z';y)| \le b(y) \|z - z'\|_2$. Moreover, there exists a positive constant $\tilde{C}$ such that $\mathbb{E}[b(Y) \mid X = x] \le \tilde{C} < \infty$.*
4. *There exist constants L_c and L_b such that $\sup_{z \in \mathcal{C}} \sup_{x' \in \mathcal{X}} |\mathbb{E}[c(z;Y) \mid X_{\mathcal{J}} = x_{\mathcal{J}}] - \mathbb{E}[c(z;Y) \mid X_{\mathcal{J}} = x'_{\mathcal{J}}]| \le L_c \|x_{\mathcal{J}} - x'_{\mathcal{J}}\|_2$ and $\sup_{x' \in \mathcal{X}} |\mathbb{E}[b(Y) \mid X_{\mathcal{J}} = x_{\mathcal{J}}] - \mathbb{E}[b(Y) \mid X_{\mathcal{J}} = x'_{\mathcal{J}}]| \le L_b \|x_{\mathcal{J}} - x'_{\mathcal{J}}\|_2$.*
5. *There exist positive constants η, η', C such that*

$$\sup_{z \in \mathcal{C}} \mathbb{E}\left[e^{\eta\,|c(z;Y) - \mathbb{E}[c(z;Y)|X=x]|} \mid X = x\right] \le C < \infty,$$
$$\mathbb{E}\left[e^{\eta'\,|b(Y) - \mathbb{E}[b(Y)|X=x]|} \mid X = x\right] \le C < \infty.$$

One important condition in Assumption 2 is that the cost function $c(z;y)$ is Lipschitz-continuous in z on the compact set $\mathcal{C}$. In the following proposition, we validate that Examples 1 to 3 all satisfy this condition.

Proposition 2. *For any $z, z' \in \mathcal{C}$:*

1. *The cost function $c(z;y) = \sum_{l=1}^d \max\{\alpha_l(z_l - y_l), \beta_l(y_l - z_l)\}$ for the newsvendor problem in Example 1 satisfies that $|c(z;y) - c(z';y)| \le \sqrt{d} \max\{\alpha_l, \beta_l\} \|z - z'\|_2$.*
2. *The cost function $c(z;y) = (y^\top z_{1:d} - z_{d+1})^2$ for the variance-based portfolio optimization problem in Example 2 satisfies that $|c(z;y) - c(z';y)| \le 4\sqrt{2}(\sup_{\tilde{z} \in \mathcal{C}} \|\tilde{z}\|_2) \max\{1, \|y\|_2^2\} \|z - z'\|_2$.*
3. *The cost function $c(z;y) = \frac{1}{\alpha}\max\{z_{d+1} - y^\top z_{1:d}, 0\} - z_{d+1}$ for the CVaR optimization problem in Example 3 satisfies that $|c(z;y) - c(z';y)| \le (\|y\|_2 + 1 + \frac{1}{\alpha}) \|z - z'\|_2$.*

Under the previous assumptions, we can prove that the forest policy is asymptotically optimal.

Theorem 5. *Let $x \in \mathcal{X}$ be fixed. If Assumptions 1 and 2 hold at the given x and if $k_n \to \infty$, $s_n/k_n \to \infty$, $\log T/k_n \to 0$, and $Tk_n/s_n \to 0$, then*

$$\sup_{z \in \mathcal{C}} \left| \sum_{i=1}^n w_i(x) c(z;Y_i) - \mathbb{E}[c(z;Y) \mid X = x] \right| \xrightarrow{p} 0. \tag{31}$$

It follows that any choice $\hat{z}_n(x) \in \arg\min_{z \in \mathcal{Z}} \sum_{i=1}^n w_i(x) c(z;Y_i)$ satisfies that as $n \to \infty$,

$$\left| \mathbb{E}[c(\hat{z}_n(x);Y) \mid X = x] - \min_{z \in \mathcal{Z}} \mathbb{E}[c(z;Y) \mid X = x] \right| \xrightarrow{p} 0. \tag{32}$$

Theorem 5 provides asymptotic optimality of $\hat{z}_n(x)$ pointwise in x. The result can straightforwardly be extended to be uniform in x if we simply assume the conditions in Assumption 2 hold for all $x \in \mathcal{X}$ with common constants.

6. Discussion

In this section, we offer some discussions. First, we discuss how our work is related to and differs from work on *estimation* using localized weights and forests in particular. Then we discuss other related work on CSO and on integrating prediction and optimization. We discuss additional related literature about tree models and perturbation analysis in Online Appendix F.

6.1. Comparison with Estimation

The idea of using localized weights to estimate parameters given covariate values has a long history in statistics and econometrics, including applications in local maximum likelihood (Tibshirani and Hastie 1987, Fan et al. 1998), local generalized method of moments (Lewbel 2007), local estimating equation (Carroll et al. 1998), and so on. These early works typically use nonadaptive localized weights like nearest-neighbor weights or Nadaraya-Watson kernel weights, which only use the information of covariates. Recently, some literature proposed to use forest-based weights for local parameter estimation (Meinshausen and Ridgeway 2006, Scornet 2016, Athey et al. 2019, Oprescu et al. 2019), which generalizes the original random forest algorithm for regression and classification problems (Breiman 2001) to other estimation problems where the estimand depends on the X-conditional distribution. These forest-based weights are derived from the proportion of trees in which each observation falls in the same terminal node as the target covariate value. Because those trees are adaptively constructed using label data as well, random forest weights are shown to be more effective in modeling complex heterogeneity in high dimensions than nonadaptive weights. Recent literature has studied the statistical guarantees of random forests in estimating conditional expectation functions (Biau and Scornet 2016, Wager and Athey 2018) or more general parameters defined by local estimating equations (Athey et al. 2019, Oprescu et al. 2019).

Among the previous statistical estimation literature, closest to our work is Athey et al. (2019), who propose the GenRandForest algorithm to estimate roots of conditional estimating equations. This is closely related to our decision-making problem, because the optimal solution of *unconstrained* CSO is also the root of a conditional estimating equation given by the first-order optimality condition. For example, the optimal solutions of conditional newsvendor problem in Example 1 without constraints are conditional quantiles, which are also considered by Athey et al. (2019) under the

conditional estimating equation framework. For computational efficiency, Athey et al. (2019) also propose a gradient-based approximation for roots in candidate subpartitions (see discussions after Equation (14)) and then find the best split that maximizes the discrepancy of the approximate roots in the subregions, thereby approximately minimizing the total *mean squared error* of the estimated roots (Athey et al. 2019, proposition 1).

In contrast, our paper has a fundamentally different goal: we target *decision-making risk* (expected cost) rather than *estimation risk* (accuracy). In our apx-risk and apx-soln criteria, we directly approximate the optimal average cost itself and use this to choose a split rather than estimation error of the solution. In Online Appendix C.1, we provide one empirical example of unconstrained newsvendor problem where the heterogeneity of optimal solution estimation is drastically different from the heterogeneity of the optimal decision making, which illustrates the benefit of targeting decision quality when the decision problem rather than the estimation problem is of interest. Moreover, our methods uniquely accommodate constraints, which are prevalent in decision-making problems but rare in statistical estimation problems. For *constrained* CSO, the optimal solution cannot be characterized by local estimating equations, so the GenRandForest algorithm is not applicable. In Section 4, we provided empirical examples of constrained CVaR optimization problems where taking into account constraints is key to constructing good trees.

6.2. CSO and Integrating Prediction and Optimization

Our paper builds on the CSO framework, and the general local learning approach, that is, estimating the objective (and stochastic constraints in Online Appendix A) by weighted averages, with weights reflecting the proximity of each covariate observation to the target value. Bertsimas and Kallus (2020), Hanasusanto and Kuhn (2013), and Hannah et al. (2010) propose the use of nonparametric weights that use only the covariate observations X and do not depend on observations of the uncertain variable Y, such as Nadaraya-Watson weights. Bertsimas and Kallus (2020) formally set up the CSO framework, propose a wide variety of machine learning methods for local weights construction, and provide rigorous asymptotic optimality guarantees. In particular, they additionally propose weights based on decision trees and random forests that incorporate the uncertain variable information and show their superiority when the covariate dimension is high. However, their tree and forest weights are constructed from standard regression algorithms that target prediction accuracy instead of downstream decision quality, primarily because targeting the latter would be too computationally expensive. Our paper resolves this computational challenge by leveraging approximate criteria that can be efficiently computed.

Optimization problems that have unknown parameters, such as an unknown distribution or a conditional expectation, are often solved by a two-stage approach: the unknown parameters are estimated or predicted, then these are plugged in, and then the approximated optimization problem is solved. The estimation or prediction step is often done independently of the optimization step, targeting standard accuracy measures such as mean squared error without taking the downstream optimization problem into account. However, all predictive models make errors and when prediction and optimization are completely divorced, the error tradeoffs may be undesirable for the end task of decision making. To deal with this problem, recent literature proposes various ways to tailor the predictions to the optimization problems.

Elmachtoub and Grigas (2022) study a special CSO problem where $c(z;y) = y^\top z$ is linear and constraints are deterministic and known. In this special case, the parameter of interest is the conditional expectation $\mathbb{E}[Y \mid X = x]$, which forms the linear objective's coefficients. They propose to fit a parametric model to predict the coefficients by minimizing a convex surrogate loss of the suboptimality of the decisions induced by predicted coefficients. Elmachtoub et al. (2020) study the same linear CSO problem and instead predict the coefficients nonparametrically by decision trees and random forests with suboptimality as the splitting criterion. In Online Appendix G, Proposition EC.15, we show this criterion is equivalent to what we termed the oracle splitting criterion in Equation (8) in the case of linear costs. Because this involves full reoptimization for each candidate split, they are limited to very few candidate splits, suggesting using one per candidate feature, and they consider a relatively small number of trees in their forests. In contrast, we consider the general CSO problem and use efficient approximate criteria, which is crucial for large-scale problems and training large tree ensembles. Hu et al. (2022) also study linear CSO problems, and they show both theoretically and empirically that with correctly specified models, integrated approaches may perform worse than the simpler predict-then-optimize approach. Our paper demonstrates the benefit of a forest-based integrated approach in *nonlinear* CSO problems, where a predict-then-optimize approach would have to learn the whole conditional distribution and not just the conditional expectation.

Donti et al. (2017) study smooth convex optimization problems with a parametric model for the conditional distribution of the uncertain variables (in both objective and constraints) given covariates and fit the parametric models by minimizing the decision

objective directly using gradient descent methods on the optimization risk instead of the log-likelihood. Wilder et al. (2019) further extend this approach to nonsmooth problems by leveraging differentiable surrogate problems. However, unless the cost function depends on the uncertain variables linearly, the stochastic optimization problem may involve complicated integrals with respect to the conditional distribution model. In contrast, our paper focuses on nonparametric forest models that cannot be trained by gradient-based methods, and we can straightforwardly target the CSO using localized weights. Notz (2020) considers convex optimization problems with nondifferentiable cost functions and proposes a subgradient boosting algorithm to directly learn a decision policy. Although this approach can handle complex objectives, it can only accommodate very simple constraints like box constraints, as it is difficult to impose complex constraints on boosting decision policies. In contrast, our approach based on the CSO framework can readily handle general constraints.

7. Concluding Remarks

In CSO problems, covariates X are used to reduce the uncertainty in the variable Y that affects costs in a decision-making problem. The remaining uncertainty is characterized by the conditional distribution of $Y \mid X = x$. A crucial element of effective algorithms for learning policies for CSO from data are the integration of prediction and optimization. One can try to fit generic models that predict the distribution of $Y \mid X = x$ for every x and then plug this in place of the true conditional distribution, but fitting such a model to minimize prediction errors without consideration of the downstream decision-making problem may lead to ill-performing policies. In view of this, we studied how to fit forest policies for CSO (which use a forest to predict the conditional distribution) in a way that directly targets the optimization costs. The naïve direct implementation of this is hopelessly intractable for many important managerial decision-making problems in inventory and revenue management, finance, and so on. Therefore, we instead developed efficient approximations based on second-order perturbation analysis of stochastic optimization. The resulting algorithm, StochOptForest, is able to grow large-scale forests that directly target the decision-making problem of interest, which empirically leads to significant improvements in decision quality over baselines.

Endnotes

[1] This is, for example, a corollary of Theorem 1, although weaker continuity conditions would be needed for this first-order statement. We omit the details as the first-order analysis is ultimately not useful.

[2] Indeed the unconstrained case for the portfolio problem is in fact uninteresting: the zero portfolio gives minimal variance, and CVaR may be sent to infinity in either direction by infinite scaling.

[3] Ties can be broken arbitrarily.

[4] We may similarly use the $\approx 1/e$ fraction of the data not selected by the bootstrap sample to construct $\mathcal{I}^{\text{dec}_j}$ to achieve honesty, but this is again uncommon as it is difficult to analyze.

[5] In the context of such linear problems, Elmachtoub et al. (2020) propose to optimize the oracle criterion by exhaustive search. As noted before, this is computationally burdensome, and indeed their focus is on smaller-scale models, with particular benefits to interpretability. In Proposition EC.15 in Appendix G, we formally argue that their criterion coincides with what we called the oracle criterion in Equation (8) in the case of linear costs.

[6] We cannot apply the original GenRandForest algorithm to solve *unconstrained* CVaR optimization: every step of tree construction requires computing the optimal *unconstrained* solution in the region R_0 to be partitioned, which, however, does not exist because without constraints the CVaR objectives can be made arbitrarily small. We thus must slightly modify the GenRandForest algorithm to compute the optimal *constrained* solution in every region to be partitioned, from which we then compute the GenRandForest splitting criterion for the first-order optimality condition of unconstrained CVaR optimization. We furthermore regularize the Hessian matrix as it is not generally invertible, as discussed after Equation (18), which would make the GenRandForest splitting criterion undefined. See Appendix C.2.

[7] GenRandForest criterion partly captures the cost function structure as it incorporates the corresponding first-order optimality condition information, but it chooses splits to maximize the discrepancy of approximate solutions in the induced subregions rather than optimize their decision costs directly.

References

Athey S, Tibshirani J, Wager S (2019) Generalized random forests. *Ann. Statist.* 47(2):1148–1178.

Bertsekas D (1995) Nonlinear programming (Athena Scientific, Nashua, NH).

Bertsimas D, Kallus N (2020) From predictive to prescriptive analytics. *Management Sci.* 66(3):1025–1044.

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

Bonnans JF, Shapiro A (2000) *Perturbation Analysis of Optimization Problems* (Springer, New York).

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

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) *Classification and Regression Trees* (CRC Press, Boca Raton, FL).

Carroll RJ, Ruppert D, Welsh AH (1998) Local estimating equations. *J. Amer. Statist. Assoc.* 93(441):214–227.

Chen Z, Leng C (2015) Local linear estimation of covariance matrices via cholesky decomposition. *Statist. Sinica* 25(3):1249–1263.

Cornuejols G, Tütüncü R (2006) *Optimization Methods in Finance* (Cambridge University Press, Cambridge, UK).

Denil M, Matheson D, Freitas ND (2014) Narrowing the gap: Random forests in theory and in practice. *Proc. 31st Internat. Conf. on Machine Learn.*, 665–673.

Donti P, Amos B, Kolter JZ (2017) Task-based end-to-end model learning in stochastic optimization. *Adv. Neural Inform. Processing Systems* 30:5484–5494.

Elmachtoub AN, Grigas P (2022) Smart "predict, then optimize". *Management Sci.* 68(1):9–26.

Elmachtoub AN, Liang JCN, McNellis R (2020) Decision trees for decision-making under the predict-then-optimize framework. *Proc. 37th Internat. Conf. Machine Learn. Proc. Machine Learn. Res.* vol. 119 (PMLR), 2858–2867.

Fan J, Yao Q (1998) Efficient estimation of conditional variance functions in stochastic regression. *Biometrika* 85(3):645–660.

Fan J, Farmen M, Gijbels I (1998) Local maximum likelihood estimation and inference. *J. Royal Statist. Soc. Ser. B Statist. Methodology* 60(3):591–608.

Hanasusanto GA, Kuhn D (2013) Robust data-driven dynamic programming. *Adv. Neural Inform. Processing Systems* 26:827–835.

Hannah L, Powell W, Blei DM (2010) Nonparametric density estimation for stochastic optimization with an observable state variable. *Adv. Neural Inform. Processing Systems* 23:820–828.

Hastie T, Tibshirani R, Friedman JH (2009) *The Elements of Statistical Learning*: *Data mining, inference, and prediction* (Springer, New York).

Hu Y, Kallus N, Mao X (2022) Fast rates for contextual linear optimization. *Management Sci.* 68(6):4236–4245.

Kleywegt AJ, Shapiro A (2001) Stochastic optimization. Salvendy G, ed. *Handbook of Industrial Engineering* (John Wiley & Sons, New York), 2625–2649.

Lewbel A (2007) A local generalized method of moments estimator. *Econom. Lett.* 94(1):124–128.

Loubes JM, Marteau C, Solís M (2020) Rates of convergence in conditional covariance matrix with nonparametric entries estimation. *Comm. Statist. Theory Methods* 49(18):4536–4558.

Meinshausen N, Ridgeway G (2006) Quantile regression forests. *J. Machine Learn. Res.* 7(6):983–999.

Mentch L, Hooker G (2016) Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *J. Machine Learn. Res.* 17(1):841–881.

Nemirovski A, Juditsky A, Lan G, Shapiro A (2009) Robust stochastic approximation approach to stochastic programming. *SIAM J. Optim.* 19(4):1574–1609.

Notz PM (2020) Explainable subgradient tree boosting for prescriptive analytics in operations management. Preprint, submitted August 4, https://dx.doi.org/10.2139/ssrn.3567665.

Oprescu M, Syrgkanis V, Wu ZS (2019) Orthogonal random forest for causal inference. Chaudhuri K, Salakhutdinov R, eds. *Proc. 36th Internat. Conf. on Machine Learn.*, vol. 97 of Proceedings of Machine Learning Research (PMLR, Long Beach, CA), 4932–4941.

Rockafellar RT, Uryasev S (2000) Optimization of conditional value-at-risk. *J. Risk* 2(3):21–42.

Scornet E (2016) Random forests and kernel methods. *IEEE Trans. Inform. Theory* 62(3):1485–1500.

Scornet E, Biau G, Vert JP (2016) Consistency of random forests. *Ann. Statist.* 43(4):1716–1741.

Shapiro A, Dentcheva D, Ruszczyński A (2014) *Lectures on Stochastic Programming: Modeling and Theory* (SIAM, Philadelphia).

Simchi-Levi D, Chen X, Bramel J (2005) The logic of logistics. *Theory, Algorithms, and Applications for Logistics and Supply Chain Management* (Springer, New York).

Talluri KT, Van Ryzin GJ (2006) *The Theory and Practice of Revenue Management* (Springer, New York).

Tibshirani R, Hastie T (1987) Local likelihood estimation. *J. Amer. Statist. Assoc.* 82(398):559–567.

Wager S, Athey S (2018) Estimation and inference of heterogeneous treatment effects using random forests. *J. Amer. Statist. Assoc.* 113(523):1228–1242.

Wilder B, Dilkina B, Tambe M (2019) Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. *Proc. AAAI Conf. on Artificial Intelligence*, 1658–1665.

Yin J, Geng Z, Li R, Wang H (2010) Nonparametric covariance model. *Statist. Sinica* 20:469–479.