


Using BART to Perform Pareto Optimization and Quantify its Uncertainties

Akira Horiguchi ^a, Thomas J. Santner^a, Ying Sun ^b, and Matthew T. Pratola^a

^a Department of Statistics, The Ohio State University, Columbus, OH ^b Statistics Program, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

ABSTRACT

Techniques to reduce the energy burden of an industrial 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 nonparametric model whose assumptions are typically less restrictive than popular alternatives, such as Gaussian Processes (GPs). These less restrictive assumptions allow BART to handle scenarios (e.g., high-dimensional input spaces, nonsmooth responses, large datasets) that GPs find difficult. 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 engineering problem. [Supplementary materials](#), which include a theorem proof, algorithms, and R code, for this article are available online.

ARTICLE HISTORY


Received 31 December 2020

Revised 3 September 2021

Accepted 6 September 2021

KEYWORDS

Band depth, Bayesian methods, Computer experiments, Pareto Set, Random Sets

CONTACT Akira Horiguchi akira.horiguchi@duke.edu  Department of Statistical Science, Duke University, Durham, NC 27708-0187.

1 Introduction

Many important Industry 4.0 problems can be formulated as multiobjective optimization (MO) problems. For example, Shukla et al. (2020) described how the use of dynamic voltage scaling in real-time embedded systems (RTES) produces the two mutually conflicting objectives of energy efficiency and timeliness of task execution. Furthermore, the timing constraints of tasks in RTES can only be approximated, which prompts the need to quantify this imprecision.

More generally, if each objective in a problem corresponds to an output dimension in the vector-valued function $\mathbf{f}(\cdot)$, the goal of MO is to “minimize” (or “maximize,” depending on the application) this function. Seldom will all of these objectives be simultaneously minimized by the same input setting. Hence, we seek to find the set of best compromises between competing objectives and the set of all inputs that produce these compromises. (Section 3 provides a mathematical definition of best compromises.) The former set is called the Pareto Front (PF); the latter set is called the Pareto Set (PS).

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. Binois, Ginsbourger, and Roustant (2015) achieved PF estimation and uncertainty quantification (UQ) by fitting a Gaussian Process (GP), simulating approximate realizations of the fitted GP, and treating the resulting approximate conditional PFs as random sets. However, their

second step requires discretizing the input space into a finite number of points, which may be computationally expensive if the input dimension is large. Furthermore, no work in general has been done to quantify the uncertainty of PS estimation with GPs.

A popular alternative to the GP for emulating single-output simulators is the Bayesian Additive Regression Trees (BART) model introduced by Chipman, George, and McCulloch (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, easily handles categorical inputs, 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 input dimensions and large datasets (Pratola et al. 2014). Breiman (2001)’s Random-Forest model is also used in surrogate-based optimization and retains many of BART’s advantages over GP, but lacks BART’s natural UQ capabilities via its Bayesian formulation.

Though BART has been used for single-objective optimization (e.g., in Chipman, Ranjan, and Wang 2012), it has never been used to perform multiobjective optimization. Our primary contribution is twofold. We first find the PF and PS of exact simulated realizations of a fitted multiple-output BART model and hence avoid grid approximations of the input space. We then quantify the uncertainty of these estimates of the PF and PS of $\mathbf{f}(\cdot)$ using random sets (Binois, Ginsbourger, and Roustant 2015) and our novel extension of the depth approach described in López-Pintado and Romo (2009), Sun, Genton, and Nychka (2012), and Whitaker, Mirzargar, and Kirby (2013).

The article is organized as follows. Section 2 introduces BART with multiple outputs. Given a multiple-output BART function, Section 3 establishes how to find its image, PF, and PS. 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. Section 6 demonstrates our BART-based methodology on an engineering application. Section 7 concludes the article with a discussion. Proofs of stated theorems can be found in the [supplementary materials](#).

2 BART

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

$$\mathbf{Y}(\mathbf{x}_i) = \mathbf{f}(\mathbf{x}_i) + \boldsymbol{\epsilon}_i, \quad \mathbf{f}(\cdot) = (f_1(\cdot), f_2(\cdot), \dots, f_d(\cdot)) : \mathcal{X} \rightarrow \mathbb{R}^d \quad (1)$$



where $\mathbf{f}(\cdot)$ is the vector-valued function described in Section 1, each $y_j(\cdot)$ and $f_j(\cdot)$ has common domain $\mathcal{X} \subset \mathbb{R}^p$, noise vectors $\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_n \stackrel{\text{iid}}{\sim} N_d(\mathbf{0}, \boldsymbol{\sigma}^2 I_d)$, and parameter $\boldsymbol{\sigma}^2 = (\sigma_1^2, \dots, \sigma_d^2)$. We assume the domain \mathcal{X} is a p -dimensional bounded hyperrectangle.

2.1 Multiple-Output BART

To make inference on the unknown $\mathbf{f}(\cdot)$, we approximate each marginal $f_j(\cdot)$ (for $j = 1, \dots, d$) by fitting a BART model to the marginal dataset $D_j := \{(y_j(\mathbf{x}_i), \mathbf{x}_i)\}_{i=1}^n$. These d independently fitted BART models define our d -output BART model: the i th posterior draw of the d -output BART model is $\boldsymbol{\Theta}^{(i)} = (\Theta_1^{(i)}, \dots, \Theta_d^{(i)})$, where each $\Theta_j^{(i)}$ is the i th posterior draw of the j th fitted BART model.

(Horiguchi, Pratola, and Santner 2021, sec. 2) described how a posterior draw $\boldsymbol{\Theta}$ of a BART model induces a regression function $\mathcal{E}(\cdot; \boldsymbol{\theta})$, where

$\theta := \{(\mathcal{T}_t, \mathcal{M}_t)\}_{t=1}^m$. Similarly, each posterior draw Θ of our d -output BART model induces the d -output regression function

$$\mathcal{E}(\cdot; \theta) = (\mathcal{E}(\cdot; \theta_1), \mathcal{E}(\cdot; \theta_2), \dots, \mathcal{E}(\cdot; \theta_d)) : \mathcal{X} \rightarrow \mathbb{R}^d \quad (2)$$

where $\theta = (\theta_1, \dots, \theta_d)$ and each θ_j comes from the j th fitted BART model.

2.2 Prior Specification

Here, we describe our prior specifications for tree-topology parameters \mathcal{T}_t , leaf-node parameters $\mu_{tk} | \mathcal{T}_t$, and noise variance σ^2 because we ultimately deviate from CGM's default hyperparameter values for our multiobjective optimization problem.

The $\pi(\mathcal{T}_t)$ prior decomposes into three components: tree depth, split variable at each internal node, and cutpoint value at each internal node. We leave details of the first two components to CGM and Chipman, George, and McCulloch (1998). For the cutpoint value of any given split variable, this article uses a discrete uniform prior of 30 values over the range of the observed input values.

CGM model $\mu_{tk} | \mathcal{T}_t$ with a Gaussian prior $N(0, (4\kappa^2 m)^{-1})$ (after centering and rescaling the output data so that the minimum and maximum observed transformed response values are, respectively, $y_{\min} = -0.5$ and $y_{\max} = 0.5$). Under the sum-of-trees model, the prior on $\mathbb{E}[Y(\mathbf{x})]$ then becomes $N(0, (4\kappa^2)^{-1})$, where CGM default to $\kappa = 2$. For single-objective optimization, however, Chipman, Ranjan, and Wang (2012) use $\kappa = 1$ to allow BART to produce more pronounced optima. For similar reasons, we 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 leaf node to ten.

For $\pi(\sigma^2)$, we 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 hyperparameter κ more strongly influences the smoothness of the response.

3 Multiobjective Optimization

This section details how we can find the PF and PS of the multiple-output BART regression function (2) given some fixed θ . However, we must first introduce the notion of Pareto dominance, which we use to identify the best compromises between competing objectives.

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

We can now precisely define a multiobjective function's PF and PS: the PF is the set of all nondominated image points; the PS is the set of all inputs that produce the PF. For example, consider [Figure 1\(c\)](#), which shows the image of a biobjective function. Any point on the dashed segment is not dominated by any other image point while any image point on a solid segment is dominated by at least one other image point. Thus, the PF is the dashed segment in [Figure 1\(c\)](#) and the PS is the interval $[0.25, 0.75]$ in [Figures 1\(a\) and 1\(b\)](#).

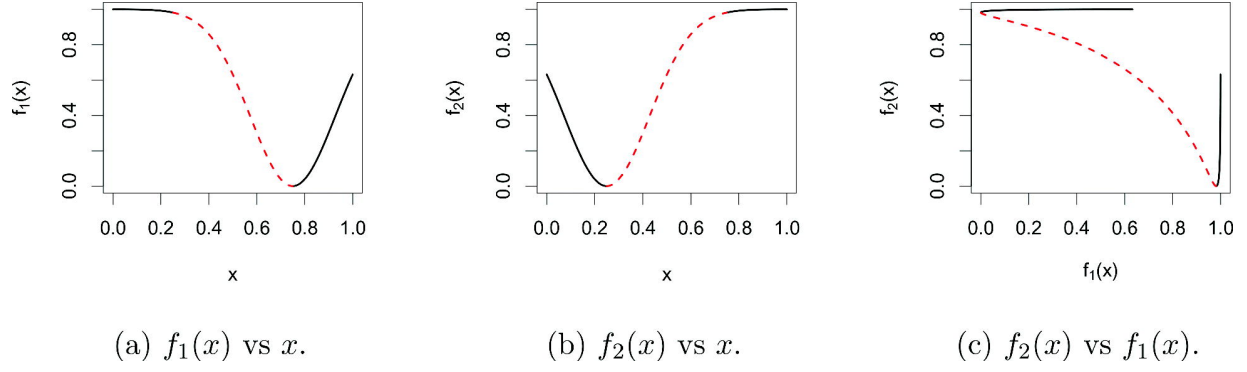


Fig. 1 A biobjective function $\mathbf{f}(x) = (f_1(x), f_2(x))$ where $x \in [0, 1]$.

We find the PF and PS of (2) using the two-step algorithm:

1. (Section 3.1) Find the image $\mathcal{E}(\mathcal{X}; \boldsymbol{\theta})$ and corresponding input hyperrectangles.
2. (Section 3.2) Find all nondominated points in the image $\mathcal{E}(\mathcal{X}; \boldsymbol{\theta})$.

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 Finding the Image of a Multiple-Output BART Function

The following example will make it clear that (2) has a finite image. We can find these image points and their corresponding input hyperrectangles using the parameter values $\boldsymbol{\theta}$.

Example 1. Here, we find the image of the biobjective function $\mathcal{E}(\cdot; \boldsymbol{\theta})$ shown in Figure 2, where $\mathcal{X} = [0, 1]^2$. Any input belongs to one leaf node per tree; these $dm = 4$ leaf nodes correspond to an image point. For example, the input $\mathbf{x}^* = (0.1, 0.07)$ belongs to the four dash-bordered leaf nodes, which correspond to the image point $\mathcal{E}(\mathbf{x}^*; \boldsymbol{\theta}) = (-5, -4)$ shown as a dashed triangle. Conversely, this image point corresponds to the hatch-filled input rectangle, which is the set of all inputs that belong to all four dash-bordered leaf nodes.

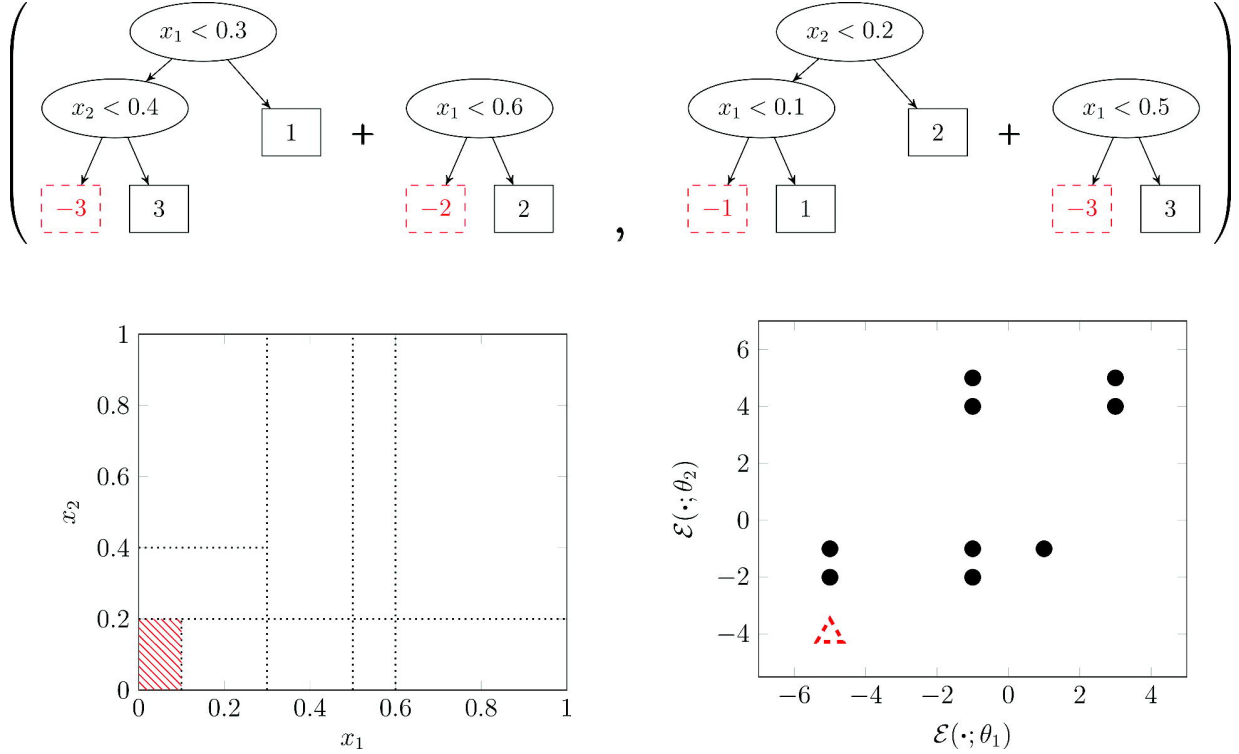


Fig. 2 A biobjective function $\mathcal{E}(\cdot; \theta) = (\mathcal{E}(\cdot; \theta_1), \mathcal{E}(\cdot; \theta_2))$ with tree ensembles (top row) with parameters θ_1 (left) and θ_2 (right). Bottom left: input space partitioned into 10 rectangles. Bottom right: output space with all 10 image points of $\mathcal{E}(\cdot; \theta)$.

Of the $(3 \times 2)(3 \times 2) = 36$ possible one-leaf-node-per-tree combinations, only 10 are valid and produce an image point. For an example of an invalid combination, an input point cannot belong to both the right and left leaf nodes of, respectively, the first and second ensemble's right tree; the conditions $x_1 \geq 0.6$ and $x_1 < 0.5$ cannot be simultaneously satisfied. With only 10 valid combinations, the function $\mathcal{E}(\cdot; \theta)$ has only 10 image points (shown in the bottom right plot of Figure 2).

Between the two ensembles, the six split rules together partition the input space such that the resulting set of partitioning rectangles (shown in the bottom left plot of Figure 2) is bijective to the set of valid one-leaf-node-per-tree combinations, which is itself bijective to the image of $\mathcal{E}(\cdot; \theta)$. Thus, the function $\mathcal{E}(\cdot; \theta)$ can be written as a linear combination of indicator functions each corresponding to a partitioning rectangle.

Theorem 1 d describes a similar result but for any d -output BART function $\mathcal{E}(\cdot; \boldsymbol{\theta})$: if we obtain every possible one-leaf-node-per-tree combination, we can find the image and corresponding input (hyper)rectangles of the d -objective function. The proof of Theorem 1 (see [supplementary material](#)) provides more insight into how these two tasks can be achieved.

Theorem 1. Any d -output BART function in the form of (2) can be written as a linear combination of indicator functions of hyperrectangles:

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

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

3.2 Finding the PF and PS of Multiple-Output BART Function

After finding the image of a d -output BART function $\mathcal{E}(\cdot; \boldsymbol{\theta})$, we find its set of nondominated points using an efficient recursive algorithm from Kung, Luccio, and Preparata (1975). For simplicity, we describe (in the [supplementary material](#)) only one of these algorithms, which finds the nondominated points in a finite set V of d -dimensional vectors. In our setting, the set V is the image of $\mathcal{E}(\cdot; \boldsymbol{\theta})$. Kung, Luccio, and Preparata (1975) showed that the algorithm's time complexity has an upper bound of $O(|V|(\log_2 |V|)^{d-2})$ if $d > 3$. This algorithm is still valid if $d \in \{2, 3\}$, but this upper bound no longer applies in these cases.

The desired PS is the union of all input hyperrectangles corresponding to the nondominated image points. For the example in [Figure 2](#), the PF is the dashed triangle image point while the PS is the hatched input rectangle.

4 Uncertainty Quantification

We can quantify the uncertainty of PF estimates induced by a fitted multiple-output BART model by making N independent draws from the posterior, creating for each draw the resulting BART regression function as defined in (2), and finding each conditional PF (CPF) as described in Section 3. This section details two UQ approaches using the sample of N CPFs, which we denote as $c^{(1)}, \dots, c^{(N)}$. Both approaches use dominated point set closures (DPSCs): for $i = 1, \dots, N$, 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 \mathbf{D} . Figure 3 shows examples of DPSCs.

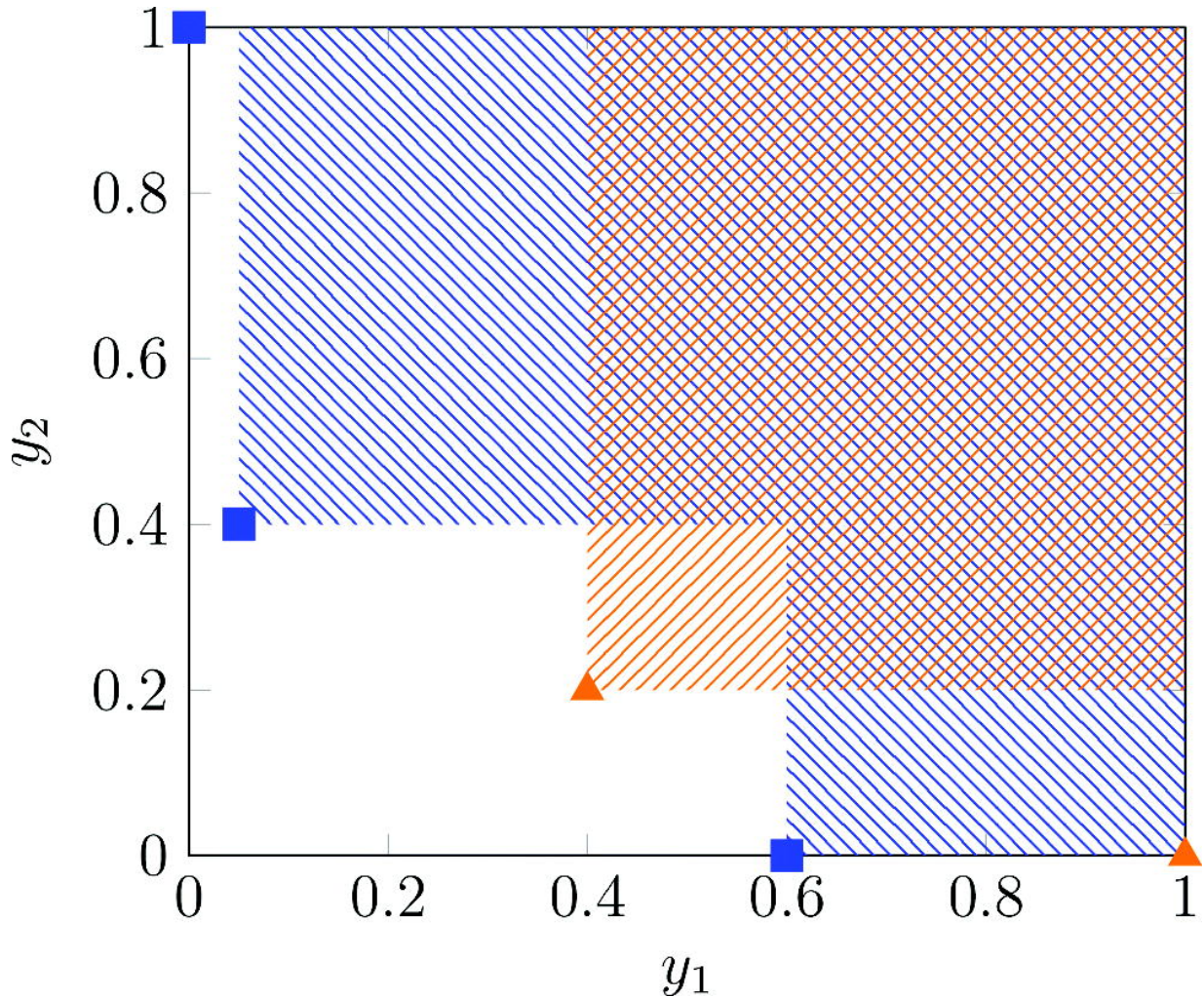


Fig. 3 Example of 2 CPFs (triangles and squares) and the corresponding DPSCs (areas with hatched lines) in objective space $\mathcal{Y} = [0, 1]^2$.

4.1 Random-Sets Approach

da Fonseca and Fonseca (2010) treated each DPSC as a realization of an attained set – a random closed set whose probability distribution is characterized by its attainment function.

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

Example 2. Consider Figure 3 and let A^{\blacksquare} (A^{\blacktriangle}) be the DPSC created from the square (triangle) points. Define an attained set \mathcal{A} to be A^{\blacksquare} with probability 0.1 and A^{\blacktriangle} otherwise. The attainment function $\alpha_{\mathcal{A}}$ of \mathcal{A} is then

$$\alpha_{\mathcal{A}}(\mathbf{y}) = \begin{cases} 0, & \mathbf{y} \notin A^{\blacksquare} \cup A^{\blacktriangle} \\ 0.1, & \mathbf{y} \in A^{\blacksquare} \setminus A^{\blacktriangle} \\ 0.9, & \mathbf{y} \in A^{\blacktriangle} \setminus A^{\blacksquare} \\ 1, & \mathbf{y} \in A^{\blacksquare} \cap A^{\blacktriangle} \end{cases}$$

which we can interpret as the probability of \mathbf{y} being in the random set \mathcal{A} .

4.1.1 PF and PS Estimation

If an attained set \mathcal{A} has uncountably many possible set realizations, then 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 $\left\{0, \frac{1}{N}, \frac{2}{N}, \dots, \frac{N-1}{N}, 1\right\}$.

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

With the empirical attainment function, we can quantify the uncertainty of the CPFs using an $\alpha_{RS}\%$ (where $0 < \alpha_{RS} < 1$) PF UQ point cloud, which we define to be the set of all CPF points \mathbf{y} such that $\hat{\alpha}_N(\mathbf{y}) \in (0.5 - \alpha_{RS}/2, 0.5 + \alpha_{RS}/2)$. That is, this UQ point cloud is the set of CPF points dominated by some proportion of CPFs, where this proportion is 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 $i = 1, \dots, N$. A single “ $\mathbf{y} \in A^{(i)}$ ” check assesses \mathbf{y} ’s dominance relationship with possibly every point in the CPF $c^{(i)}$ and thus takes $\mathcal{O}(d|c^{(i)}|)$ time. Checking “ $\mathbf{y} \in A^{(i)}$ ” for all CPF points \mathbf{y} and all $i = 1, \dots, N$ then takes $\mathcal{O}(d(\sum_{i=1}^N |c^{(i)}|)^2)$ time.

Regarding PS estimation, recall from [Theorem 1](#) that each image point of a d -output BART function $\mathcal{E}(\cdot; \boldsymbol{\theta})$ corresponds to a partitioning hyperrectangle. Hence, the PS of $\mathcal{E}(\cdot; \boldsymbol{\theta})$ corresponds to a collection of hyperrectangles. This article thus quantifies the uncertainty of conditional PSs to be the union of the hyperrectangles corresponding to the points in the PF UQ point cloud: $\widehat{\mathcal{P}}_{\mathcal{X}} := \bigcup_{\mathbf{y} \in \widehat{\mathcal{P}}_{\mathcal{Y}}} \mathcal{E}^{-1}(\mathbf{y}; \boldsymbol{\theta})$, where $\widehat{\mathcal{P}}_{\mathcal{Y}}$ is a PF UQ point cloud and $\mathcal{E}^{-1}(\mathbf{y}; \boldsymbol{\theta})$ is the hyperrectangle corresponding to the objective point \mathbf{y} .

4.2 Band Depth Approach

A second approach to quantify the variability of CPFs $c^{(1)}, c^{(2)}, \dots, c^{(N)}$ (with associated DPSCs $A^{(1)}, A^{(2)}, \dots, A^{(N)}$) 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) introduced the concept of band depth for univariate functions. Whitaker, Mirzargar, and Kirby (2013) generalized this band depth definition to operate on sets, which we use to order $c^{(1)}, c^{(2)}, \dots, c^{(N)}$. We say that a CPF $c^{(i)}$ lies in the band delimited by two CPFs $c^{(j)}$ and $c^{(k)}$ if and only if $[A^{(j)} \cap A^{(k)}] \subseteq A^{(i)} \subseteq [A^{(j)} \cup A^{(k)}]$. We denote this relationship by $c^{(i)} \subseteq^* B(c^{(j)}, c^{(k)})$. Figure 4 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 $c^{(1)}, c^{(2)}, \dots, c^{(N)}$, the band depth of CPF $c^{(i)}$ is $BD_N(c^{(i)}) = \binom{N}{2}^{-1} \sum_{j=1}^{N-1} \sum_{k=j+1}^N \mathbb{I}(c^{(i)} \subseteq^* B(c^{(j)}, c^{(k)}))$.

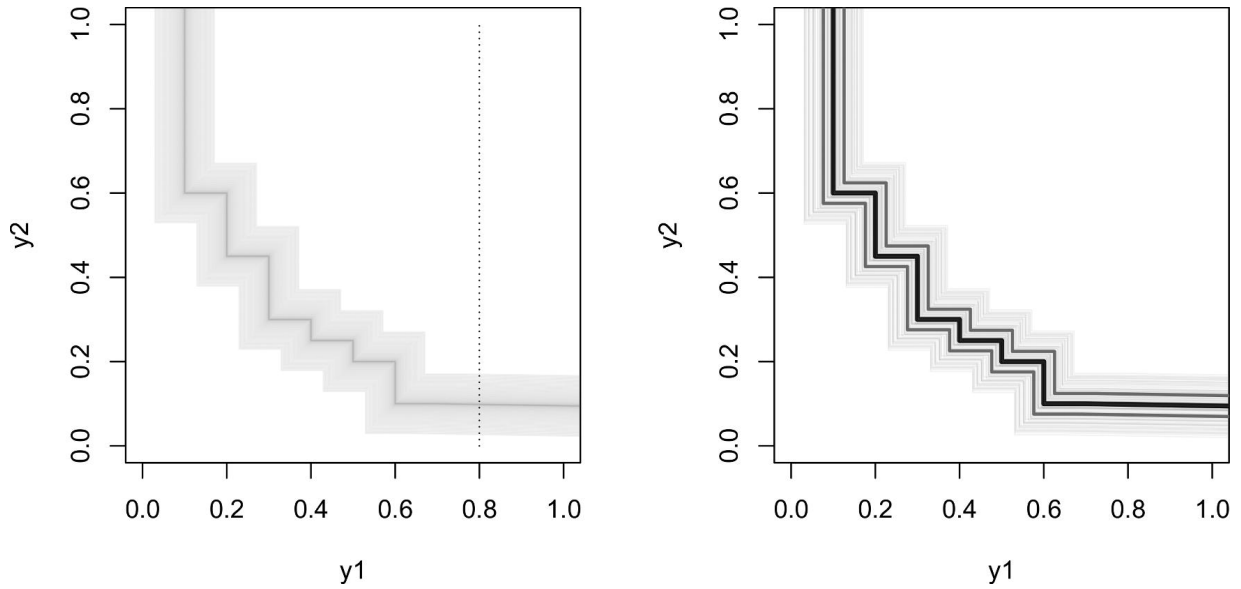


Fig. 4 The lower-left boundaries of $N = 101$ DPSCs. Left: faint gray curves correspond to the N CPFs. Right: thick black curve corresponds to the deepest CPF. Thick gray curves correspond to the 50% deepest CPFs corresponding to $\alpha_{\text{MBD}} = 0.5$.

4.2.1 Modified Band Depth

Whitaker, Mirzargar, and Kirby (2013) noted 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 one-dimensional functions, López-Pintado and Romo (2009) defined a modified band depth that measures the proportion of a function's graph that is in the band. Sun, Genton, and Nychka (2012) introduced an efficient algorithm to compute this modified band depth. We extend this efficient algorithm to compute modified band depth for d -dimensional CPFs, where $d \geq 2$.

Example 3. *We motivate our algorithm through an illustrative example in which we compute the depth of an arbitrary CPF, denoted as $c^{(*)}$, among the $N = 101$ CPFs in Figure 4, where we assume $\mathcal{Y} = [0, 1]^2$. First, we define and compute $c^{(*)}$'s depth at the vertical dotted line $y_1 = 0.8$, which we denote as $d^{(*)}|_{y_1=0.8}$. Let $A^{(*)}|_{y_1=0.8} = \min\{y_2 \in [0, 1] : (0.8, y_2) \in A^{(*)}\}$ be the minimum y_2 value of the lower-left boundary of DPSC $A^{(*)}$ 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 $c^{(*)}$ at the line $y_1 = 0.8$ if and only if $A^{(*)}|_{y_1=0.8}$ is in the closed interval $[A^{(j)}|_{y_1=0.8}, A^{(k)}|_{y_1=0.8}]$. Then $d^{(*)}|_{y_1=0.8}$ is defined to be the fraction of pairs of the N CPFs that contain $c^{(*)}$, where the number of CPF pairs that contain $c^{(*)}$ is $|\{i : A^{(i)}|_{y_1=0.8} \leq A^{(*)}|_{y_1=0.8}\}| \times |\{i : A^{(i)}|_{y_1=0.8} \geq A^{(*)}|_{y_1=0.8}\}| - 1$ (we subtract one to avoid counting the band delimited by $c^{(*)}$ with itself). Note that $|\{i : A^{(i)}|_{y_1=0.8} \leq A^{(*)}|_{y_1=0.8}\}|$ is simply the rank of $A^{(*)}|_{y_1=0.8}$ among $\{A^{(i)}|_{y_1=0.8} : i = 1, \dots, N\}$ and is also equal to $N - |\{i : A^{(i)}|_{y_1=0.8} \geq A^{(*)}|_{y_1=0.8}\}| + 1$.*

We may repeat the process above for any vertical line, e.g. $y_1 = s$, to obtain the depth $d^{()}|_{y_1=s}$ of CPF $c^{(*)}$ at $y_1 = s$. Similarly, we may easily alter the process above to obtain depth $d^{(*)}|_{y_2=t}$ of $c^{(*)}$ at any horizontal line, for example, $y_2 = t$. If we create a dense uniform grid of q lines for each output*

dimension and find the depth of $c^{(*)}$ at each of the $2q$ lines, then we can approximate the “overall” depth of $c^{(*)}$ by the sample mean of these $2q$ depths: $0.5q^{-1} \sum_{j=1}^q d^{(*)}|_{y_1=t_j} + 0.5q^{-1} \sum_{j=1}^q d^{(*)}|_{y_2=t_j}$, where $t_j := (j - 1)/(q - 1)$.

Following a process similar to Example 3, Section B.2 of the Supplement provides an explicit algorithm to find the depth of all N CPFs when $d = 2$. Step 1 creates the $2q$ lines while Step 2 finds the y_2 and y_1 intersection values for all N CPFs. At each line, Step 3 ranks the CPFs while Step 4 computes for each CPF the number of pairs of CPFs that contain it. Step 5 then calculates the depth of each CPF. Section B.3 of the supplement material extends this $d = 2$ process to any $d \geq 2$, which has a runtime of $\mathcal{O}(dq^{d-1}N \log N)$.

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 $0 < \alpha_{\text{MBD}} < 1$. 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

To compare the runtime of the two approaches for any $d \geq 2$, we first express the random-sets runtime as $\mathcal{O}(dN^2\bar{n}^2)$, where $\bar{n} := N^{-1} \sum_{i=1}^N |c^{(i)}|$ is the mean number of points in each CPF. Inference can be controlled by varying either the number of posterior draws (N) or the training size (n). If we fix N and increase n (and hence also \bar{n}), the random-sets runtime grows quadratically in \bar{n} , but we would also want to grow q proportionally to \bar{n} in order to faithfully capture the ranks of the increasingly refined CPFs, which would affect the depth runtime via q^{d-1} . Hence, the comparison between the two runtimes depends on d . In practice,

however, the training size is fixed (which also roughly fixes \bar{n}). As N increases, the depth runtime grows more slowly than does the random-sets runtime ($N \log N$ vs N^2), which makes the depth approach the more computationally tractable option in this case.

5 Simulation Study

5.1 Simulation Settings

We generate data from one of four test functions: MOP2 (Fonseca and Fleming 1995), ZDT3 (Zitzler, Deb, and Thiele 2000), DTLZ2 (Deb et al. 2005), and ZLT1 (Laumanns 2005), which for brevity are defined in the Supplement. Figure 5 shows the MOP2 function ($p = d = 2$) to be the simplest of the four. ZDT3 ($p = d = 2$) has a disconnected PF and PS. DTLZ2's ($p = 4$ and $d = 2$) PF and PS (not shown) are similar to MOP2's. ZLT1's ($p = d = 3$) PF is a convex 2-dimensional surface while its PS is the 2-dimensional probability simplex $\left\{ (x_1, x_2, x_3) \in [0, 1]^3 : x_1 + x_2 + x_3 = 1 \right\}$. Though the methodology in Sections 3.2 and 4 is invariant to shifts and scales of inputs or outputs, our performance metrics in Section 5.2 are not. Thus, we shift and scale the input space to be $\mathcal{X} = [0, 1]^p$ and each objective to have range $[0, 1]$.

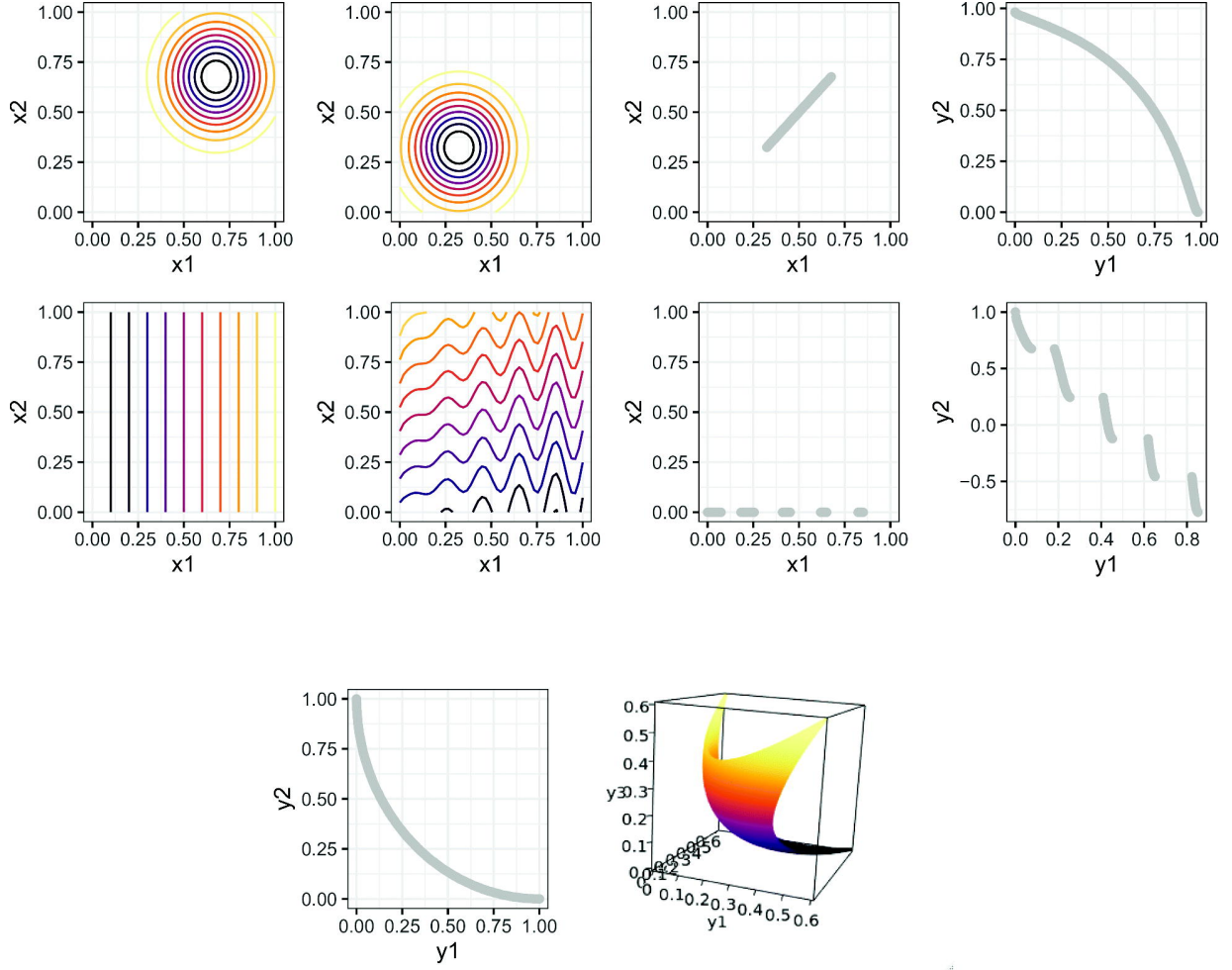


Fig. 5 The first row (MOP2) and second row (ZDT3) show, from left to right, the respective function's f_1 contour, f_2 contour, PS, and PF. For these plots, darker contour lines indicate lower objective values. The third row shows the PF of DTLZ2 (left) and ZLT1 (right), where darker values indicate lower y_2 values.

Given data from (1) 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 UQ.

For each test function $\mathbf{f}(\cdot)$, we explore the six possible combinations of the following parameter settings: noise-variance multiplier

$\zeta \in \{0.0, 0.1, 0.25\}$ and training size $n \in \{32p, 64p\}$. Given any $(\mathbf{f}(\cdot), n, \zeta)$ combination, we can create a dataset $\mathbf{D} := \{(\mathbf{y}(\mathbf{x}_i), \mathbf{x}_i)\}_{i=1}^n$ by simulating $\mathbf{y}(\mathbf{x}_i)$ from (1) at each design point \mathbf{x}_i of an n -point maximin LHS on $[0, 1]^p$ (from Edwin van Dam et al. 2015), where $\sigma_j^2 = \zeta \text{var}(f_j(\mathbf{X}))$ for all $j = 1, \dots, d$ and $\mathbf{X} = (X_1, \dots, X_p)$ with each $X_i \stackrel{\text{iid}}{\sim} U(0, 1)$. For each $(\mathbf{f}(\cdot), n, \zeta)$ combination, we generate either 100 such datasets $\mathbf{D} := \{(\mathbf{y}(\mathbf{x}_i), \mathbf{x}_i)\}_{i=1}^n$ if $\mathbf{f}(\cdot)$ is one of the three $d=2$ functions, or 50 such datasets if $\mathbf{f}(\cdot)$ is ZLT1. To each dataset, we fit two models: a multiple-output BART model and a multiple-output GP model produced by fitting an independent single-output GP to each marginal dataset \mathbf{D}_j for $j = 1, \dots, d$. For each model our BTE (burn-in B steps, terminate after T steps, sample every E steps) is (300, 800, 1), resulting in 500 posterior draws per generated dataset. To summarize, we fit a multiple-output BART model and a multiple-output GP model to each of the 2, 100 datasets, obtain 500 posterior draws from each of the 4,200 models, find or approximate the image at each posterior draw, find the CPF of each image, compute the depth and empirical attainment function value for each CPF to produce PF (and PS if using a BART model) UQ point clouds, and compute performance metrics (defined in Section 5.2) for each point cloud. Running the simulation study pipeline for ZLT1 (where $d=3$) for 50 datasets took roughly 6 days with 32 cores (approximately 8.3 hours per BART model run). However, we note the much shorter runtime for the engineering application in Section 6, where $d=2$.

We use the OpenBT implementation of BART (Pratola 2021) with default parameter settings unless otherwise stated in Section 2. For the GP method, we use the km and simulate functions of Roustant, Ginsbourger, and Deville (2012)’s DiceKriging package with error variance set to the scenario’s noise variance. Because noise variance is not known in most applications, the GP fits can be seen as idealized. For UQ, we use $\alpha_{\text{RS}} = 0.25$ and $\alpha_{\text{MBD}} = 0.5$.

5.2 Performance Metrics

This section defines two performance metrics that jointly quantify how well a point cloud $\widehat{\mathcal{P}}$ (either $\widehat{\mathcal{P}}_{\mathcal{F}}$ or $\widehat{\mathcal{P}}_{\mathcal{X}}$) estimates its target set \mathcal{P} (either $\mathcal{P}_{\mathcal{F}}$ or $\mathcal{P}_{\mathcal{X}}$):

$$d_1 := d(\widehat{\mathcal{P}}, \mathcal{P}) \quad \text{and} \quad d_2 := d(\mathcal{P}, \widehat{\mathcal{P}})$$

where $d(\mathcal{A}, \mathcal{B}) := |\mathcal{A}|^{-1} \sum_{\mathbf{a} \in \mathcal{A}} (\min_{\mathbf{b} \in \mathcal{B}} \|\mathbf{a} - \mathbf{b}\|_2)$ is the average distance from points in a finite point set \mathcal{A} to a set \mathcal{B} (Dubuisson and Jain 1994). These two metrics are analogous to the Type I/Type II error of a hypothesis test: a point cloud with many points far from the target set (similar to false negatives) will have a large d_1 value while a point cloud far away from many points in the target set (similar to false positives) will have a large d_2 value. That is, the metrics d_1 and d_2 measure the degree to which a point cloud exhibits these two undesirable behaviors. As examples, the point clouds in the first and third panels of Table 1 have large d_1 values while the point clouds in the second and third panels have large d_2 values. Conversely, a point cloud with small d_1 and d_2 values, e.g. in the fourth panel, indicates it is a high-performing estimate of the target set.

Table 1 Each plot shows a different PF point cloud (black triangles) that attempts to capture the same, disconnected target set (gray lines). (Table view)

Undercoverage	Overcoverage	Biased coverage	Good coverage
small d_1 , large d_2	large d_1 , small d_2	large d_1 , large d_2	small d_1 , small d_2

NOTE: Of the four point clouds, the one in the fourth panel is the only one that adequately captures target set. Qualitative values of d_1 and d_2 for each point cloud are provided.

Our metric d_1 is equivalent to the function \mathcal{M}_1 in Zitzler, Deb, and Thiele (2000), which is one of 63 performance indicators in the MO

literature reviewed by Audet et al. (2020). Though many of these indicators penalize only one of the two mentioned undesirable behaviors, some more recent indicators penalize both behaviors with a single metric to be used in a sequential MO design. This article, however, focuses on characterizing the performance of estimates of $\mathcal{P}_{\mathcal{F}}$ and $\mathcal{P}_{\mathcal{X}}$. Hence, we penalize these two behaviors separately to see *how* a point cloud might underperform.

5.3 Simulation Results

Figure 6 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 Supplement). 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, Ruts, and Tukey 1999). For visual clarity, we include only two of these features: the depth median, which is the point with the highest possible halfspace depth, and the “bag,” which is a polygon that encloses 50% of the points around the depth median.

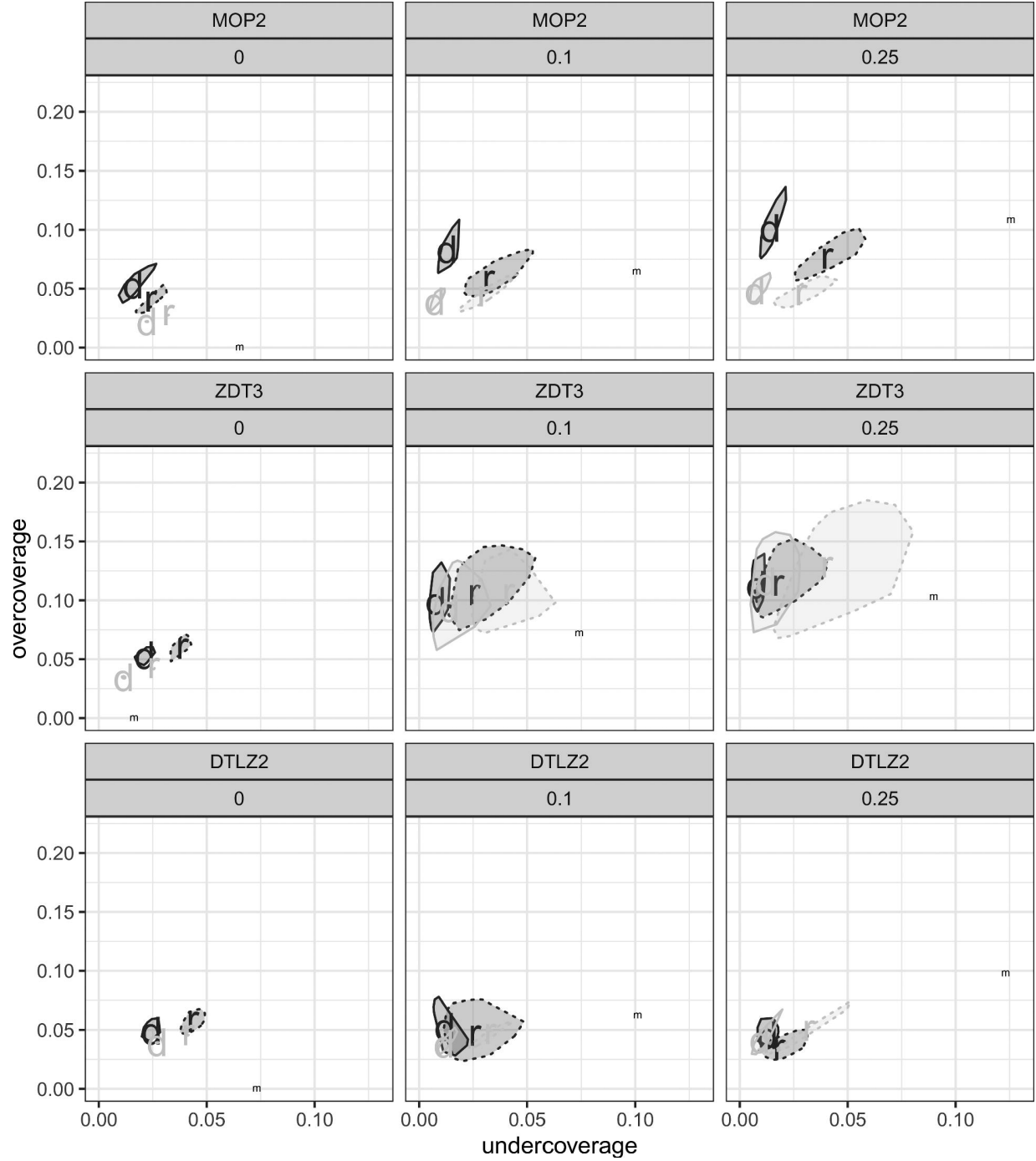


Fig. 6 Bagplots of the 100 values of $d(\widehat{\mathcal{P}}_{\mathcal{F}}, \mathcal{P}_{\mathcal{F}})$ and $d(\mathcal{P}_{\mathcal{F}}, \widehat{\mathcal{P}}_{\mathcal{F}})$. Each plot represents a $n = 64p$ simulation scenario where the variance multiplier is a value in $\{0, 0.1, 0.25\}$. Bags with solid (dashed) outline and median labeled 'd' ('r') display depth (random sets) approach. Black (gray) bags display BART (GP) model.

As another point of comparison, we also plot in [Figure 6](#) 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 Q.1, we make several observations in [Figure 6](#). We first compare the difficulty of MO between the three $d=2$ 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 $d=2$ functions. The ZLT1 PF point cloud ([supplementary material, Figures 5a](#)), however, produces much more undercoverage than those of the other three functions, presumably because 2-dimensional surfaces (e.g., ZLT1’s PF) are usually more difficult to cover than 1-dimensional sets (e.g., the other three PFs). However, the PS point clouds ([supplement, Figures 2b, 3b, 4b, and 5b](#)) perform differently between the three $d=2$ functions. When $d=2$, 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 (when $d=2$), 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. Similar to the PF regime, the ZLT1 PS point cloud produces much more undercoverage than those of the other three functions, which again may be explained by ZLT1’s 2-dimensional PS. Surprisingly, the

ZLT1 PS point clouds have fairly low overcoverage, which suggests we could increase α_{MBD} and α_{RS} to improve undercoverage. From these observations, we conclude that among the $d=2$ functions, ZDT3 has the most difficult PF to capture while DTLZ2 has the most difficult PS to capture, but the increased dimensionality of ZLT1’s PF and PS makes point clouds prone to have large undercoverage.

We also compare performance between BART and (idealized) GP. In the MOP2 PF results, the no-noise scenario shows an 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 Q.2, we refer to [Figures 2b, 3b, 4b, and 5b](#) in the [supplementary material](#). For each $d=2$ 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. For ZLT1, the difference between the two approaches is slightly more pronounced. When $n=96$, there seems to be a tradeoff between overcoverage and undercoverage, but when $n=192$, the depth approach seems to have slightly less undercoverage and roughly equal

overcoverage as the random sets approach. Interestingly, undercoverage of ZLT1's PS point clouds remains large and does not decrease with larger sample size, which suggests there is a region of the PS that the point clouds consistently fail to cover.

To address Q.1, we now compare the depth approach to the random sets approach. We first look at BART's PF point clouds. For the MOP2 and ZLT1 functions, the depth approach tends to produce more overcoverage and less undercoverage than the random sets approach. For ZDT3, the depth approach tends to produce less overcoverage and undercoverage than the random sets approach. For DTLZ2, 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 $d=2$ test functions, which suggests the depth approach produces overall better GP-based PF point clouds than does the random sets approach. From these observations and from the depth approach's runtime advantage as discussed in Section 4.3, we conclude that the depth approach should be used over the random sets approach if using either BART or GP.

6 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, $C_m : \mathcal{X} \rightarrow \mathbb{R}$ and $C_t : \mathcal{X} \rightarrow \mathbb{R}$, respectively, for an industrial engineering application where

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

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 7, no image point is far from the PF. 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 7), PS estimates will tend to have large uncertainty.

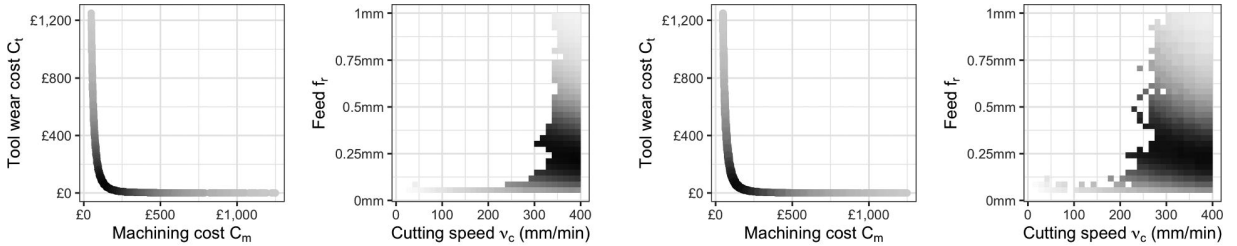


Fig. 7 Left: image (gray) and PF (black) of the single cut turning cost operation biobjective function. Middle: the left plot zoomed into the lower-left region. Right: input space (gray) and PS (black).

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 BTE (300, 2300, 1)) to perform PF/PS estimation and UQ on two training datasets generated from (1), where $\sigma^2 = 0$ and (3) takes the role of $\mathbf{f}(\cdot)$. We generate noiseless data, but approach the problem as if we do not a priori know the level of noise in the observations. 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 5.3. The runtime of this analysis for $n = 15,300$ was 2 min and 45 sec with 32 cores.

Figure 8 shows the 50% PF and PS point clouds for two training sizes: $n = 15,300$ 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 (Figures 8(b) and 8(d)) show large uncertainty. On the other hand, the two PF point clouds (Figures 8(a) and 8(c)) 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 $n = 1500$.



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

Fig. 8 50% UQ point clouds for the PF and PS.

If such a single cut turning cost operation is embedded in a continuously monitored and adapting manufacturing system, it makes sense to allow cutting speed to be continuously manipulated by automated systems using relevant manufacturing information obtained in real-time. For example, a sudden jump in the price of raw materials might increase the impact of

cutting speed on tool wear cost and hence change the nature of the PF. Our methodology provides an automated approach to understanding which input settings will result in low-cost outcomes.

7 Summary and Discussion

Using the fact that BART produces a set of hyperrectangles which partition the input domain ([Theorem 1](#)), this article 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 article suggests several topics for additional research. First, our UQ comparisons used $\alpha_{\text{MBD}} = 0.50$ and $\alpha_{\text{RS}} = 0.25$, 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 article 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, this article used one-stage maximin LHDs. Presumably, BART’s PF and PS estimates could be improved using alternative input designs. For example, Chipman, Ranjan, and Wang ([2012](#)) performed

sequential design using single-output BART prediction, but additional design research is required for multiple-output BART.

Third, this article assumes an independent correlation structure in the distribution of ϵ_i in (1). Introducing dependence would require a modeling approach that cannot be accomplished via independent BART model fits. To date, no such approach exists.

The implementation for finding the PF and PS of a two-output or three-output BART model can be found in the Open Bayesian Trees (OpenBT) project at <https://bitbucket.org/mpratola/openbt/>.

Acknowledgments

AH would like to acknowledge the Graduate School at The Ohio State University for support during the dissertation year.

Funding

TJS was supported in part by the National Science Foundation under Agreements DMS-0806134 and DMS-1310294 (The Ohio State University). YS was supported by King Abdullah University of Science and Technology (KAUST), Office of Sponsored Research (OSR) under Award No: OSR-2019-CRG7-3800. 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.

Supplementary Material

Proof, Algorithms, Function definitions, Additional simulation results Proof of [Theorem 1](#).
Pseudocode for Kung, Luccio, and Preparata (1975)’s algorithm to find the PF of a set of d –dimensional vectors as described in Section 3.2. Description of the algorithm to compute modified band depth as described in Section 4.2.1. Definitions of the test functions introduced in Section 5. ZLT1 simulation study plots, 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)

References

- Audet, C., Bignon, J., Cartier, D., Le Digabel, S., and Salomon, L. (2020), “Performance Indicators in Multiobjective Optimization,” *European Journal of Operational Research*, 292, 397–422. [Crossref](#). [ISI](#).
- Binois, M., Ginsbourger, D., and Roustant, O. (2015), “Quantifying Uncertainty on Pareto Fronts

- With Gaussian Process Conditional Simulations,” *European Journal of Operational Research*, 243, 386–394. [Crossref](#). [ISI](#).
- Breiman, L. (2001), “Random Forests,” *Machine Learning*, 45, 5–32. [Crossref](#). [ISI](#).
- Chipman, H., Ranjan, P., and Wang, W. (2012), “Sequential Design for Computer Experiments With a Flexible Bayesian Additive Model,” *Canadian Journal of Statistics*, 40, 663–678. [Crossref](#). [ISI](#).
- Chipman, H. A., George, E. I., and McCulloch, R. E. (1998), “Bayesian Cart Model Search,” *Journal of the American Statistical Association*, 93, 935–948. [Crossref](#). [ISI](#).
- Chipman, H. A., George, E. I., and McCulloch, R. E. (2010), “Bart: Bayesian Additive Regression Trees,” *The Annals of Applied Statistics*, 4, 266–298. [Crossref](#). [ISI](#).
- da Fonseca, V. G., and Fonseca, C. M. (2010), “The Attainment-Function Approach to Stochastic Multiobjective Optimizer Assessment and Comparison,” in *Experimental Methods for the Analysis of Optimization Algorithms*, eds. T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, Berlin: Springer, pp. 103–130. [Crossref](#).
- Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005), “Scalable Test Problems for Evolutionary Multiobjective Optimization,” in *Evolutionary Multiobjective Optimization*, eds. A. Abraham, L. Jain, and R. Goldberg, London: Springer, pp. 105–145. [Crossref](#).
- Dubuisson, M.-P., and Jain, A. K. (1994), “A Modified Hausdorff Distance for Object Matching,” in *Proceedings of 12th International Conference on Pattern Recognition* (Vol. 1). IEEE, pp. 566–568. [Crossref](#).
- Edwin van Dam, D. H., Husslage, B., and Rennen, G. (2015), “Space-Filling Designs.”
- Fonseca, C. M., and Fleming, P. J. (1995), “Multiobjective Genetic Algorithms Made Easy: Selection Sharing and Mating Restriction,” *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, available at <https://ieeexplore.ieee.org/document/501646>. [Crossref](#).
- Horiguchi, A., Pratola, M. T., and Santner, T. J. (2021), “Assessing Variable Activity for Bayesian Regression Trees,” *Reliability Engineering & System Safety*, 207, 107391. [Crossref](#). [ISI](#).
- Kung, H.-T., Luccio, F., and Preparata, F. P. (1975), “On Finding the Maxima of a Set of Vectors,” *Journal of the ACM (JACM)*, 22, 469–476. [Crossref](#). [ISI](#).
- Laumanns, M. (2005), “Self-adaptation and Convergence of Multiobjective Evolutionary Algorithms in Continuous Search Spaces,” in *Evolutionary Multiobjective Optimization*, Springer, pp. 33–53. [Crossref](#).
- López-Pintado, S. and J. Romo (2009), “On the Concept of Depth for Functional Data,” *Journal of the American Statistical Association*, 104, 718–734. [Crossref](#). [ISI](#).
- Pratola, M. T. (2021), “Open Bayesian Trees,” available at <https://bitbucket.org/mpratola/openbt/>.
- Pratola, M. T., Chipman, H. A., Gattiker, J. R., Higdon, D. M., McCulloch, R., and Rust, W. N. (2014), “Parallel Bayesian Additive Regression Trees,” *Journal of Computational and Graphical Statistics*, 23, 830–852. [Crossref](#). [ISI](#).
- Rousseeuw, P. J., Ruts, I., and Tukey, J. W. (1999), “The Bagplot: A Bivariate Boxplot,” *The American Statistician*, 53, 382–387. [Crossref](#). [ISI](#).
- Roustant, O., Ginsbourger, D., and Deville, Y. (2012), “DiceKriging, DiceOptim: Two R packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization,” *Journal of Statistical Software*, 51, 1–55. [Crossref](#). [PubMed](#). [ISI](#).
- Shukla, A. K., Nath, R., Muhuri, P. K., and Lohani, Q. D. (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. [Crossref](#). [ISI](#).
- Sun, Y., Genton, M. G., and Nychka, D. W. (2012), “Exact Fast Computation of Band Depth for Large Functional Datasets: How Quickly Can One Million Curves be Ranked?” *Statistics*, 1, 68–74. [Crossref](#).
- Trautmann, H., and Mehnen, J. (2009), “Preference-Based Pareto Optimization in Certain and Noisy Environments,” *Engineering Optimization*, 41, 23–38. [Crossref](#). [ISI](#).
- Whitaker, R. T., Mirzargar, M., and Kirby, R. M. (2013), “Contour Boxplots: A Method for Characterizing Uncertainty in Feature Sets From Simulation Ensembles,” *IEEE Transactions on Visualization and Computer Graphics*, 19, 2713–2722. [Crossref](#). [PubMed](#). [ISI](#).
- Zitzler, E., Deb, K., and Thiele, L. (2000), “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results,” *Evolutionary Computation*, 8, 173–195. [Crossref](#). [PubMed](#). [ISI](#).

Sections

1. Abstract
2. 1 Introduction
3. 2 BART
 1. 2.1 Multiple-Output BART
 2. 2.2 Prior Specification
4. 3 Multiobjective Optimization
 1. 3.1 Finding the Image of a Multiple-Output BART Function
 2. 3.2 Finding the PF and PS of Multiple-Output BART Function
5. 4 Uncertainty Quantification
 1. 4.1 Random-Sets Approach
 2. 4.2 Band Depth Approach
 3. 4.3 Comparing Time Complexity
6. 5 Simulation Study
 1. 5.1 Simulation Settings
 2. 5.2 Performance Metrics
 3. 5.3 Simulation Results
7. 6 Engineering Application
8. 7 Summary and Discussion
9. Acknowledgments
10. Funding
11. Supplementary Material
12. References

List of Illustrations

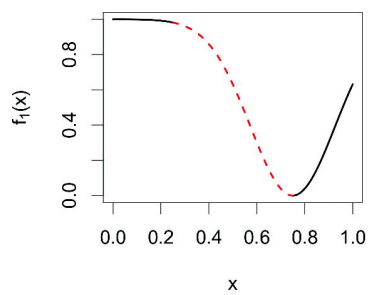
1. Fig. 1
2. Fig. 2
3. Fig. 3
4. Fig. 4

5. Fig. 5

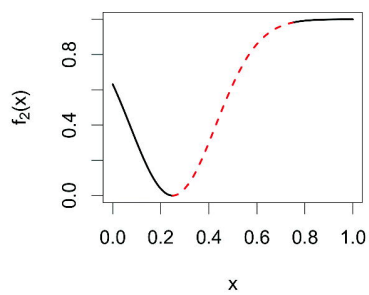
6. Fig. 6

7. Fig. 7

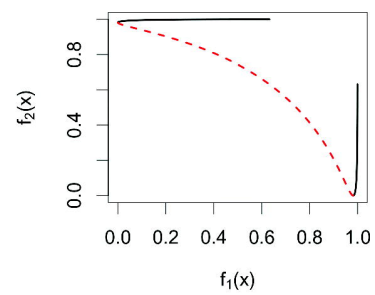
8. Fig. 8



(a) $f_1(x)$ vs x .



(b) $f_2(x)$ vs x .



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

Fig. 1 A biobjective function $(x) = (f_1(x), f_2(x))$ where $x \in [0, 1]$.

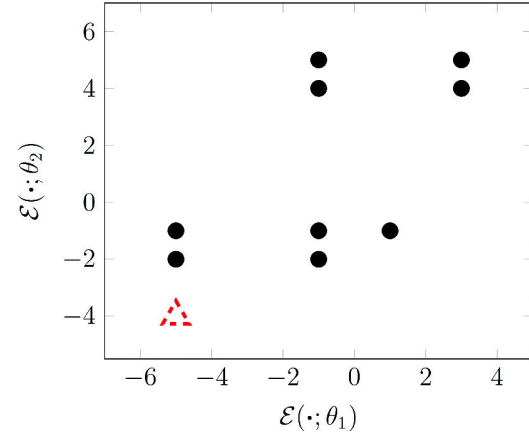
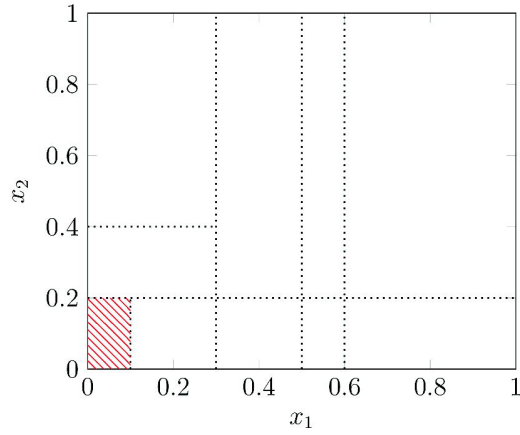
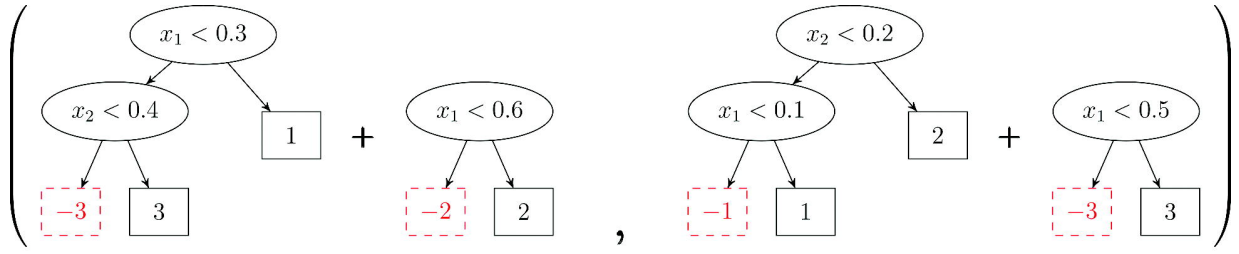


Fig. 2 A biobjective function $\mathcal{E}(\cdot; \cdot) = (\mathcal{E}(\cdot; \theta_1), \mathcal{E}(\cdot; \theta_2))$ with tree ensembles (top row) with parameters θ_1 (left) and θ_2 (right). Bottom left: input space partitioned into 10 rectangles. Bottom right: output space with all 10 image points of $\mathcal{E}(\cdot; \cdot)$.

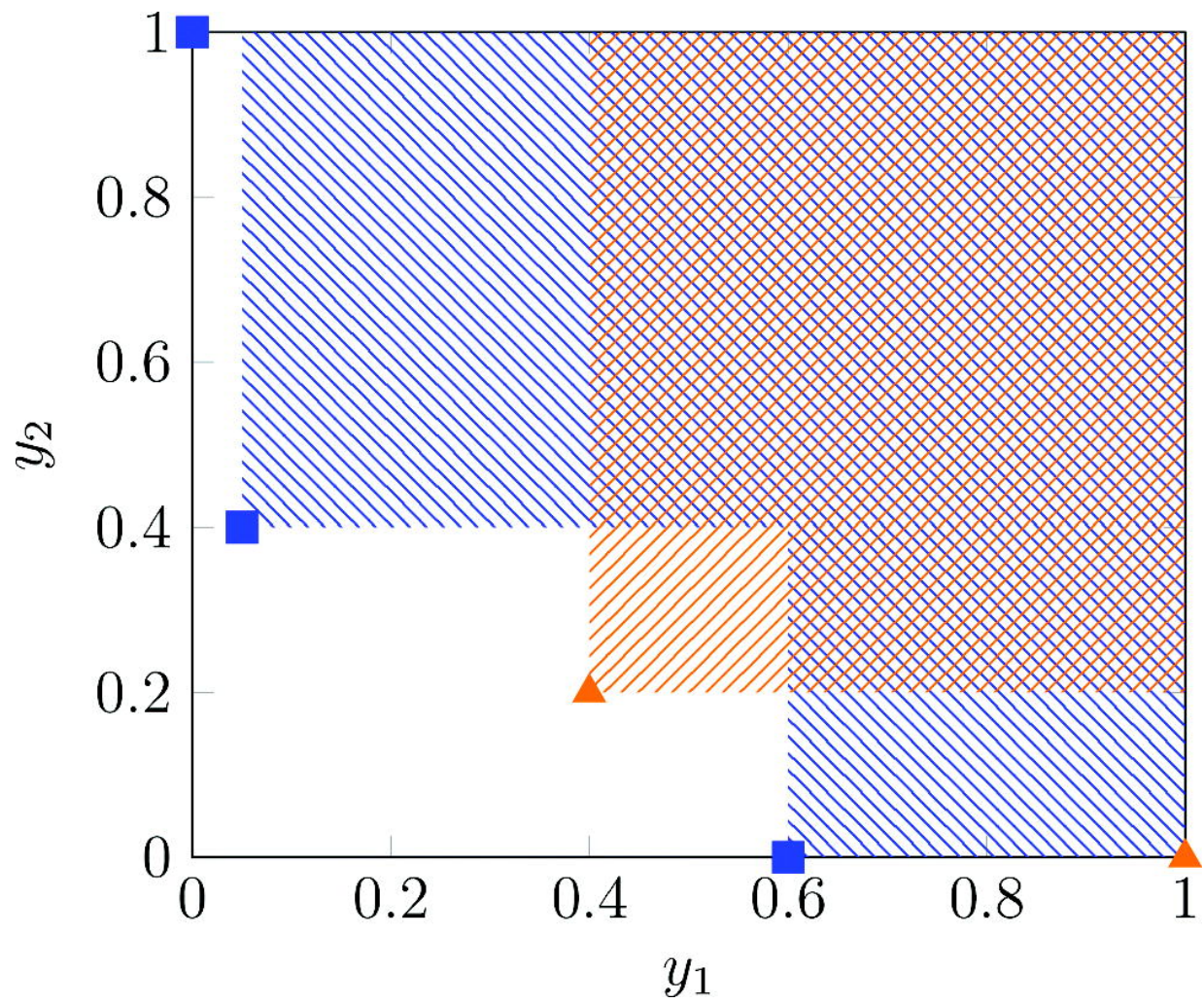


Fig. 3 Example of 2 CPFs (triangles and squares) and the corresponding DPSCs (areas with hatched lines) in objective space

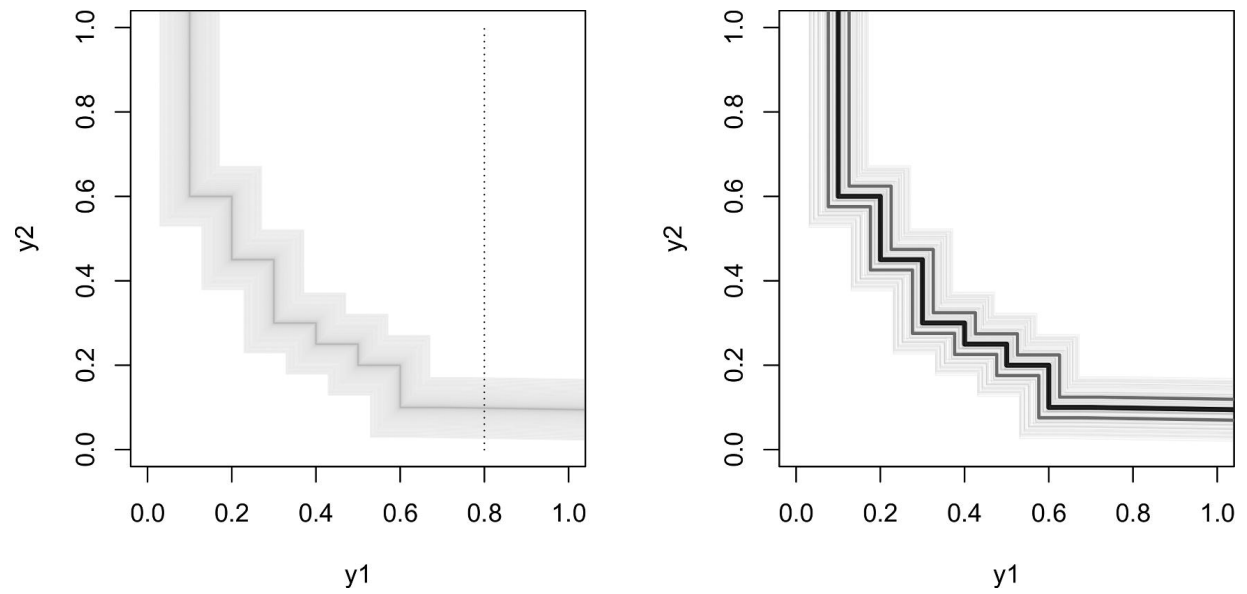


Fig. 4 The lower-left boundaries of $N = 101$ DPSCs. Left: faint gray curves correspond to the N CPFs. Right: thick black curve corresponds to the deepest CPF. Thick gray curves correspond to the 50% deepest CPFs corresponding to .

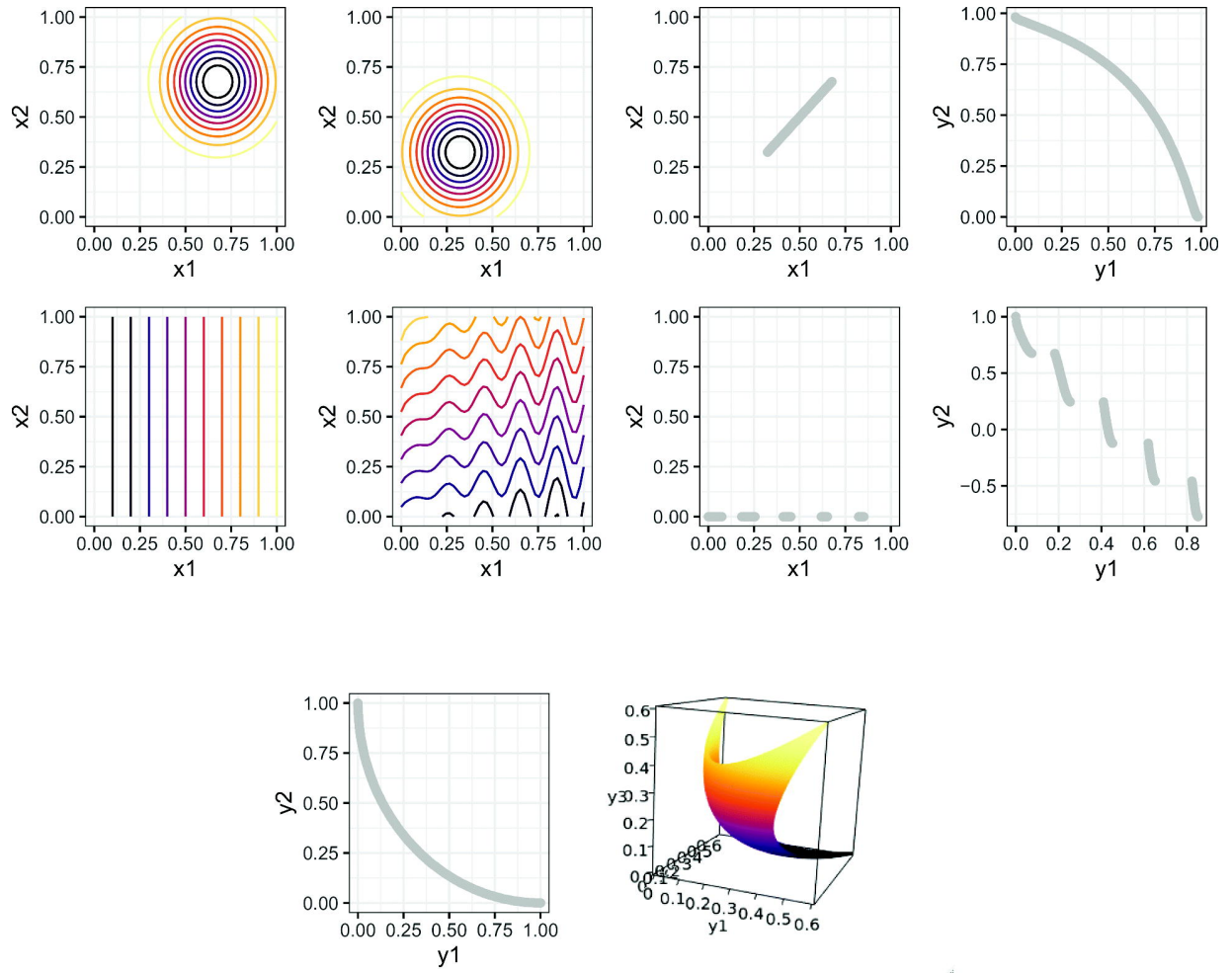


Fig. 5 The first row (MOP2) and second row (ZDT3) show, from left to right, the respective function's f_1 contour, f_2 contour, PS, and PF. For these plots, darker contour lines indicate lower objective values. The third row shows the PF of DTLZ2 (left) and ZLT1 (right), where darker values indicate lower y_2 values.

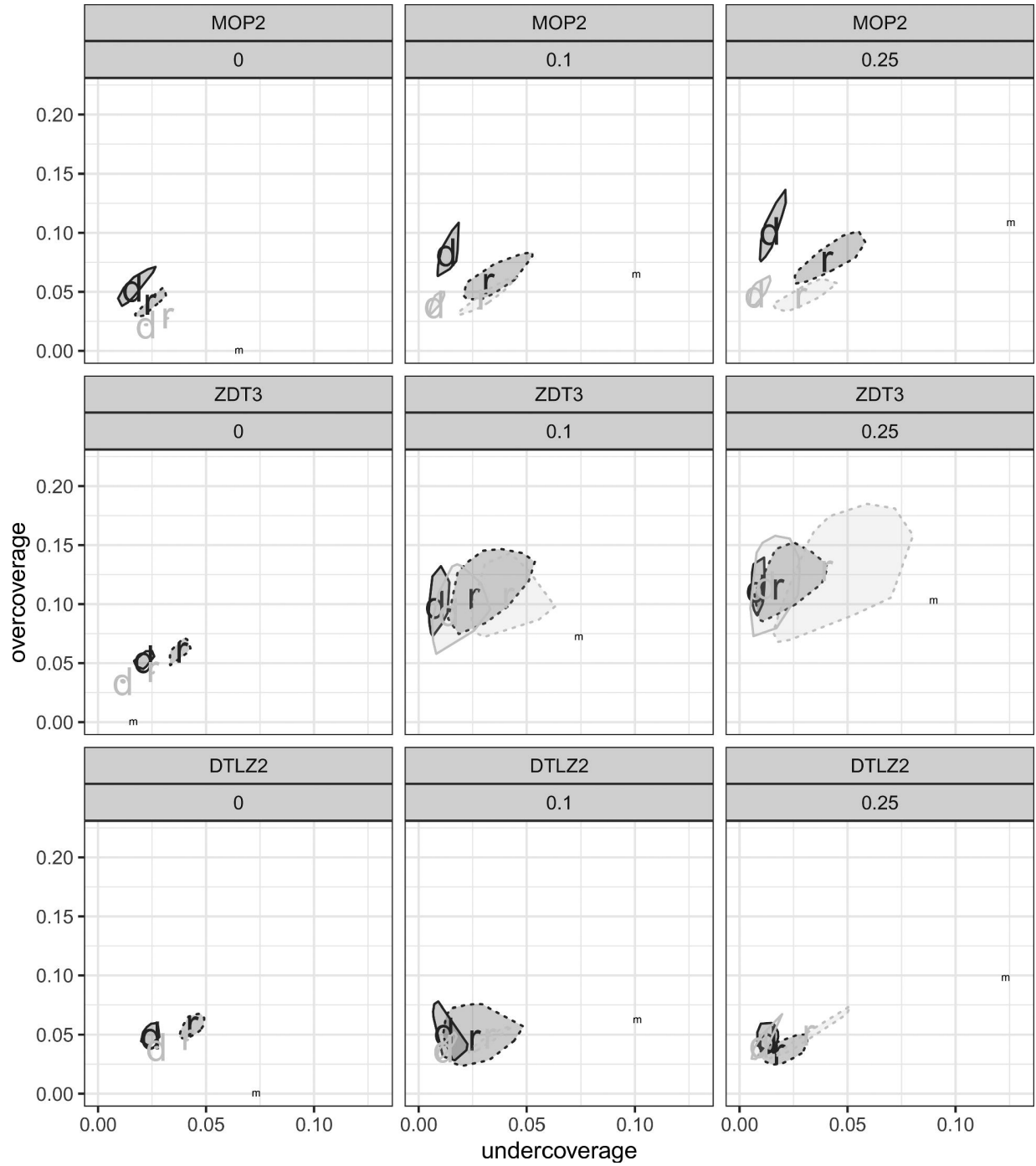


Fig. 6 Bagplots of the 100 values of $d(\mathcal{P}_{\mathcal{F}}, \mathcal{P}_{\mathcal{F}})$ and $d(\mathcal{P}_{\mathcal{F}}, \mathcal{P}_{\mathcal{F}})$. Each plot represents a $n = 64p$ simulation scenario where the variance multiplier is a value in $\{0, 0.1, 0.25\}$. Bags with solid (dashed) outline and median labeled 'd' ('r') display depth (random sets) approach. Black (gray) bags display BART (GP) model.

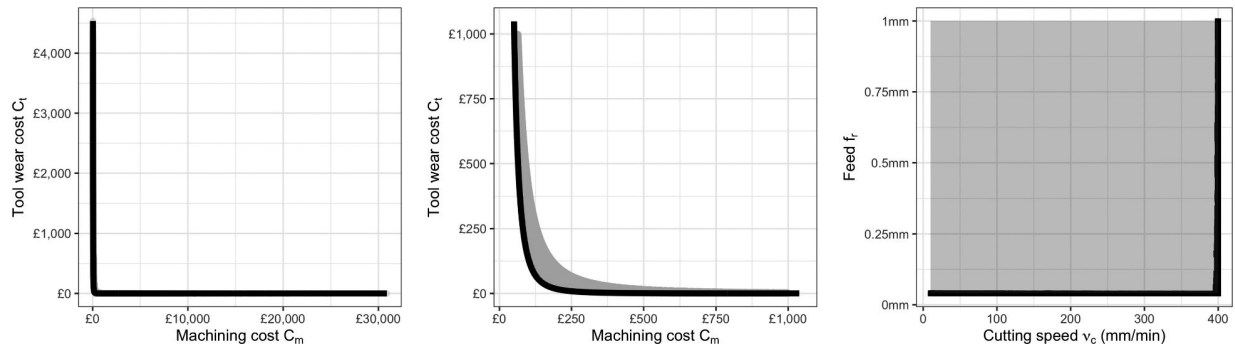
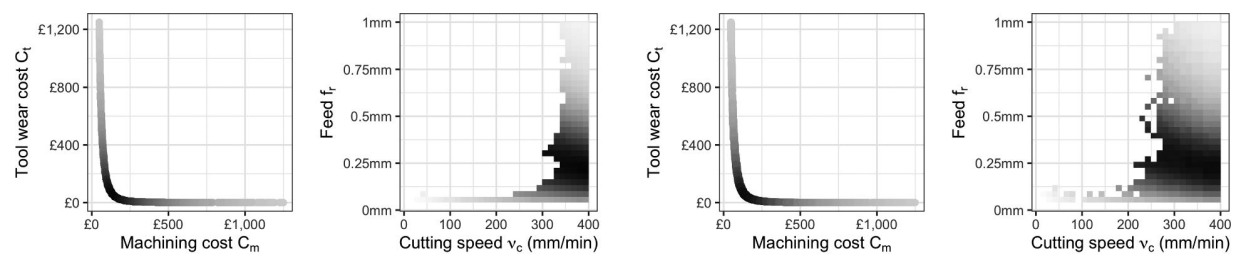


Fig. 7 Left: image (gray) and PF (black) of the single cut turning cost operation biobjective function. Middle: the left plot zoomed into the lower-left region. Right: input space (gray) and PS (black).



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

Fig. 8 50% UQ point clouds for the PF and PS.

Table 1 Each plot shows a different PF point cloud (black triangles) that attempts to capture the same, disconnected target set (gray lines).

Undercoverage	Overcoverage	Biased coverage	Good coverage
small d_1 , large d_2	large d_1 , small d_2	large d_1 , large d_2	small d_1 , small d_2

NOTE: Of the four point clouds, the one in the fourth panel is the only one that adequately captures target set. Qualitative values of d_1 and d_2 for each point cloud are provided.