Using BART for Multiobjective Optimization of Noisy Multiple Objectives

Akira Horiguchi*[1], Thomas J. Santner[1], Ying Sun[2], and Matthew T. Pratola[1]

[1]*Department of Statistics*
The Ohio State University
Cockins Hall
1958 Neil Ave.
Columbus, OH 43210
[2]*Statistics Program*
King Abdullah University of Science and Technology
Thuwal 23955-6900
Saudi Arabia

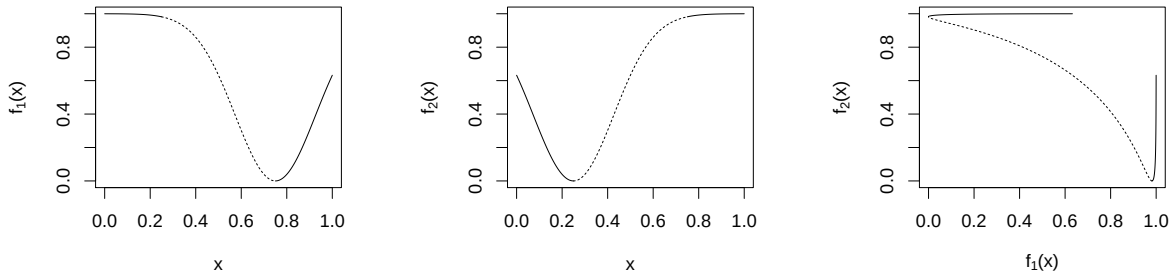
January 8, 2021

Abstract

Techniques to reduce the energy burden of an Industry 4.0 ecosystem often require solving a multiobjective optimization problem. However, collecting experimental data can often be either expensive or time-consuming. In such cases, statistical methods can be helpful. This article proposes Pareto Front (PF) and Pareto Set (PS) estimation methods using Bayesian Additive Regression Trees (BART), which is a non-parametric model whose assumptions are typically less restrictive than popular alternatives, such as Gaussian Processes. The performance of our BART-based method is compared to a GP-based method using analytic test functions, demonstrating convincing advantages. Finally, our BART-based methodology is applied to a motivating Industry 4.0 engineering problem. Supplementary materials, which include a theorem proof, algorithms, and R code, for this article are available online.

Keywords: Industry 4.0, Uncertainty Quantification, Random Sets, Band Depth

*Corresponding author. Email: horiguchi.6@osu.edu



(a) $f_1(x)$ vs x. (b) $f_2(x)$ vs x. (c) $f_2(x)$ vs $f_1(x)$.

Figure 1: A biobjective function $\mathbf{f}(x) = (f_1(x), f_2(x))$ where $x \in \mathcal{X} = [0, 1]$.

1 Introduction

Many important Industry 4.0 problems (e.g. Xie et al., 2014; Han et al., 2016; Ivanov et al., 2016; Fu et al., 2018; Shukla et al., 2020) can be formulated as multiobjective optimization (MO) problems,. The goal of MO is to "minimize" (or "maximize", depending on the application) some multiobjective deterministic function

$$\mathbf{f}(\cdot) = (f_1(\cdot), f_2(\cdot), \ldots, f_d(\cdot)) : \mathcal{X} \to \mathbb{R}^d, \tag{1}$$

where each real-valued function $f_j(\cdot)$, for $j = 1, \ldots, d$, has common domain $\mathcal{X} \subset \mathbb{R}^p$. Seldom will all functions $f_j(\cdot)$ be simultaneously minimized by the same input setting. Hence, dominance is used to identify the best compromises between competing objectives.

Definition 1 (Pareto dominance)**.** *The objective point* $\mathbf{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d$ *(weakly) dominates the point* $\mathbf{w} = (w_1, \ldots, w_d) \in \mathbb{R}^d$ *(denoted* $\mathbf{v} \succeq \mathbf{w}$*) if* $v_i \leq w_i$ *for all dimensions* $i = 1, \ldots, d$*. If at least one of these inequalities is strict, we say* $\mathbf{v}$ *strictly dominates* $\mathbf{w}$ *(denoted* $\mathbf{v} \succ \mathbf{w}$*).*

For example, consider Figure 1c, which shows the image of a biobjective function. The dashed segment is the set of all solutions not dominated by any other image point. Generally speaking, the set of all nondominated image points (e.g. the dashed segment in Figure 1c) of a multiobjective $\mathbf{f}(\cdot)$ is called its Pareto Front (PF), while the set of all inputs in $\mathcal{X}$ that produce the PF is called the Pareto Set (PS) (e.g. the interval $[0.25, 0.75]$ in Figures 1a and 1b). The goal of MO is to estimate the PF and PS of a multiobjective $\mathbf{f}(\cdot)$ and to quantify the uncertainty of these estimates.

When the function cannot be explicitly evaluated or where the number of evaluations is limited, statistical methods can be helpful. A common strategy in computer experiments for PF and PS estimation is to approximate $\mathbf{f}(\cdot)$ by a surrogate model trained on a small number of evaluated points and perform inference on this fitted surrogate model (e.g. see Knowles, 2006; Zhang et al., 2009; Emmerich et al., 2011; Svenson, 2011). Binois et al. (2015) achieve PF estimation and uncertainty quantification (UQ) by fitting a GP and using concepts from the theory of random sets (Molchanov, 2005). However, Binois et al. (2015)'s approach to PF estimation introduces three layers of approximation: emulating $\mathbf{f}(\cdot)$ with a GP, approximating conditional simulations of the trained GP by discretizing the input space into a finite number of points, and finding the PF of these approximate GP realizations. Furthermore, no work in general has been done to quantify the uncertainty of PS estimation with GPs.

An increasingly popular alternative to the GP for emulating single-output simulators is the Bayesian Additive Regression Trees (BART) model introduced by Chipman et al. (2010) (CGM). BART partitions the input space into hyperrectangles and applies a constant mean

model to each hyperrectangle. Unlike GP, BART can capture nonstationarity, avoids $O(n^3)$ matrix decompositions during fitting, and typically has fewer restrictive assumptions, which makes BART feasible in a wider range of scenarios if enough training samples are provided. In particular, BART is well-suited to problems with large datasets (Pratola et al., 2014), which is a common feature in Industry 4.0 applications. Chipman et al. (2012) use BART to estimate a single-output simulator's global minimum. To our knowledge, no one has used BART to perform multiobjective optimization.

Our primary contribution is to estimate the PF and PS of a function $\mathbf{f}(\cdot)$ with a multiple-output BART emulator and to quantify the uncertainty of these estimates. We find the exact PF and PS of a multiple-output BART regression function, which allows us to avoid the second and third approximation layers of Binois et al. (2015)'s GP approach. Furthermore, we perform UQ on PF and PS estimation using the random sets theory of Binois et al. (2015) as well as our extension of the depth approach in López-Pintado and Romo (2009); Sun et al. (2012); Whitaker et al. (2013).

The paper is outlined as follows. Section 2 reviews BART and introduces BART with multiple outputs. Section 3 establishes how to find the image, PF, and PS of a multiple-output BART function. In Section 4, we derive UQ measures for BART-based PF and PS estimates. In Section 5 we perform simulation studies, comparing our approach to the popular GP approach. In Section 6, we demonstrate our BART-based methodology to our engineering application. Section 7 concludes the paper with a discussion. Proofs of stated theorems can be found in the Supplementary Material.

2 BART Modeling

We observe data $\mathscr{D} := \{(\mathbf{y}(\mathbf{x}_i), \mathbf{x}_i)\}_{i=1}^n$, where each output $\mathbf{y}(\mathbf{x}) = (y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_d(\mathbf{x})) \in \mathbb{R}^d$ is assumed to be a realization of the random variable

$$\mathbf{Y}(\mathbf{x}_i) = \mathbf{f}(\mathbf{x}_i) + \boldsymbol{\epsilon}_i, \tag{2}$$

where each $y_j(\cdot)$ has common domain $\mathcal{X} \subset \mathbb{R}^p$, $\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_n \overset{iid}{\sim} N_d(\mathbf{0}, \boldsymbol{\sigma}^2 I_d)$, and $\boldsymbol{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_d^2)$. We assume the domain $\mathcal{X}$ is a $p-$dimensional bounded hyperrectangle.

To make inference about the unknown function $\mathbf{f}(\cdot)$, we approximate each marginal $f_j(\cdot)$, for $j = 1, \ldots, d$, independently by a sum of m regression trees. That is, we independently fit a BART model to each marginal data set $\mathscr{D}_j := \{(y_j(\mathbf{x}_i), \mathbf{x}_i)\}_{i=1}^n$. Hence, the approximation for $\mathbf{f}(\cdot)$ consists of d sums of m regression trees.

2.1 Single-output BART

For notational and conceptual ease, we assume in this section (until Section 2.2) that $d = 1$ and suppress output-dimension indexing, which results in the following simplified notation: $f(\cdot) := f_1(\cdot)$ and $y(\cdot) := y_1(\cdot)$. We approximate $f(\cdot)$ by a sum of m regression trees:

$$f(\mathbf{x}) \approx \sum_{t=1}^{m} g(\mathbf{x}; \mathcal{T}_t, \mathcal{M}_t), \tag{3}$$

where each $g(\cdot; \mathcal{T}_t, \mathcal{M}_t) : \mathcal{X} \to \mathbb{R}$ denotes a regression tree function and the parameters $(\mathcal{T}_t, \mathcal{M}_t)$ are tree t's topology and terminal node parameters for $t = 1, \ldots, m$. The param-

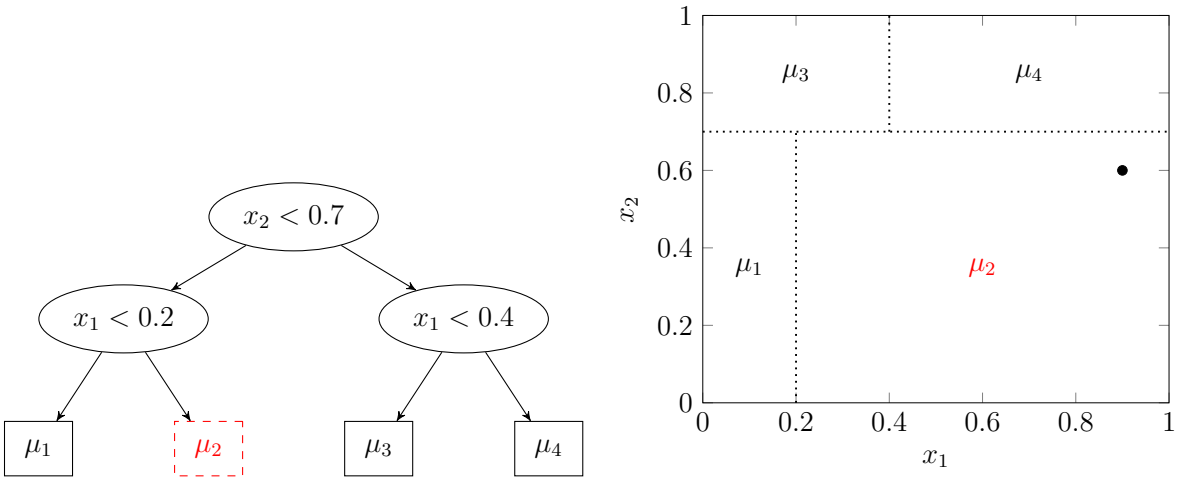


Figure 2: Two different views of the same regression tree. Left: binary tree representation. Right: input space representation.

eters $\{\mathcal{T}_t, \mathcal{M}_t\}_{t=1}^m$ are given a prior distribution as in CGM.

Each regression tree function in Equation 3 partitions the input space and assigns a scalar value to each subregion (see Figure 2 for an example). Each function is defined by a binary tree; the binary tree's internal node parameters construct the function's partition while the binary tree's terminal node parameters are the values the function assigns to each subregion. To form the partition, each internal node contains a boolean split rule defined as an inequality involving a (variable, cutpoint) pair as seen in Figure 2. Starting at the root node, if an input $\mathbf{x}$ satisfies the split rule, it will travel to the node's left child; otherwise $\mathbf{x}$ will travel to the right child. The input $\mathbf{x}$ will continue to traverse through the tree until it reaches a terminal node. This terminal node's parameter is the value that the tree assigns to every input that follows the path to this terminal node. The union of all such inputs is a hyperrectangular subregion of the input space. Hence, the tree assigns this terminal node parameter to this entire hyperrectangular subregion.

Example 1. The binary tree in Figure 2 defines how the regression tree function partitions

the 2−dimensional input space and assigns a value to each hyperrectangle. Starting at the root node, the input $\mathbf{x}^* = (x_1^*, x_2^*) = (0.9, 0.6)$ satisfies the split rule $x_2^* < 0.7$ and hence moves to the left child. Because $\mathbf{x}^*$ does not satisfy $x_1^* < 0.2$, it moves to the right child. Because this node is terminal, the regression tree assigns the value μ_2 to $\mathbf{x}^*$. In fact, the regression tree function assigns μ_2 to any input in the rectangle $[0.2, 1] \times [0, 0.7)$. ■

Returning to Equation 3, let $\mathcal{T}_t$ denote the set of parameters associated with tree t's split rules (i.e. the split variable and cutpoint for each internal node) and topology. Let $\mathcal{M}_t$ denote the set $\{\mu_{t,k} : k = 1, \ldots, |\mathcal{M}_t|\}$ of parameters associated with $\mathcal{T}_t$'s terminal nodes. If the parameters $\theta := \{(\mathcal{T}_t, \mathcal{M}_t)\}_{t=1}^m$ have been established, we let the function

$$\mathcal{E}(\cdot; \theta) := \sum_{t=1}^{m} g(\cdot; \mathcal{T}_t, \mathcal{M}_t) = \sum_{t=1}^{m} \sum_{k=1}^{|\mathcal{M}_t|} \mu_{t,k} \mathbb{I}_{\mathbf{R}_{t,k}}(\cdot) \tag{4}$$

denote the sum-of-trees approximation in Equation 3, where $\mathbf{R}_{t,k} \subset \mathcal{X}$ denotes the hyperrectangular subregion associated with $\mathcal{T}_t$'s terminal node that contains parameter $\mu_{t,k}$.

2.1.1 BART's Bayesian model

Equation 3 is specified by the parameters $\{(\mathcal{T}_t, \mathcal{M}_t)\}_{t=1}^m$ and σ^2. Hence, a trained BART model will sample from the posterior distribution $\pi(\Theta|\mathscr{D}) \propto L(\mathscr{D}|\Theta)\,\pi(\Theta)$, where $\Theta = \{(\mathcal{T}_1, \mathcal{M}_1), (\mathcal{T}_2, \mathcal{M}_2), \ldots, (\mathcal{T}_m, \mathcal{M}_m), \sigma^2\}$ are the parameters, $\mathscr{D} = \{(y(\mathbf{x}_i), \mathbf{x}_i)\}_{i=1}^n$ is the observed data, $L(\mathscr{D}|\Theta) \propto \sigma^{-n} \exp\big(-\frac{1}{2\sigma^2} \sum_{i=1}^n (y(\mathbf{x}_i) - \mathcal{E}(\mathbf{x}_i; \theta))^2\big)$ is the Gaussian likelihood, and $\pi(\Theta)$ is the prior. CGM describe the Gibbs sampling scheme used for posterior sampling. We describe their prior specification here because we ultimately deviate from their default hyperparameter values for our multiobjective optimization problem.

CGM specify the full prior $\pi(\Theta)$ by assuming: (1) $(\mathcal{T}_1, \mathcal{M}_1), \ldots, (\mathcal{T}_m, \mathcal{M}_m)$, and σ^2 are mutually independent, (2) $\mathcal{T}_1, \ldots, \mathcal{T}_m$ are identically distributed, and (3) all terminal node parameters $\mu_{t,k}|\mathcal{T}_t$ are conditionally i.i.d. (independent and identically distributed) for all m trees. These independence assumptions simplify the problem of specifying the prior $\pi(\Theta)$ to stating priors $\pi(\mathcal{T}_t)$, $\pi(\mu_{tk}|\mathcal{T}_t)$, and $\pi(\sigma^2)$.

The $\pi(\mathcal{T}_t)$ prior can be broken down into three components: tree depth, split variable at each internal node, and cutpoint value at each internal node. We leave details of tree depth prior specification to CGM and Chipman et al. (1998). For the split variable prior, CGM use a discrete uniform distribution over the set of available variables. For the prior on the cutpoint value of any given split variable, this paper uses a discrete uniform distribution of 30 values over the range of the observed input values.

For $\pi(\mu_{tk}|\mathcal{T}_t)$, CGM use the Gaussian distribution $N(\mu_\mu, \sigma_\mu^2)$ with hyperparameters μ_μ and σ_μ^2. Under the sum-of-trees model, the prior on $\mathbb{E}[Y(\mathbf{x})]$ becomes $N(m\mu_\mu, m\sigma_\mu^2)$. For convenience, CGM center and rescale the output data so that the lowest and highest observed transformed response values are, respectively, $y_{min} = -0.5$ and $y_{max} = 0.5$. The prior on $\mathbb{E}[Y(\mathbf{x})]$ is taken to be $N(0, m\sigma_\mu^2)$, where σ_μ^2 is chosen so that $m\sigma_\mu^2 = 1/(4\kappa^2)$ for some specified value of κ. Instead of using CGM's default value of $\kappa = 2$, Chipman et al. (2012) use $\kappa = 1$ in their single-objective optimization setting to reduce the smoothness of the response, which allows BART to produce more pronounced optima. For similar reasons, we also use $\kappa = 1$ and $m = 30$ trees for our applications in Section 5, but other situations may call for different (κ, m) values. We also set the minimum number of observations allowed in each terminal node to ten.

For $\pi(\sigma^2)$, CGM use the scaled inverse chi-square distribution $\sigma^2 \sim \text{Scale-}\chi^{-2}(\nu, \lambda)$ with values $\nu = 3$ and $\lambda = 0.01^2$ chosen to induce a prior mean of 0.0003 (see CGM for details of ν and λ selection). However, we find that the value of hyperparameter κ more strongly influences the smoothness of the response.

2.2 Multiple-output BART

Now we allow $d > 1$. We then approximate $\mathbf{f}(\cdot)$: $\mathcal{X} \to \mathbb{R}^d$ from Equation 1 by the function

$$\boldsymbol{\mathcal{E}}(\cdot; \boldsymbol{\theta}) = (\mathcal{E}(\cdot; \theta_1), \mathcal{E}(\cdot, \theta_2), \ldots, \mathcal{E}(\cdot; \theta_d)) : \mathcal{X} \to \mathbb{R}^d \tag{5}$$

where $\boldsymbol{\theta} = \{\theta_j\}_{j=1}^d$, $\mathcal{E}(\cdot; \theta_j)$, for $j = 1, \ldots, d$, is defined in Equation 4, and each θ_j results from a posterior draw of a single-output BART model fit to the marginal data set $\mathscr{D}_j$. That is, we model the d outputs as d independent BART models.

3 Multiobjective optimization

Our two-step algorithm to find the PF and PS of $\boldsymbol{\mathcal{E}}(\cdot; \boldsymbol{\theta})$ is as follows:

1. (Section 3.1) Find the image $\boldsymbol{\mathcal{E}}(\mathcal{X}; \boldsymbol{\theta})$. Find each image point's corresponding input hyperrectangle.

2. (Section 3.2) Find all nondominated points in the image $\boldsymbol{\mathcal{E}}(\mathcal{X}; \boldsymbol{\theta})$.

Step 2 produces the desired PF. The desired PS is the union of all input hyperrectangles (found in Step 1) that yields a point in the PF.

3.1 Find image of multivariate BART function

Any single-objective BART function has a finite image (see Equation 4). Thus, the multi-objective BART function $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$ also has a finite image. Furthermore, we may find these image points and their corresponding input hyperrectangles using the parameter values $\boldsymbol{\theta}$.

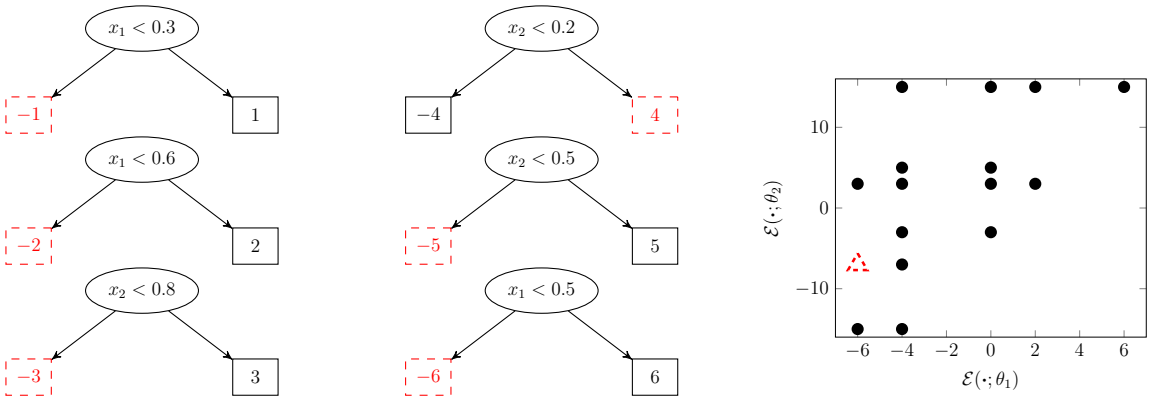


Figure 3: A biobjective function $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta}) = (\mathcal{E}(\cdot;\theta_1), \mathcal{E}(\cdot;\theta_2))$ with tree ensembles with parameters θ_1 (left) and θ_2 (middle). Right: all image points of $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$.

Example 2. Here, we find the image of the biobjective function $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta}) = (\mathcal{E}(\cdot;\theta_1), \mathcal{E}(\cdot;\theta_2))$ shown in Figure 3, where $\mathcal{X} = [0.1]^2$. The input $\mathbf{x}^* = (0.1, 0.3)$ belongs to one terminal node per tree. These six terminal nodes (each denoted by a dashed border) correspond to the image point $\boldsymbol{\mathcal{E}}(\mathbf{x}^*;\boldsymbol{\theta}) = (-6, -7)$. Conversely, this image point corresponds to any input that belongs to these six terminal nodes. If we obtain every one-terminal-node-per-tree combination, we can find the finite image of $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$ and the corresponding input hyperrectangles for each image point.

However, not all such terminal-node combinations are possible and produce an image point. Because both $x_1 < 0.3$ and $x_1 \geq 0.6$ cannot be simultaneously satisfied, an input point cannot belong to both the left and right terminal nodes of, respectively, the top and middle trees of ensemble $\mathcal{E}_1$. Despite the $2^6 = 64$ possible one-terminal-node-per-tree

combinations, $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$ produces only 16 image points (shown in Figure 3). ■

Theorem 1 describes a similar process for a general d-objective BART function $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$. That is, any arbitrary $\mathbf{x} \in \mathcal{X}$ belongs to a one-terminal-node-per-tree combination (dm total terminal nodes) that corresponds to the image point $\boldsymbol{\mathcal{E}}(\mathbf{x};\boldsymbol{\theta})$. Conversely, every image point of the BART function corresponds to a one-terminal-node-per-tree combination that contains a subset of $\mathcal{X}$. If we obtain every such combination of dm terminal nodes, we can find the image and corresponding input hyperrectangles of the d-objective BART function.

Theorem 1. *Using the linear-combination representation of single-objective BART functions in Equation 4, any d-objective BART function in the form of Equation 5 can be written as a linear combination of indicator functions of hyperrectangles:*

$$\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta}) = \sum_{q \in B_{\boldsymbol{\mathcal{E}}}} \boldsymbol{\alpha}_q \mathbb{I}_{\mathbf{R}_q}(\cdot),$$

where the set $B_{\boldsymbol{\mathcal{E}}}$ indexes the valid one-terminal-node-per-tree combinations in $\boldsymbol{\theta}$, each $\boldsymbol{\alpha}_q \in \mathbb{R}^d$ is an image point of $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$, and the set of hyperrectangles $\{\mathbf{R}_q\}_{q \in B_{\boldsymbol{\mathcal{E}}}}$ partitions $\mathcal{X}$.

3.2 Find PF and PS of multiple-output BART function

After finding the image of a multiobjective BART function $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$, we find its set of nondominated points. Because this image is finite, we can use an efficient recursive algorithm from Kung et al. (1975), which finds the nondominated points in a set V of $d-$dimensional vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{|V|}$, where $|V|$ denotes the cardinality of V. In our setting, each vector in V is an image point of $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$. For simplicity, we describe only one of these algorithms

(see the Supplementary Material), where Kung et al. (1975) shows for $d > 3$ that the time complexity has an upper bound of $O(|V|(\log_2 |V|)^{d-2})$. We can still apply this algorithm when $d = 2$ or $d = 3$, but this upper bound no longer applies in these cases.

4 Uncertainty quantification

We quantify the uncertainty of $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$'s PF by independently drawing $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)}$ in the same way that $\boldsymbol{\theta}$ is drawn as described in Equation 5. For each $i = 1, \ldots, N$, we find the resulting function $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta}^{(i)})$ and conditional PF (CPF) $c^{(i)}$. This section details two UQ approaches using the sample of N CPFs. Both approaches use dominated point set closures (DPSCs): for $i = 1, \ldots, N$, define the DPSC $A^{(i)}$ of a CPF $c^{(i)}$ to be the closure of the set of points dominated by at least one point in $c^{(i)}$. That is, $A^{(i)} := \{\mathbf{y}' \in \mathcal{Y} \mid \mathbf{y}' \preceq \mathbf{y} \text{ for at least one } \mathbf{y} \in c^{(i)}\}$, where $\mathcal{Y} \subset \mathbb{R}^d$ is the smallest compact hyperrectangle that contains every objective point in the training set $\mathscr{D}$. Figure 4 shows examples of DPSCs.

4.1 Random-sets approach

da Fonseca and Fonseca (2010) treat each DPSC as a realization of an attained set, which is a random closed set whose probability distribution is characterized by its attainment function (see Molchanov (2005) for a full treatment of random closed sets).

Definition 2 (Attained set and attainment function)**.** *The DPSC of a random PF is the set attained by the random PF. The attainment function* $\alpha_{\mathcal{A}} : \mathbb{R}^d \to [0, 1]$ *of an attained set* $\mathcal{A}$ *is defined as* $\alpha_{\mathcal{A}}(\mathbf{y}) := \mathbb{P}(\mathbf{y} \in \mathcal{A})$ *for every point* $\mathbf{y} \in \mathbb{R}^d$.

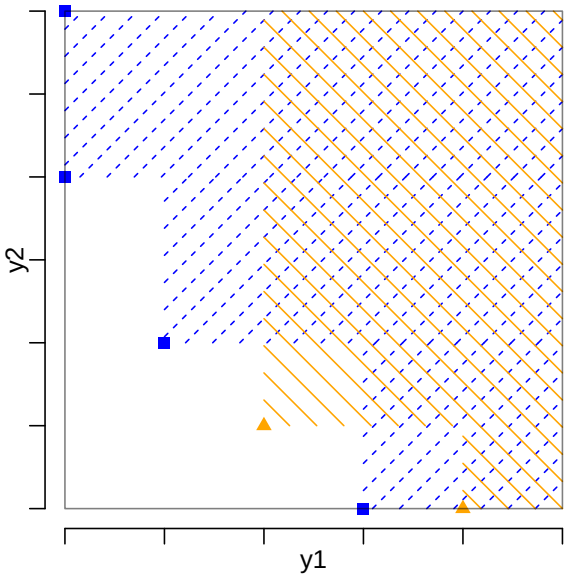


Figure 4: Example of 2 CPFs (triangles and squares) and the corresponding DPSCs (areas with hatched lines) in objective space $\mathcal{Y}$ (enclosed by solid gray border).

Example 3. Consider Figure 4 and suppose an attained set $\mathcal{A}$ has 0.5 probability of being the DPSC created from the square points and is otherwise the DPSC created from the triangle points. The attainment function $\alpha_{\mathcal{A}}$ of $\mathcal{A}$ is then defined as follows:

$$\alpha_{\mathcal{A}}(\mathbf{y}) = \begin{cases} 0, & \mathbf{y} \text{ is not dominated by any square or triangle point} \\ 1, & \mathbf{y} \text{ is dominated by both a square and a triangle point} \\ 1/2, & \text{otherwise.} \end{cases}$$

■

4.1.1 PF estimation

If an attained set $\mathcal{A}$ has uncountably many possible set realizations, its attainment function may be difficult to formulate. We may instead estimate the attainment function using an

empirical version, which takes on values in the set $\{0, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N-1}{N}, 1\}$.

Definition 3 (Empirical Attainment Function)**.** *Let $A^{(1)}, A^{(2)}, \ldots, A^{(N)}$ be realizations of the attained set $\mathcal{A}$ on $\mathcal{Y}$. The empirical attainment function $\hat{\alpha}_N : \mathcal{Y} \to [0, 1]$ is defined to be the fraction of attained set realizations that contain its argument: $\hat{\alpha}_N(\cdot) := \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{A^{(i)}}(\cdot)$.*

This paper quantifies the uncertainty of the CPFs using an $\alpha_{RS}\%$ PF UQ point cloud, which is the set of all CPF points whose empirical attainment function value is between $0.5 - \alpha_{RS}/2$ and $0.5 + \alpha_{RS}/2$, where $0 < \alpha_{RS} < 1$. That is, this UQ point cloud is the set of CPF points dominated by some proportion of CPFs between $(0.5 - \alpha_{RS}/2, 0.5 + \alpha_{RS}/2)$. Obtaining this point cloud requires evaluating the function $\hat{\alpha}_N(\cdot)$ at every CPF point, which means checking the condition "$\mathbf{y} \in A^{(i)}$" for all CPF points $\mathbf{y}$ and all $i = 1, \ldots, N$. A single "$\mathbf{y} \in A^{(i)}$" check, which requires checking $\mathbf{y}$'s dominance relationship with every point in the CPF $c^{(i)}$, takes $\mathcal{O}(dn_i)$ time, where n_i is the number of points in $c^{(i)}$. Checking "$\mathbf{y} \in A^{(i)}$" for all CPF points $\mathbf{y}$ and all $i = 1, \ldots, N$ then takes $\mathcal{O}\big(d(\sum_{i=1}^{N} n_i)^2\big)$ time.

4.1.2 PS estimation

Recall from Section 3.1 that each image point of a multiple-output BART regression function $\boldsymbol{\mathcal{E}}(\cdot; \boldsymbol{\theta})$ corresponds to a partitioning hyperrectangle. Hence, the PS of $\boldsymbol{\mathcal{E}}(\cdot; \boldsymbol{\theta})$ corresponds to a collection of hyperrectangles. This paper thus quantifies the uncertainty of conditional PSs to be the union of the hyperrectangles corresponding to the points in the PF UQ point cloud: $\widehat{\boldsymbol{\mathcal{P}}_{\boldsymbol{\mathcal{X}}}} := \bigcup_{\mathbf{y} \in \widehat{\boldsymbol{\mathcal{P}}_{\boldsymbol{\mathcal{F}}}}} \boldsymbol{\mathcal{E}}^{-1}(\mathbf{y}; \boldsymbol{\theta})$, where $\widehat{\boldsymbol{\mathcal{P}}_{\boldsymbol{\mathcal{F}}}}$ is a PF UQ point cloud and $\boldsymbol{\mathcal{E}}^{-1}(\mathbf{y}; \boldsymbol{\theta})$ is the hyperrectangle corresponding to the objective point $\mathbf{y}$.

4.2 Band depth approach

A second approach to quantify the variability of CPFs $c^{(1)}, c^{(2)}, \ldots, c^{(N)}$ (with associated DPSCs $A^{(1)}, A^{(2)}, \ldots, A^{(N)}$) is to order them using a graph-based notion of depth. The idea is to measure the centrality of a curve with respect to either a set of curves or a population distribution. A sample of curves can then be ordered from the center outward, where the "deepest" curve would be the "median" curve. López-Pintado and Romo (2009) introduce the concept of band depth for univariate functions. Whitaker et al. (2013) generalize this band depth definition to operate on sets, which we use to order $c^{(1)}, c^{(2)}, \ldots, c^{(N)}$. We say that a CPF $c^{(i)}$ lies in the band delimited by two CPFs $c^{(j)}$ and $c^{(k)}$ if and only if $\left[A^{(j)} \cap A^{(k)}\right] \subseteq A^{(i)} \subseteq \left[A^{(j)} \cup A^{(k)}\right]$. We denote this relationship by $c^{(i)} \subseteq^* B(c^{(j)}, c^{(k)})$. Figure 5 shows an example of a band delimited by two CPFs. We now define the band depth of $c^{(i)}$ to be the proportion of bands delimited by two of the N CPFs containing $c^{(i)}$ in the $\subseteq^*$ sense. That is, given CPFs $c^{(1)}, c^{(2)}, \ldots, c^{(N)}$, the band depth of CPF $c^{(i)}$ is $BD_N(c^{(i)}) = \binom{N}{2}^{-1} \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} \mathbb{I}\big(c^{(i)} \subseteq^* B(c^{(j)}, c^{(k)})\big)$.

4.2.1 Modified band depth

Whitaker et al. (2013) note that if N is small and the CPFs strongly vary in shape, this band depth definition can produce many zero-depth CPFs. To circumvent this issue for univariate, single-output functions, López-Pintado and Romo (2009) define a modified band depth that measures the proportion of a function's graph that is in the band. Sun et al. (2012) later introduce an algorithm that efficiently computes this modified band depth. We extend this approach by creating an algorithm that efficiently computes modified band

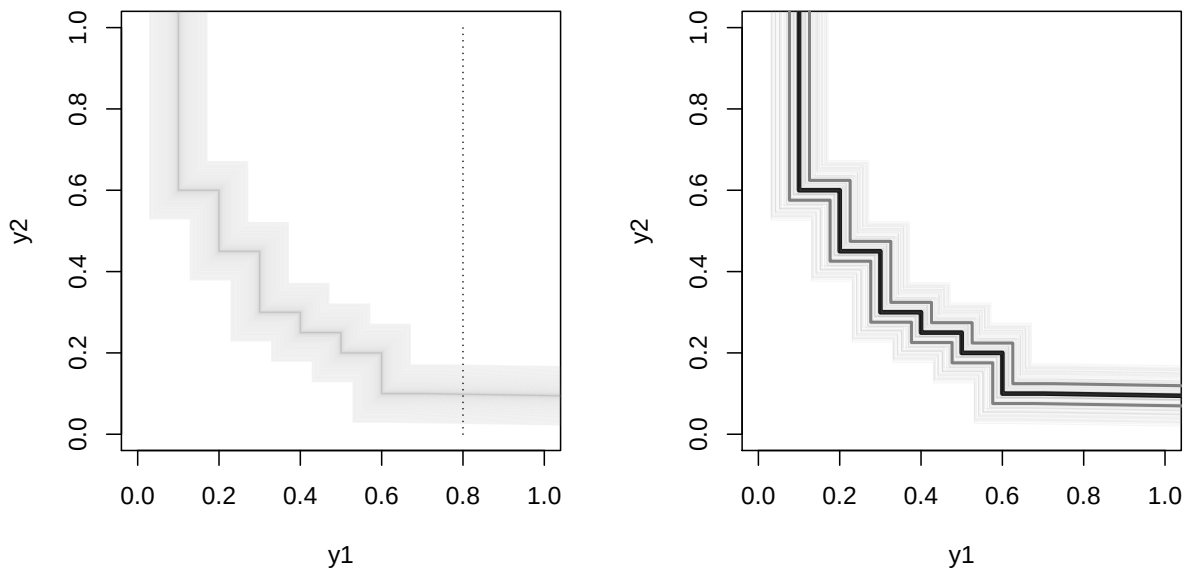


Figure 5: The lower-left boundaries of $N = 101$ DPSCs. Left: faint gray curves correspond to the N CPFs. Right: thick black curve corresponds to the deepest CPF. Thick gray curves correspond to the 50% deepest CPFs corresponding to $\alpha_{MBD} = 0.5$.

depth for two-dimensional CPFs.

Example 4. We motivate our algorithm through an illustrative example in which we compute the depth of an arbitrary CPF, denoted as $c^{(7)}$, among the $N = 101$ CPFs in Figure 5, where we assume $\mathcal{Y} = [0,1]^2$. First, we define and compute $c^{(7)}$'s depth at the vertical dotted line $y_1 = 0.8$, which we denote as $d^{(7)}|_{y_1=0.8}$. Let $A^{(7)}|_{y_1=0.8} = \min\{y_2 \in [0,1] : (0.8, y_2) \in A^{(7)}\}$ be the minimum y_2 value of the lower-left boundary of DPSC $A^{(7)}$ that intersects the vertical line $y_1 = 0.8$. Analogous to the $\subseteq^*$ relation, a pair of CPFs $c^{(j)}$ and $c^{(k)}$ is said to contain CPF $c^{(7)}$ at the line $y_1 = 0.8$ if and only if $A^{(7)}|_{y_1=0.8}$ is a value loosely between $A^{(j)}|_{y_1=0.8}$ and $A^{(k)}|_{y_1=0.8}$. Then $d^{(7)}|_{y_1=0.8}$ is defined to be the fraction of pairs of the N CPFs that contain $c^{(7)}$. We note that the numerator, i.e. the number of CPF pairs that contain $c^{(7)}$, is (`# CPFs below or equal to` $c^{(7)}$ `at` $y_1 = 0.8) \times$ (`# CPFs above or equal to` $c^{(7)}$ `at` $y_1 = 0.8) - 1$, where we subtract one to avoid

counting the band delimited by $c^{(7)}$ with itself. Both values in the product, i.e. both "`# CPFs`" values, are determined by the rank of $c^{(7)}$ according to its $A^{(\cdot)}|_{y_1=0.8}$ value.

We may repeat the process above for any vertical line, e.g. $y_1 = s$, to obtain the depth $d^{(7)}|_{y_1=s}$ of CPF $c^{(7)}$ at $y_1 = s$. Similarly, we may easily alter the process above to obtain depth $d^{(7)}|_{y_2=t}$ of CPF $c^{(7)}$ at any *horizontal* line, e.g. $y_2 = t$. We can then approximate the depth of CPF $c^{(7)}$ by: $0.5q^{-1}\sum_{j=1}^{q} d^{(7)}|_{y_1=t_j} + 0.5q^{-1}\sum_{j=1}^{q} d^{(7)}|_{y_2=t_j}$, where $t_j := (j-1)/(q-1)$ and q is some large number of lines per output dimension. ■

Now we introduce our two-dimensional extension, which generalizes the process above to find the depth of all N CPFs. Step 1 creates the $q = 201$ lines per output dimension while Step 2 computes the y_2 and y_1 intersection values for all N CPFs. At each of these lines from Step 1, Step 3 ranks the CPFs while Step 4 computes for each CPF the number of pairs of CPFs that contain it. Step 5 then computes the depth of each CPF. This algorithm has an asymptotic time complexity of $\mathcal{O}(2qN \log N)$, where the 2 comes from the output dimension ($d = 2$). The algorithm can be found in the Supplemental Material.

4.2.2 PF and PS estimation

Our α_{MBD}% PF UQ point cloud for the depth approach is the union of the α_{MBD} deepest CPFs, where $0 < \alpha_{MBD} < 1$. Our PS UQ region is then the union of the hyperrectangles corresponding to the points in the PF UQ point cloud.

4.3 Comparing time complexity

We recall that to obtain a PF UQ point cloud from N CPFs, the random-sets approach takes $\mathcal{O}\big(d(\sum_{i=1}^{N} n_i)^2\big) = \mathcal{O}\big(d(N\bar{n})^2\big) = \mathcal{O}(dN^2\bar{n}^2)$ time where $\bar{n} := N^{-1}\sum_{i=1}^{N} n_i$ is the mean number of points in each CPF (see Section 4.1.1), while the depth approach takes $\mathcal{O}(2qN \log N)$ time (see Section 4.2.1). If we fix $d = 2$, q, and $\bar{n}$ while increasing N, the random-sets approach slows more quickly than does the depth approach. However, it is unclear how to extend our depth algorithm to a scenario where $d > 2$, whereas the random-sets approach can be readily applied to a $d > 2$ scenario.

Though we fixed q and $\bar{n}$ in this comparison, increasing training size may result in more information about the PF $\mathcal{P}_{\mathcal{F}}$, which may induce more refined BART PF estimates $\widehat{\mathcal{P}_{\mathcal{F}}}$ and hence increase $\bar{n}$. If $\bar{n}$ does increase, it may behoove us to increase q to maintain the fidelity of the ranks of the curves.

5 Simulation study

Simulation settings. Given data from Equation 2 and $\mathbf{f}(\cdot)$'s PF and PS, this section explores how sample size and measurement error magnitude affect the accuracy of:

Q.1 BART's PF point clouds and GP's PF approximate point clouds.

Q.2 BART's PS point clouds.

Q.3 Depth approach and random sets approach to uncertainty quantification.

We explore each possible combination of the following different parameter settings: $n \in \{32p, 64p\}$, and $\sigma_j^2 \in \{0, 0.1\texttt{Var}(f_j(\mathbf{X})), 0.25\texttt{Var}(f_j(\mathbf{X}))\}$ for all $j = 1, \ldots, d$, where

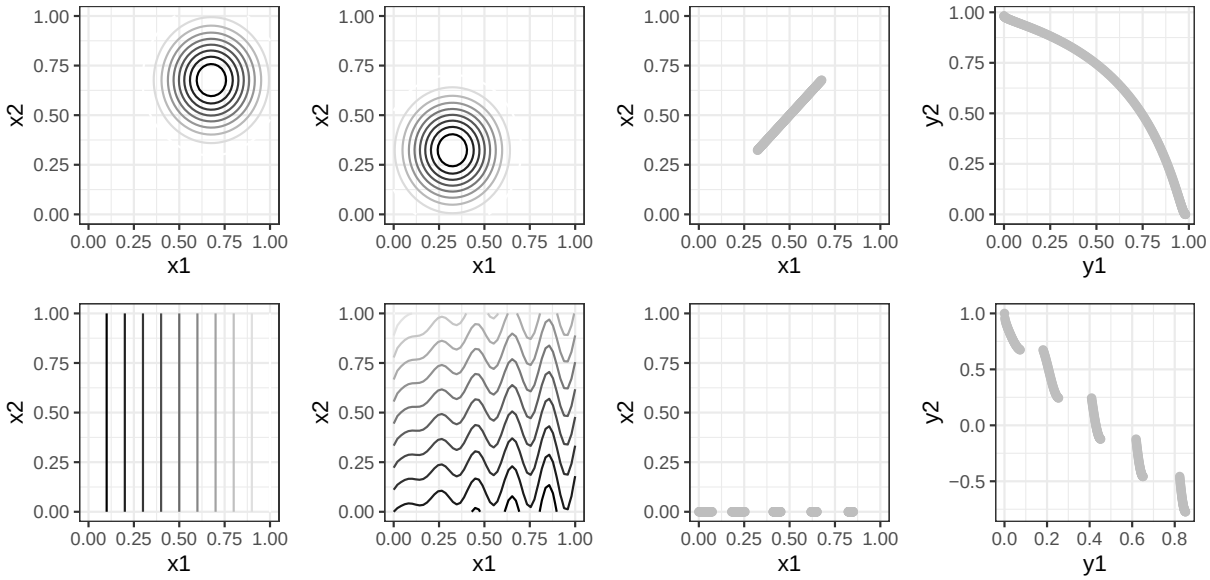


Figure 6: MOP2 function (top row) and ZDT3 function (bottom row). From left to right, the first (second) column shows the contour of f_1 (f_2), where darker contour lines indicate lower objective values. The third (fourth) column shows the PS (PF) of the function.

$\mathbf{X} = (X_1, \ldots, X_p)$ and each $X_i \overset{iid}{\sim} U(0,1)$. For each simulation setting, we generate 100 data sets $\mathscr{D} := \{(\mathbf{y}(\mathbf{x}_i), \mathbf{x}_i)\}_{i=1}^n$ from Equation 2, where each input point $\mathbf{x}_i$ comes from an $n-$point maximin LHS on $[0,1]^p$ from Edwin van Dam et al. (2015). To each data set, we fit two models: a multiple-output BART model as described in Section 2.2 and a multiple-output GP model produced from fitting an independent single-output GP to each marginal data set $\mathscr{D}_j$ for $j = 1, \ldots, d$. For each model we produce 500 posterior samples. We use the OpenBT implementation of BART with default parameter settings unless otherwise stated in Section 2.1.1. For the GP method, we use the `km` and `simulate` functions of Roustant et al. (2012)'s DiceKriging package with error variance set to the noise variance of the scenario. Because noise variance is not known in most applications, these GP fits can be thought of as idealized GP fits. For UQ, we use $\alpha_{RS} = 0.25$ and $\alpha_{MBD} = 0.5$.

We examine three test functions, MOP2, ZDT3, and DTLZ2, defined in the Supple-

mentary Material. Figure 6 shows the MOP2 function ($p = d = 2$) to be the simplest of the three. DTLZ2's ($p = 4$ and $d = 2$) PF and PS (not shown) are similar to MOP2's while ZDT3 ($p = d = 2$) has a disconnected PF and PS. The methodology in Sections 3.2 and 4 is invariant to shifts and scales of inputs or outputs. However, we shift and scale the input space to be $\mathcal{X} = [0, 1]^p$ and each objective to have range $[0, 1]$ to render our performance metrics in Section 5.1 unaffected by scale differences between objective functions.

5.1 Performance metrics

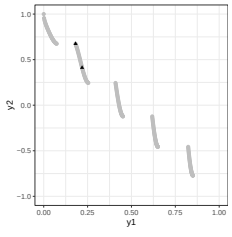
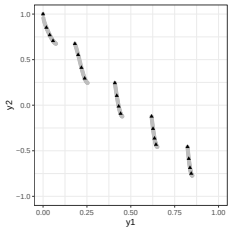
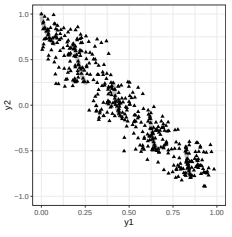
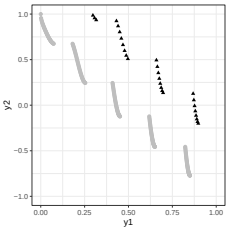
	undercoverage	good coverage	overcoverage	biased coverage
				
$d(\widehat{\mathcal{P}}, \mathcal{P})$	small	small	large	large
$d(\mathcal{P}, \widehat{\mathcal{P}})$	large	small	small	large

Table 1: Each plot shows a different PF or PS estimate $\widehat{\mathcal{P}}$ (black triangles) and PF or PS $\mathcal{P}$ (gray line or lines). These three examples illustrate the need for two different metrics (the second and third row of the table) to measure the accuracy of a PF or PS estimate.

Here we define two metrics to assess a point cloud's performance. Generally speaking, let $d_0(\mathbf{a}, \mathcal{B}) = \min_{\mathbf{b} \in \mathcal{B}} \|\mathbf{a} - \mathbf{b}\|_2$ be the distance from a point $\mathbf{a}$ to a point set $\mathcal{B}$. Dubuisson and Jain (1994) defines $d(\mathcal{A}, \mathcal{B}) := |\mathcal{A}|^{-1} \sum_{\mathbf{a} \in \mathcal{A}} d_0(\mathbf{a}, \mathcal{B})$ to be the average distance from points in a finite point set $\mathcal{A}$ to $\mathcal{B}$. This function increases in its first argument (i.e. $d(\mathcal{A}_1, \mathcal{B}) \leq d(\mathcal{A}_2, \mathcal{B})$ holds if $\mathcal{A}_1 \subseteq \mathcal{A}_2$), decreases in its second argument (i.e. $d(\mathcal{A}, \mathcal{B}_1) \geq d(\mathcal{A}, \mathcal{B}_2)$ holds if $\mathcal{B}_1 \subseteq \mathcal{B}_2$), and is not symmetric (i.e. $d(\mathcal{A}, \mathcal{B}) \neq d(\mathcal{B}, \mathcal{A})$).

For a point cloud $\widehat{\mathcal{P}}$ and true Pareto set/front $\mathcal{P}$, we define the two quantities

$$\texttt{overcoverage} := d(\widehat{\mathcal{P}}, \mathcal{P}) \quad \text{and} \quad \texttt{undercoverage} := d(\mathcal{P}, \widehat{\mathcal{P}})$$

whose values for four example scenarios are shown in Table 1. Intuitively, the quantity $d(\widehat{\mathcal{P}}, \mathcal{P})$ measures the average distance from a point in $\widehat{\mathcal{P}}$ to $\mathcal{P}$. This quantity punishes the estimates $\widehat{\mathcal{P}_{\mathcal{F}}}$ in both the "overcoverage" and "biased coverage" scenarios for having many points far away from $\mathcal{P}_{\mathcal{F}}$, but at the same time is small for the estimates $\widehat{\mathcal{P}_{\mathcal{F}}}$ in both the "undercoverage" and "good coverage" scenarios because no points are far away from the set $\mathcal{P}_{\mathcal{F}}$. Similarly, the quantity $d(\mathcal{P}, \widehat{\mathcal{P}})$ captures how badly $\widehat{\mathcal{P}}$ misses any points in $\mathcal{P}$. This quantity punishes the estimates $\widehat{\mathcal{P}_{\mathcal{F}}}$ in both the "undercoverage" and "biased coverage" scenarios for not having points close to many of the points in $\mathcal{P}_{\mathcal{F}}$, but at the same time is small for the estimates $\widehat{\mathcal{P}_{\mathcal{F}}}$ in both the "overcoverage" and "good coverage" scenarios because no points in $\mathcal{P}_{\mathcal{F}}$ are far away from at least one point in $\widehat{\mathcal{P}_{\mathcal{F}}}$.

5.2 Simulation results

Figure 7 displays bagplots of the 100 values of $d(\widehat{\mathcal{P}_{\mathcal{F}}}, \mathcal{P}_{\mathcal{F}})$ and $d(\mathcal{P}_{\mathcal{F}}, \widehat{\mathcal{P}_{\mathcal{F}}})$ for each $n = 64p$ simulation scenario (PS plots and similar $n = 32p$ scenario results can be found in the Supplementary Materials). A bagplot extends the common boxplot for two-dimensional outputs and contains three main features analogous to the common univariate median, the box, and the whiskers on a conventional boxplot (Rousseeuw et al., 1999). For visual clarity, we include only two of these features: the depth median, which is the point with the highest possible halfspace depth (Tukey, 1975), and the "bag", which is a polygon that

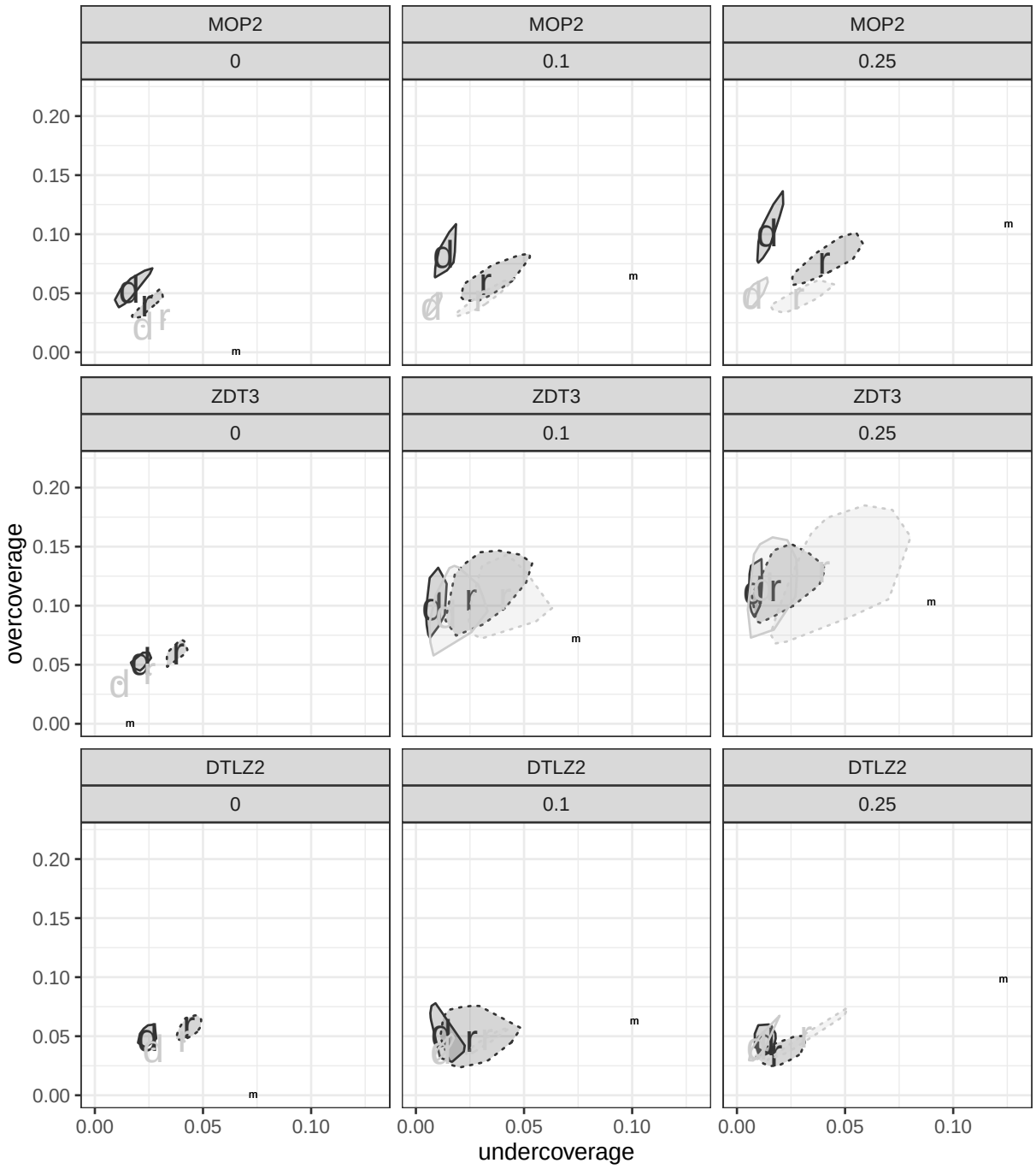


Figure 7: Bagplots of the 100 values of $d(\widehat{\mathcal{P}_{\mathcal{F}}}, \mathcal{P}_{\mathcal{F}})$ and $d(\mathcal{P}_{\mathcal{F}}, \widehat{\mathcal{P}_{\mathcal{F}}})$. Each plot represents a $n = 64p$ simulation scenario where the variance multiplier is a value in $\{0, 0.1, 0.25\}$. Bags with solid (dashed) outline and median labeled 'd' ('r') display depth (random sets) approach. Black (gray) bags display BART (GP) model.

encloses 50% of the points around the depth median.

As another point of comparison, we also plot in Figure 7 the median undercoverage and overcoverage of 100 sets of 10 points randomly selected from the underlying function's PF and perturbed according to the scenario's noise level. These medians are the points labeled 'm'. Each set of 10 randomly selected points from $\mathcal{P}_{\mathcal{F}}$ provides a sense of the undercoverage and overcoverage one can expect from a "well-fitting" statistical model that produces a 10-point CPF. The subsequent perturbation accounts for the quality of the data that a statistical model is trained on. We do not perform this comparison for the PS figures for lack of a natural mapping of observation noise level to input noise level.

To address quantity **Q.1**, we make several observations in Figure 7. We first compare the difficulty of MO between the three functions. The ZDT3 PF point clouds tend to produce higher overcoverage than either the MOP2 or DTLZ2 PF point clouds, which can be explained by ZDT3's disconnected PF. The PF point cloud undercoverage is roughly the same between the three functions. However, the PS point clouds (Supplementary Material, Figures 10b, 11b, and 12b) perform differently between the three functions. The DTLZ2 PS point clouds have the most undercoverage, which may be due to a larger input dimension ($p = 4$). These point clouds also have the most overcoverage, but the overcoverage difference between DTLZ2 and ZDT3 is roughly the same as the overcoverage difference between ZDT3 and MOP2, which again can be explained by ZDT3's PS being disconnected and on the boundary of the input space. From these observations, we conclude that ZDT3 has the most difficult PF to capture while DTLZ2 has the most difficult PS to capture.

We also compare performance between BART and (idealized) GP. In the MOP2 PF

results, the no-noise scenario shows a overcoverage/undercoverage tradeoff between BART and GP while the two noisy $n = 128$ scenarios show GP outperforming BART in both metrics. The DTLZ2 PF results show roughly equal performance between BART and GP. The ZDT3 PF results show BART's performance improving relative to GP's performance as noise increases, which suggests that the fitted stationary GP models struggle with ZDT3's irregular oscillations in its image. These observations imply that BART performs possibly worse than idealized GP (which presumably performs better than a fitted GP when noise is not known) in "simpler" scenarios but better adapts to "complex" behaviors in the underlying data-generating function. We conclude that when the underlying function and noise level are not known, BART may be a safer bet than the GP.

To address quantity **Q.2**, we refer to Figures 10b, 11b, and 12b in the Supplementary Material. For each test function, the depth approach tends to produce similar overcoverage and undercoverage as the random sets approach. Overcoverage tends to be larger than undercoverage for each function and each approach, which suggests α_{MBD} and α_{RS} could be lowered to produce point clouds with less overcoverage and minimally more undercoverage.

To address quantity **Q.3**, we now compare the depth approach to the random sets approach. We first look at BART's PF point clouds. For the MOP2 function, the depth approach tends to produce more overcoverage and less undercoverage than the random sets approach. For the ZDT3 function, the depth approach tends to produce less overcoverage and undercoverage than the random sets approach. For the DTLZ2 function, the depth approach tends to produce less overcoverage and undercoverage than the random sets approach when the observations are not noisy, but more overcoverage when noise is present.

In all of these BART PF observations, the depth approach either outperforms or produces an overcoverage/undercoverage tradeoff with the random sets approach. That is, in no BART PF scenario does the random sets approach outperform the depth approach, which suggests the depth approach produces either as good or better PF point clouds than does the random sets approach. For GP, the depth approach tends to produce less overcoverage and less undercoverage than the random sets approach for all three test functions, which suggests the depth approach produces overall better GP-based PF point clouds than does the random sets approach. We conclude from these observations that the depth approach should be used over the random sets approach if using either BART or GP.

6 Industry 4.0 engineering application

Consider the single cut turning cost operation from Trautmann and Mehnen (2009) (TM), who consider the MO problem of simultaneously minimizing the machining and tool costs, $C_m\colon \mathcal{X} \to \mathbb{R}$ and $C_t\colon \mathcal{X} \to \mathbb{R}$, respectively, for an industrial engineering application where

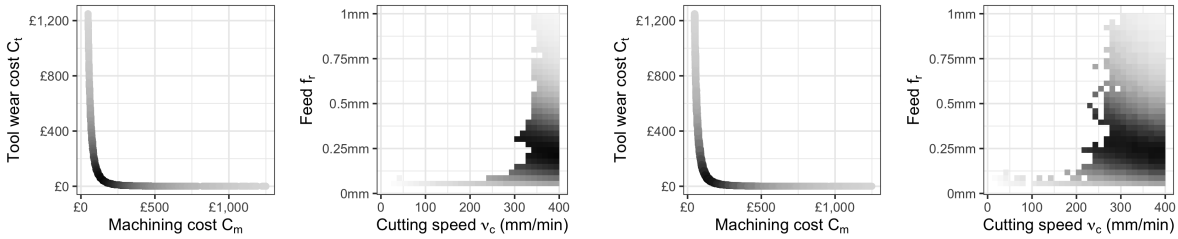
$$C_m(\nu_c, f_r) = b_1 \nu_c^{-1} f_r^{-1} \quad \text{and} \quad C_t(\nu_c, f_r) = b_2 \nu_c^2 f_r^3 \tag{6}$$

with constants $b_1 \approx 12,354\,£\ \text{mm}^2\ \text{min}^{-1}$ and $b_2 \approx 0.0284\,£\ \text{mm}^{-5}\ \text{min}^2$. Each cost has two input variables: cutting speed ν_c with typical values between 10 and 400 mm/min, and the feed f_r with typical values between 0.04 and 1 mm. Thus, we use the input space $\mathcal{X} = [10, 400] \times [0.04, 1]$.

As shown in the left and middle plots of Figure 8, no image point is far from the PF.

Figure 8: Left: image (gray) and PF (black) of the single cut turning cost operation biobjective function. Middle: the left plot zoomed into the lower-left region. Right: input space (gray) and PS (black).

Thus, it is "easy" to estimate the PF. However, this same property in the output space makes PS estimation very difficult, as any input will map to an image point close to or on the PF. Because many input points are far from the PS (shown in the right plot of Figure 8), PS estimates will tend to have large uncertainty.



(a) $n = 15300$ PF UQ. (b) $n = 15300$ PS UQ. (c) $n = 1500$ PF UQ. (d) $n = 1500$ PS UQ.

Figure 9: 50% UQ point clouds for the PF and PS.

TM perform PF estimation but not UQ or PS estimation, which hides the aforementioned large uncertainty in PS estimates. In contrast, we apply our BART-based methodology (with 2,000 posterior samples) to perform PF/PS estimation and UQ on two training data sets generated from Equation 2, where $\sigma^2 = 0$ and Equation 6 takes the role of $\mathbf{f}(\cdot)$. Due to sharp peaks in C_m and C_t in the generated data, we fit our two BART models to $\log C_m$ and $\log C_t$. We then transform the BART predictions back to C_m and C_t before

performing PF/PS estimation and UQ. For UQ, we use only the depth approach per our simulation study conclusions in Section 5.2. Figure 9 shows the 50% PF and PS point clouds for two training sizes: $n = 15300$ to match the number of function evaluations made by TM, and $n = 1500$ to consider the case of an expensive simulator. The dark PF points correspond to the dark PS regions and indicate relatively low C_m and C_t values. As explained in the previous paragraph, both PS point clouds (Figs. 9b and 9d) show large uncertainty. On the other hand, the two PF point clouds (Figs. 9a and 9c) indicate small uncertainty and differ only slightly from each other, which implies only a minor loss in PF inference even with a training size reduction of 90%. Hence, a practitioner can pick an input setting to achieve relatively low C_m and C_t values even with $n = 1500$.

7 Summary and Discussion

Using the fact that BART produces a set of hyperrectangles which partition the input domain (Theorem 1), this paper describes the details of using BART for performing multiobjective optimization, provides an algorithm to find the PF and PS of the multiple outputs and inputs, and compares two different approaches of UQ for the PF and PS. A "random-sets" approach and a newly proposed "depth" approach are used to quantify the uncertainty of BART-generated and GP-generated PF and PS estimates. The depth approach performs similarly or better than the random sets approach (while being computationally advantageous). When the underlying function and noise level is unknown, UQ based on BART-based MO optimization may be superior to GP-based MO optimization. We also note that BART can readily handle categorical inputs, which are often a challenge

in GP models. Finally, we demonstrated our BART-based PF and PS estimation to data generated from an engineering application.

This paper suggests several topics for additional research. First, our UQ comparisons used $\alpha_{MBD} = 0.50$ and $\alpha_{RS} = 0.25$, but we could lower these values to decrease the expected overcoverage and increase the expected undercoverage (or raise these α values to increase expected overcoverage and decrease expected undercoverage). Indeed, the empirical results in this paper suggest using lower values than our α choices but it is unknown what α_{MBD} and α_{RS} values will produce desirable overcoverage/undercoverage values in general.

Second, all test functions in this paper have $d = 2$ objectives in which the depth approach has a speed advantage over the random-sets approach. The random-sets approach, however, can be readily applied to a $d > 2$ scenario. It would be desirable to develop a computationally-efficient depth algorithm for $d > 2$ scenarios and compare the undercoverage and overcoverage to those of the random-sets approach.

Third, this paper used one-stage maximin LHDs. Presumably, BART's PF and PS estimates could be improved using alternative input designs. For example, Chipman et al. (2012) performed sequential design using single-output BART prediction, but additional design research is required for multiple-output BART.

The implementation for finding the PF and PS of a two-dimensional BART model can be found in the Open Bayesian Trees (OpenBT) project at `https://bitbucket.org/mpratola/openbt/` (Pratola, 2020).

ACKNOWLEDGEMENTS

AH would like to acknowledge the Graduate School at The Ohio State University for

support during the dissertation year. The work of MTP was supported in part by the National Science Foundation under Agreement DMS-1916231 and in part by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-2018-CRG7-3800.3. TJS was supported in part by the National Science Foundation under Agreements DMS-0806134 and DMS-1310294 (The Ohio State University).

SUPPLEMENTARY MATERIAL

Proof of Theorem 1: Proof of Theorem 1. (pdf file)

Algorithm to find the PF of a set of vectors: Pseudocode for Kung et al. (1975)'s algorithm to find the PF of a set of $d-$dimensional vectors as described in Section 3.2. (pdf file)

Algorithm to compute modified band depth: Description of the algorithm to compute modified band depth as described in Section 4.2.1. (pdf file)

Test function definitions: Definitions of the test functions introduced in Section 5. (pdf file)

Simulation study full results: PS plots and $n = 32p$ scenario results as described in Section 5. (pdf file)

R code for Section 6: R code for Section 6. (zip file)

APPENDIX A: PROOFS OF THEOREMS

Proof of Theorem 1. First we rewrite each single-output function $\mathcal{E}(\cdot;\theta_j)$ in Equation 5 as a linear combination of indicator functions of hyperrectangles that partition the input space $\mathcal{X}$. To do so, we pick an arbitrary $\mathcal{E}(\cdot;\theta_j)$ and, for notational ease, suppress the index j so that $\mathcal{E}(\cdot;\theta_j) = \mathcal{E}(\cdot;\theta)$. Then $\mathcal{E}(\cdot;\theta)$ can be expressed as the sum in Equation 4. For any tree t in the ensemble, the hyperrectangle set $\{\mathbf{R}_{t,k} : k = 1,\ldots,|\mathcal{M}_t|\}$ partitions the input space $\mathcal{X}$. If we take any one hyperrectangle from each tree, the intersection of these m hyperrectangles will also be a hyperrectangle, albeit possibly empty. Thus, the set of all such intersections, i.e.

$$\Big\{\mathbf{R}_{1,k_1} \cap \mathbf{R}_{2,k_2} \cap \cdots \cap \mathbf{R}_{m,k_m} : k_1 \in \{1,\ldots,|\mathcal{M}_1|\},\ldots,k_m \in \{1,\ldots,|\mathcal{M}_m|\}\Big\},$$

also partitions the input space. (Equation 5 of Pratola (2016) describes how to programatically implement this intersection.) For the single-output ensemble $\mathcal{E}$, let $B_{\mathcal{E}}$ index the set of the nonempty m-wise intersections. Then we may rewrite the function $\mathcal{E}(\cdot;\theta)$ as a linear combination of indicators[1] of these nonempty intersections, i.e.

$$\mathcal{E}(\cdot;\theta) = \sum_{l \in B_{\mathcal{E}}} \alpha_l \mathbb{I}_{\mathbf{R}_l}(\cdot),$$

where each α_l is the sum of the m terminal node parameter values that correspond to the original hyperrectangles that created $\mathbf{R}_l$, i.e. if $\mathbf{R}_l = \mathbf{R}_{1,k_1} \cap \mathbf{R}_{2,k_2} \cap \cdots \cap \mathbf{R}_{m,k_m}$ then $\alpha_l = \mu_{1,k_1} + \mu_{2,k_2} + \cdots + \mu_{m,k_m}$.

[1]The indicator function $\mathbb{I}_{\mathbf{R}}(\cdot) : \mathcal{X} \to \{0,1\}$ of a hyperrectangle $\mathbf{R} \subset \mathcal{X}$ is defined as follows: for an input $\mathbf{x} \in \mathcal{X}$, the value $\mathbb{I}_{\mathbf{R}}(\mathbf{x})$ equals one if $\mathbf{x} \in \mathbf{R}$ but equals zero if $\mathbf{x} \notin \mathbf{R}$.

Now we rewrite the $d-$objective ensemble function $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$, for which we bring back the previously suppressed index notation. For each ensemble $\mathcal{E}_i$, the hyperrectangle set $\{\mathbf{R}_{l_i} : l_i \in B_{\mathcal{E}_i}\}$ partitions the input space $\mathcal{X}$. Similar to the previous paragraph, the set of all such $d-$wise intersections, i.e.

$$\Big\{\mathbf{R}_{l_1} \cap \mathbf{R}_{l_2} \cap \cdots \cap \mathbf{R}_{l_d} : l_1 \in B_{\mathcal{E}_1}, l_2 \in B_{\mathcal{E}_2}, \ldots, l_d \in B_{\mathcal{E}_d}\Big\},$$

also partitions the input space, where each hyperrectangle $\mathbf{R}_{l_i}$ has associated with it the value α_{l_i}. Hence, each $d-$wise intersection $\mathbf{R}_{l_1} \cap \mathbf{R}_{l_2} \cap \cdots \cap \mathbf{R}_{l_d}$ has associated with it the $d-$tuple $(\alpha_{l_1}, \alpha_{l_2}, \ldots, \alpha_{l_d}) \in \mathbb{R}^d$. For the multiple-output ensemble $\boldsymbol{\mathcal{E}}$, let $B_{\boldsymbol{\mathcal{E}}}$ index the set of the nonempty $d-$wise intersections. Then we may express the function $\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta})$ as a linear combination of indicators of these nonempty intersections, i.e.

$$\boldsymbol{\mathcal{E}}(\cdot;\boldsymbol{\theta}) = \sum_{q \in B_{\boldsymbol{\mathcal{E}}}} \boldsymbol{\alpha}_q \mathbb{I}_{\mathbf{R}_q}(\cdot),$$

where each $\boldsymbol{\alpha}_q = (\alpha_{l_1}, \ldots, \alpha_{l_d}) \in \mathbb{R}^d$ when $\mathbf{R}_q = \mathbf{R}_{l_1} \cap \mathbf{R}_{l_2} \cap \cdots \cap \mathbf{R}_{l_d}$.

□

APPENDIX B: ALGORITHM PSEUDOCODE

B.1 Algorithm to find PF of a set of vectors.

Algorithm 1 Find the PF of a set of d-dimensional vectors.

Input: Set $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_{|V|}\}$ of d-dimensional vectors arranged in increasing order of their first coordinates. In case of ties, arrange elements in increasing order of their second coordinates. Repeat for further coordinates as necessary.

Output: PF of V.

```
PF ← FindPF(V)

function FindPF(V)

    if |V| == 1 then
        return V
    end if

    Set R ← FindPF({v_1, ..., v_⌊|V|/2⌋})  // recursive call
    Set S ← FindPF({v_⌊|V|/2⌋+1, ..., v_|V|})  // recursive call

    // Create T to be the set of vectors in S not dominated by any vectors in R.
    Set T ← ∅.
    for v_i ∈ S do
        if v_i ⊁ R then
            Set T ← T ∪ {v_i}
        end if
    end for

    return R ∪ T

end function
```

B.2 Algorithm to compute modified band depth of CPFs.

Input: CPFs $c^{(1)}, \ldots, c^{(N)}$, where $d = 2$.

Output: Modified band depth of each CPF.

1. Create cuts. (Time: $\mathcal{O}(N)$)

   (a) Define endpoints for each axis. Let y_1^{min} (y_1^{max}) be the minimum (maximum) y_1 value of the points in any of the N CPFs. Then for the y_1 axis, we will consider only the interval $I_1 := [y_1^{min}, y_1^{max}]$. Similarly define y_2^{min} and y_2^{max}. Then for the y_2 axis, we will consider only the interval $I_2 := [y_2^{min}, y_2^{max}]$.

   (b) Create q values $t_1, t_2, \ldots, t_q$ equally spaced in interval I_1, where $t_1 = y_1^{min}$ and $t_q = y_1^{max}$. Similarly, create q values $s_1, s_2, \ldots, s_q$ equally spaced in interval I_2, where $s_1 = y_2^{min}$ and $s_q = y_2^{max}$.

2. Create $(dq) \times N$ matrix M by creating two $q \times N$ matrices M^1 and M^2, where the ijth element of M^1 is defined to be $h_1(t_i; c^{(j)}) := \min\{y_2 \in I_2 : (t_i, y_2) \in A^{(j)}\}$ and the ijth element of M^2 is defined to be $h_2(s_i; c^{(j)}) := \min\{y_1 \in I_1 : (y_1, s_i) \in A^{(j)}\}$. Matrix M is the vertical stacking of matrices M^1 and M^2. (Time: $\mathcal{O}(dqN)$)

   (a) Put ∞ for the entries in the matrix M corresponding to nonexistent values of $h_1(y_1; c^{(j)})$ or $h_2(y_2; c^{(j)})$.

3. Create $(dq) \times N$ matrix $R^{\max}$ ($R^{\min}$) whose ith row is defined to be the ascending max (min) rank of the ith row of M, where ascending rank means the smallest value gets rank 1 while the largest value gets rank N. (Time: $\mathcal{O}(dqN \log N)$)

4. Create $(dq) \times N$ matrix T whose ijth element is $T_{ij} = [R_{ij}^{\max}(N - R_{ij}^{\min} + 1) - (R_{ij}^{\max} - R_{ij}^{\min} + 1)^2]/\binom{N}{2}$, where $R_{ij}^{\max}$ ($R_{ij}^{\min}$) is the ijth element of the matrix $R^{\max}$ ($R^{\min}$). (Time: $\mathcal{O}(dqN)$)

5. Create $N-$tuple whose jth element, defined to be the mean of the elements in matrix T's jth column (i.e. $(2q)^{-1}\sum_{i=1}^{2q} T_{ij}$), is the depth of CPF $c^{(j)}$ as defined in Example 4 for CPF $c^{(7)}$. (Time: $\mathcal{O}(dqN)$)

APPENDIX C: TEST FUNCTION DEFINITIONS

1. MOP2 from Fonseca and Fleming (1995) has $p = 2$ inputs and $d = 2$ objectives:

$$f_1(\mathbf{x}) = 1 - \exp\left\{ -\sum_{i=1}^{2} \left(4x_i - 2 - \frac{1}{\sqrt{2}}\right)^2 \right\}$$
$$f_2(\mathbf{x}) = 1 - \exp\left\{ -\sum_{i=1}^{2} \left(4x_i - 2 + \frac{1}{\sqrt{2}}\right)^2 \right\}.$$

   The PS is the line segment $\mathcal{P}_{\mathcal{X}} = \{(t,t) : (2 - 1/\sqrt{2})/4 \leq t \leq (2 + 1/\sqrt{2})/4\}$ and the PF is the concave curve segment shown in Figure 6. Also, $\mathtt{Var}_{\mathbf{X}}(f_1(\mathbf{X})) = \mathtt{Var}_{\mathbf{X}}(f_2(\mathbf{X})) \approx 0.0636$.

2. ZDT3 from Zitzler et al. (2000) has a disconnected PF and PS. For $p = 2$ inputs, the $d = 2$ objectives are

$$f_1(\mathbf{x}) = x_1$$
$$f_2(\mathbf{x}) = \left[(1 - \sqrt{x_1/g}) - \frac{x_1}{g}\sin(10\pi x_1) + c_3\right]/(10 + c_3),$$

   where $c_3 := 0.70238$ and $g := 1 + \frac{9}{p-1}\sum_{i=2}^{p} x_i = 1 + 9x_2$. Figure 6 show the PF and PS. Also, $\mathtt{Var}_{\mathbf{X}}(f_1(\mathbf{X})) = \frac{1}{12} \approx 0.0833$ and $\mathtt{Var}_{\mathbf{X}}(f_2(\mathbf{X})) \approx 0.0461$.

3. We modify DTLZ2 from Deb et al. (2005) for $p = 4$ inputs and $d = 2$ objectives so

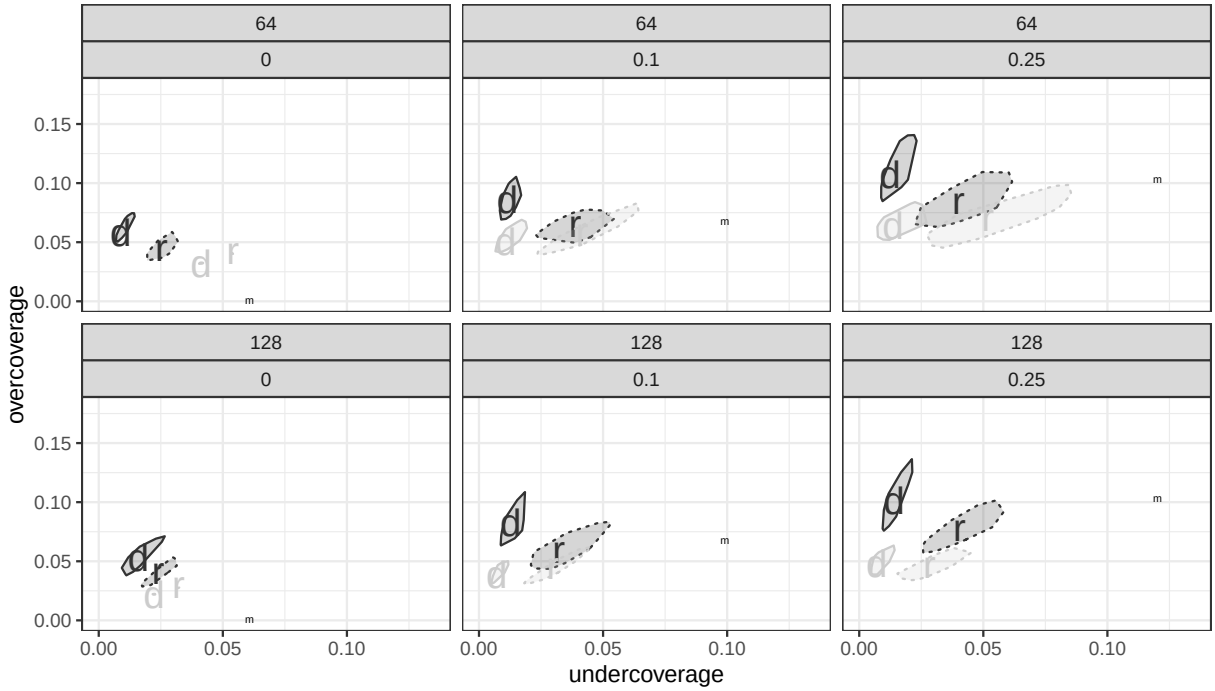
that the resulting PF is convex rather than concave. Our modified objectives are

$$f_1(\mathbf{x}) = (g(\mathbf{x}_{2:4}) - 1)\cos\Big(\frac{\pi x_1}{2}\Big) + 1$$
$$f_2(\mathbf{x}) = (g(\mathbf{x}_{2:4}) - 1)\sin\Big(\frac{\pi x_1}{2}\Big) + 1,$$

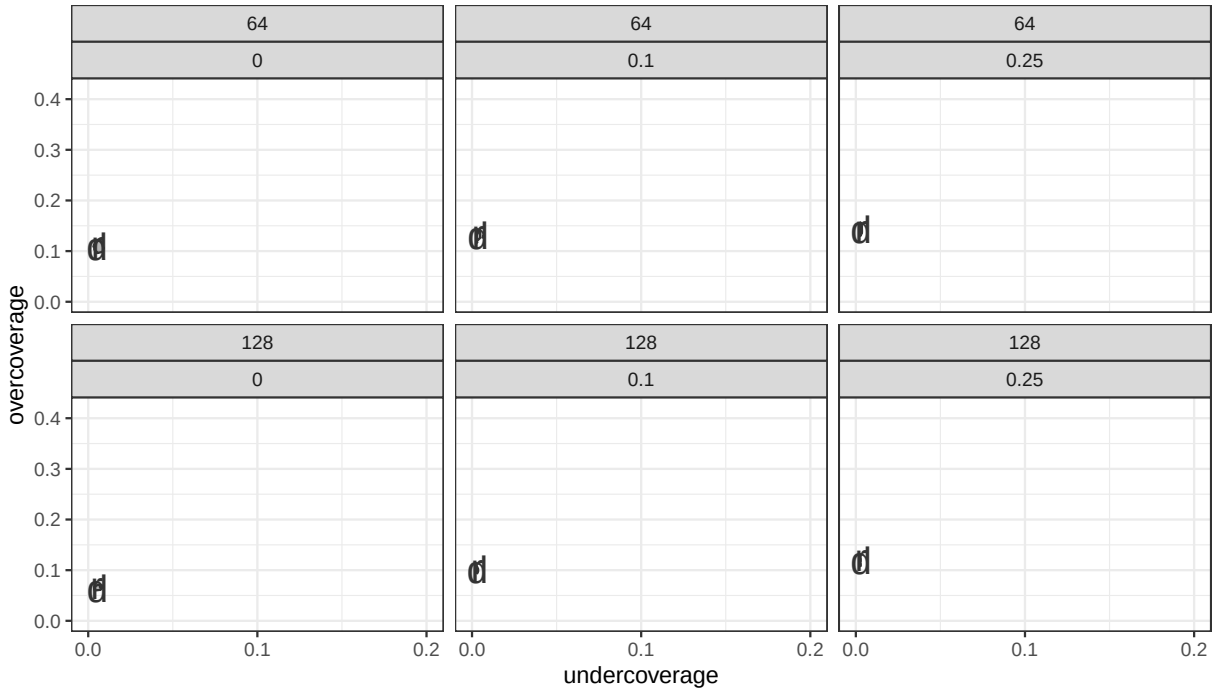
where $g(\mathbf{x}_{2:4}) := \sum_{i=2}^{4}(x_i - 0.5)^2$. The PS is the line segment $\mathcal{P}_{\mathcal{X}} = \{(x_1, 0.5, 0.5, 0.5) : x_1 \in [0, 1]\}$ and the PF is the convex quarter-circle $\mathcal{P}_{\mathcal{F}} = \{(1 - \cos t, 1 - \sin t) : t \in [0, \pi/2]\}$. Also, $\mathtt{Var}_{\mathbf{X}}(f_1(\mathbf{X})) = \mathtt{Var}_{\mathbf{X}}(f_2(\mathbf{X})) \approx 0.0616$.

APPENDIX D: SIMULATION STUDY FULL RESULTS

See Figures 10, 11, and 12.

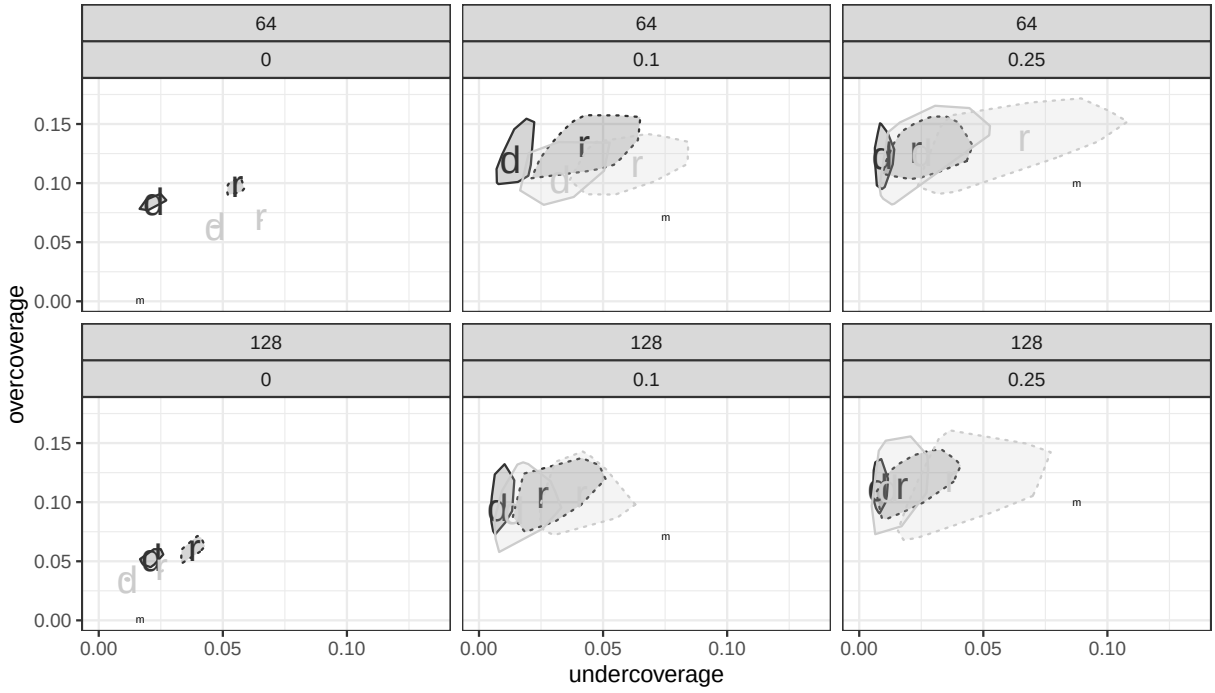


(a) MOP2 PF data.

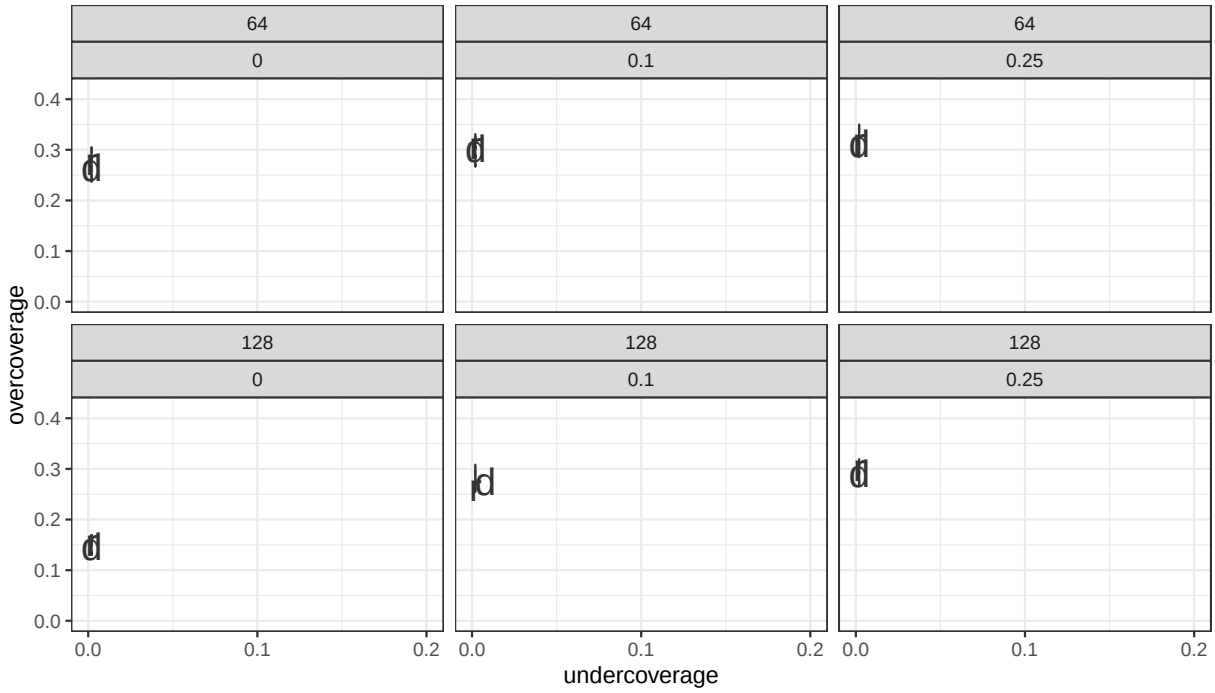


(b) MOP2 PS data.

Figure 10: Bagplots of the 100 values of $d(\widehat{\mathcal{P}_{\mathcal{F}}}, \mathcal{P}_{\mathcal{F}})$ and $d(\mathcal{P}_{\mathcal{F}}, \widehat{\mathcal{P}_{\mathcal{F}}})$. Each plot represents a simulation scenario where $n \in \{64, 128\}$, and the variance multiplier is a value in $\{0, 0.1, 0.25\}$. Bags with solid (dashed) outline and median labeled 'd' ('r') display depth (random sets) approach. Black (gray) bags display BART (GP) model.

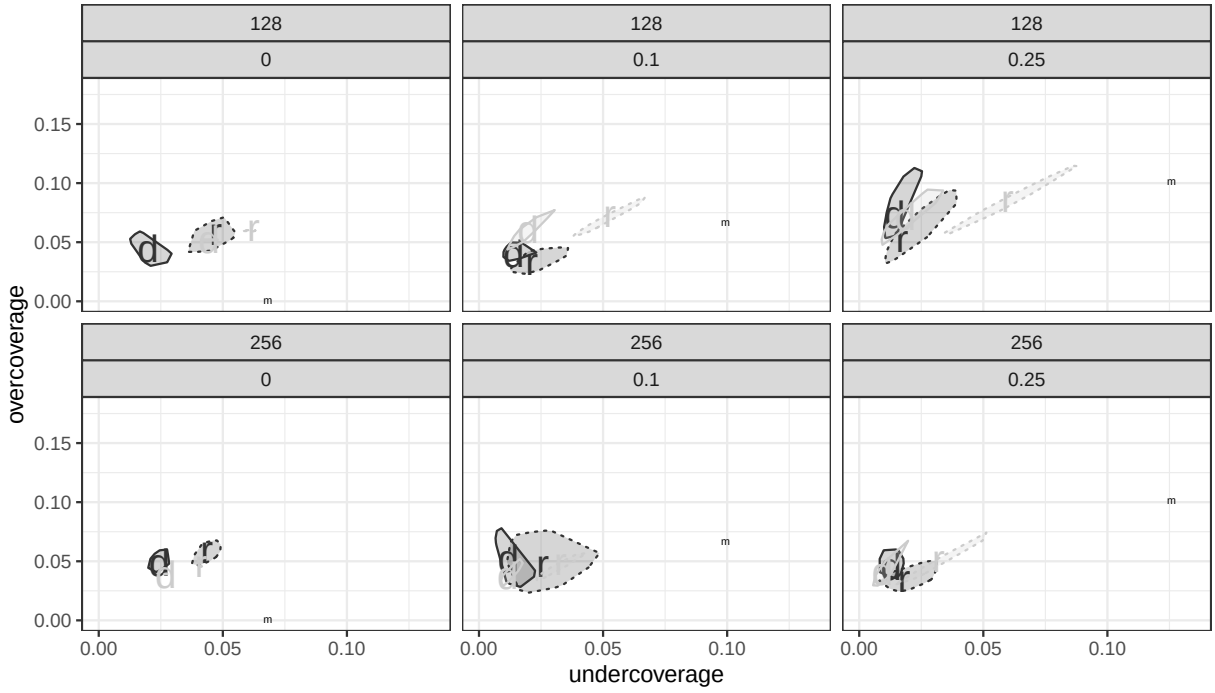


(a) ZDT3 PF data.

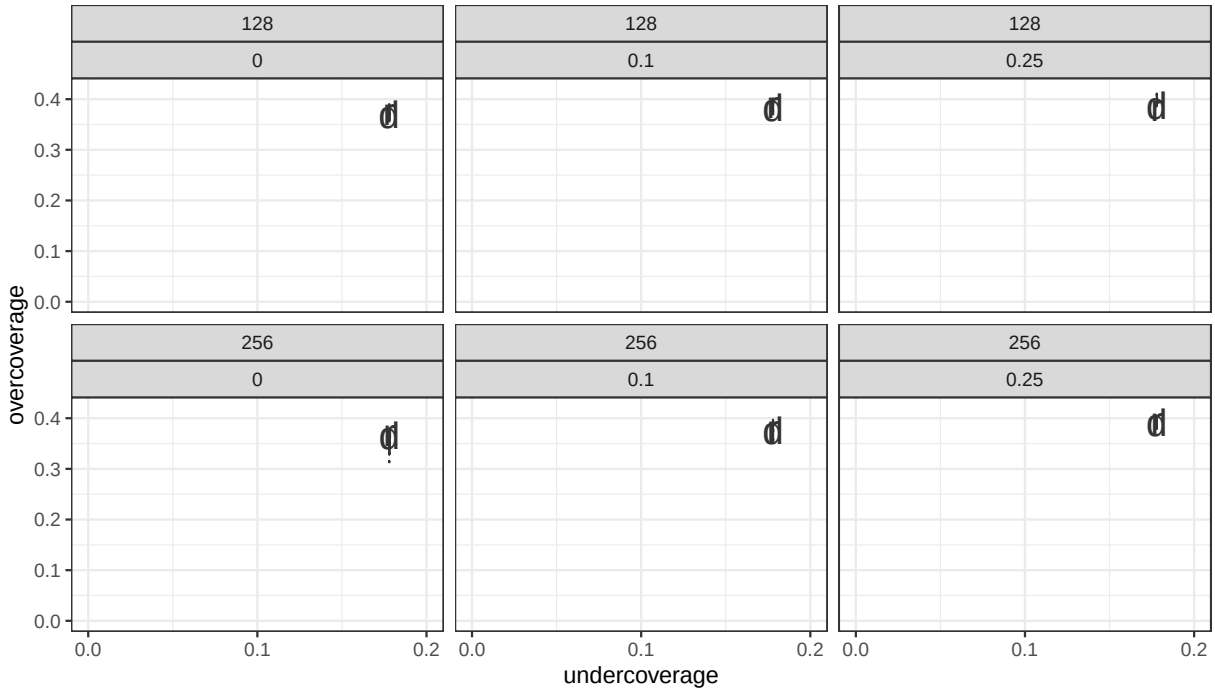


(b) ZDT3 PS data.

Figure 11: Bagplots of the 100 values of $d(\widehat{\mathcal{P}_{\mathcal{X}}}, \mathcal{P}_{\mathcal{X}})$ and $d(\mathcal{P}_{\mathcal{X}}, \widehat{\mathcal{P}_{\mathcal{X}}})$. Each plot represents a simulation scenario where $n \in \{64, 128\}$, and the variance multiplier is a value in $\{0, 0.1, 0.25\}$. Bags with solid (dashed) outline and median labeled 'd' ('r') display depth (random sets) approach. Black (gray) bags display BART (GP) model.



(a) DTLZ2 PF data.



(b) DTLZ2 PS data.

Figure 12: Bagplots of the 100 values of $d(\widehat{\mathcal{P}_{\mathcal{X}}}, \mathcal{P}_{\mathcal{X}})$ and $d(\mathcal{P}_{\mathcal{X}}, \widehat{\mathcal{P}_{\mathcal{X}}})$. Each plot represents a simulation scenario where $n \in \{64, 128\}$, and the variance multiplier is a value in $\{0, 0.1, 0.25\}$. Bags with solid (dashed) outline and median labeled 'd' ('r') display depth (random sets) approach. Black (gray) bags display BART (GP) model.

References

Binois, M., D. Ginsbourger, and O. Roustant (2015). Quantifying uncertainty on pareto fronts with gaussian process conditional simulations. European Journal of Operational Research 243(2), 386–394.

Chipman, H., P. Ranjan, and W. Wang (2012). Sequential design for computer experiments with a flexible bayesian additive model. Canadian Journal of Statistics 40(4), 663–678.

Chipman, H. A., E. I. George, and R. E. McCulloch (1998). Bayesian cart model search. Journal of the American Statistical Association 93, 935–948.

Chipman, H. A., E. I. George, and R. E. McCulloch (2010). Bart: Bayesian additive regression trees. The Annals of Applied Statistics 4, 266–298.

da Fonseca, V. G. and C. M. Fonseca (2010). The attainment-function approach to stochastic multiobjective optimizer assessment and comparison. In Experimental methods for the analysis of optimization algorithms, pp. 103–130. Springer.

Dubuisson, M.-P. and A. K. Jain (1994). A modified hausdorff distance for object matching. In Proceedings of 12th international conference on pattern recognition, Volume 1, pp. 566–568. IEEE.

Edwin van Dam, Dick den Hertog, Bart Husslage, and Gijs Rennen (2015). Space-filling designs. Accessed: 2020-07-01.

Emmerich, M. T., A. H. Deutz, and J. W. Klinkenberg (2011). Hypervolume-based ex-

pected improvement: Monotonicity properties and exact computation. In 2011 IEEE Congress of Evolutionary Computation (CEC), pp. 2147–2154. IEEE.

Fu, Y., J. Ding, H. Wang, and J. Wang (2018). Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in industry 4.0-based manufacturing system. Applied Soft Computing 68, 847–855.

Han, Y., D. Gong, Y. Jin, and Q.-k. Pan (2016). Evolutionary multi-objective blocking lot-streaming flow shop scheduling with interval processing time. Applied Soft Computing 42, 229–245.

Ivanov, D., A. Dolgui, B. Sokolov, F. Werner, and M. Ivanova (2016). A dynamic model and an algorithm for short-term supply chain scheduling in the smart factory industry 4.0. International Journal of Production Research 54(2), 386–402.

Knowles, J. (2006). Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 10(1), 50–66.

Kung, H.-T., F. Luccio, and F. P. Preparata (1975). On finding the maxima of a set of vectors. Journal of the ACM (JACM) 22(4), 469–476.

López-Pintado, S. and J. Romo (2009). On the concept of depth for functional data. Journal of the American Statistical Association 104(486), 718–734.

Molchanov, I. S. (2005). Theory of random sets, Volume 19. Springer.

Pratola, M. T. (2020). Open Bayesian trees. `https://bitbucket.org/mpratola/openbt/src/master/`. Accessed: 2020-10-09.

Pratola, M. T., H. A. Chipman, J. R. Gattiker, D. M. Higdon, R. McCulloch, and W. N. Rust (2014). Parallel bayesian additive regression trees. Journal of Computational and Graphical Statistics 23, 830–852.

Rousseeuw, P. J., I. Ruts, and J. W. Tukey (1999). The bagplot: a bivariate boxplot. The American Statistician 53(4), 382–387.

Roustant, O., D. Ginsbourger, and Y. Deville (2012). DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. Journal of Statistical Software 51(1), 1–55.

Shukla, A. K., R. Nath, P. K. Muhuri, and Q. D. Lohani (2020). Energy efficient multi-objective scheduling of tasks with interval type-2 fuzzy timing constraints in an industry 4.0 ecosystem. Engineering Applications of Artificial Intelligence 87, 103257.

Sun, Y., M. G. Genton, and D. W. Nychka (2012). Exact fast computation of band depth for large functional datasets: How quickly can one million curves be ranked? Stat 1(1), 68–74.

Svenson, J. (2011). Computer experiments: Multiobjective optimization and sensitivity analysis. Ph. D. thesis, The Ohio State University.

Trautmann, H. and J. Mehnen (2009). Preference-based pareto optimization in certain and noisy environments. Engineering Optimization 41(1), 23–38.

Tukey, J. W. (1975). Mathematics and the picturing of data. In Proceedings of the International Congress of Mathematicians, Vancouver, 1975, Volume 2, pp. 523–531.

Whitaker, R. T., M. Mirzargar, and R. M. Kirby (2013, Dec). Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. IEEE Transactions on Visualization and Computer Graphics 19(12), 2713–2722.

Xie, Z., C. Zhang, X. Shao, W. Lin, and H. Zhu (2014). An effective hybrid teaching–learning-based optimization algorithm for permutation flow shop scheduling problem. Advances in Engineering Software 77, 35–47.

Zhang, Q., W. Liu, E. Tsang, and B. Virginas (2009). Expensive multiobjective optimization by moea/d with gaussian process model. IEEE Transactions on Evolutionary Computation 14(3), 456–474.