Diversity-Guided Multi-Objective Bayesian Optimization With Batch Evaluations

Mina Konaković Luković[*]
MIT CSAIL
minakl@mit.edu

Yunsheng Tian[*]
MIT CSAIL
yunsheng@csail.mit.edu

Wojciech Matusik
MIT CSAIL
wojciech@csail.mit.edu

Abstract

Many science, engineering, and design optimization problems require balancing the trade-offs between several conflicting objectives. The objectives are often black-box functions whose evaluations are time-consuming and costly. Multi-objective Bayesian optimization can be used to automate the process of discovering the set of optimal solutions, called Pareto-optimal, while minimizing the number of performed evaluations. To further reduce the evaluation time in the optimization process, testing of several samples in parallel can be deployed. We propose a novel multi-objective Bayesian optimization algorithm that iteratively selects the best batch of samples to be evaluated in parallel. Our algorithm approximates and analyzes a piecewise-continuous Pareto set representation. This representation allows us to introduce a batch selection strategy that optimizes for both hypervolume improvement and diversity of selected samples in order to efficiently advance promising regions of the Pareto front. Experiments on both synthetic test functions and real-world benchmark problems show that our algorithm predominantly outperforms relevant state-of-the-art methods. The code is available at `https://github.com/yunshengtian/DGEMO`.

1 Introduction

Various experimental design problems in science and engineering seek optimal solutions that require simultaneous optimization of a number of conflicting objectives. Typically the objectives are black-box functions that are expensive to evaluate. Hence the number of evaluated experiments is severely limited and designing experiments by hand does not provide optimal performance. Recently, there has been an increasing interest in effective data-driven algorithms to efficiently guide the experimental design process and lead to the best performing designs. Some use cases include material design [53], battery optimization [1], clinical drug trials [50, 47], and chemical design [16]. An approach that has proven to be powerful in optimizing the expensive-to-evaluate black-box functions is Bayesian optimization (BO) [20, 41]. A fundamental concept behind the BO is two-fold: first, a inexpensive surrogate model is used to model the black-box objective function based on the evaluated experiments; second, an acquisition function is employed to adaptively sample the design space and efficiently improve the set of optimal solutions. Both single- and multi-objective Bayesian optimization (MOBO) have been studied.

In a variety of physical experimental setups, evaluating several experiments in parallel, e.g., in batches, reduces the time and cost of the optimization process. Experiments can take days, weeks, even months to complete; hence taking advantage of batching is important. Furthermore, in design problems with multiple conflicting objectives, in most cases, there is no single optimal solution, but rather a set of optimal designs with different trade-offs. These designs often tend to group in different regions based on their properties and performance. In this paper, we develop a novel MOBO approach

[*]Equal contribution.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

with a batch selection strategy that combines the knowledge from both design and performance space. Our selection strategy enforces the exploration of diverse identified regions in approximated optimal space. In addition, it measures which regions are the most beneficial for optimization to explore by taking into consideration the *hypervolume improvement*[37]. A relatively limited amount of literature can be found on the batch selection for MOBO that applies to the problem setup we are trying to address [6, 52]. To the best of our knowledge, no previous published work has considered diversity in both design and performance space in the batch selection strategy.

In summary, the key contributions of this paper are the following:

- We propose a novel approach, referred to as DGEMO (Diversity-Guided Efficient Multi-objective Optimization), for solving multi-objective optimization problems of black-box functions, while minimizing the number of function evaluations. Our algorithm is based on multi-objective Bayesian optimization that operates on batches of evaluated samples in each iteration and can handle an arbitrary batch size.
- Our batch selection strategy is guided by diversity in both design and performance space of selected samples. The selection strategy divides the space into diversity regions that provide additional information in terms of shared properties and performance of optimal solutions. This information can be valuable for further physical experiments and problem understanding.
- We conduct comprehensive experiments on both synthetic test functions and real-world benchmark problems. Our algorithm exhibits consistent high-quality performance and outperforms the relevant existing works in the field.

2 Related work

Bayesian optimization (BO) BO originated from [24, 33], and was popularized by the Efficient Global Optimization (EGO) algorithm [20]. BO achieves a minimal number of function evaluations by utilizing the surrogate model and sampling guided by carefully designed selection criteria. There are two types of selection criteria: *sequential selection* that proposes only a single optimal sample to evaluate, and *batch selection* proposing a set of samples to evaluate in each iteration. The sequential selection has long been investigated in the literature [24, 20, 43, 17], which usually achieves better performance w.r.t. the number of evaluations, at the cost of more algorithm iterations and slower convergence. On the other hand, batch selection sacrifices some performance, yet can benefit from a parallel evaluation, which is more common and time-efficient in modern engineering experiments [11, 8, 21, 15]. In this paper, we focus on the time-efficient batch selection scheme.

Multi-objective optimization (MOO) MOO is applied to problems involving several conflicting objectives and optimizes for a set of Pareto-optimal solutions [32]. To this end, many population-based multi-objective evolutionary algorithms (MOEA) are proposed, e.g., NSGA-II [9] and MOEA/D [51], which are widely used in various multi-objective problems, including financial portfolio design [45], manufacturing design [39], and optimal control [13]. However, MOEA algorithms typically require a substantial number of evaluations due to the nature of evolutionary computation. This requirement prevents them from being applied to many real-world problems, where the evaluation can be computationally expensive and becomes the main bottleneck of the whole optimization process.

Multi-objective Bayesian optimization (MOBO) Taking the best of both worlds from Bayesian optimization and multi-objective optimization, MOBO is designed to solve multi-objective problems that are expensive to evaluate. Among MOBO algorithms there are also two streams of work focusing on sequential selection and batch selection; we call them the *single-point* method and *batch* method. One of the first single-point methods is ParEGO [22], which randomly scalarizes the multi-objective problem to a single-objective problem and selects a sample with maximum *expected improvement* (EI). Similarly, EHI [12] and SUR [35] extend ParEGO by using different acquisition functions. Another line of work, such as PAL [56], focuses on uncertainty reduction on the surrogate model for better performance. PESMO [38] and MESMO [3] rely on entropy-based acquisition functions and select a sample that maximizes the information gain about the optimal Pareto set. A very recent approach USeMO [2] uses maximum uncertainty as a selection criterion based on the Pareto set generated by NSGA-II. USeMO achieves state-of-the-art performance and generalizes to any acquisition function.

Even though these algorithms fall into the category of the single-point method, some of them could have a straight-forward extension to the batch selection scheme, which we demonstrate in Section 5.

For batch MOBO methods, MOEA/D-EGO [52] can be considered as the earliest attempt, which generalizes ParEGO by multiple scalarization weights and performs parallel optimization with MOEA/D. A subsequent work MOBO/D is proposed as an extension to MOEA/D-EGO by changing the acquisition function [25]. A recent work TSEMO [6] suggests using Thompson Sampling (TS) on the GP posterior as an acquisition function, optimizing multiple objectives with NSGA-II, and selecting the next batch of samples by maximizing the hypervolume improvement. BS-MOBO [26] applies MOBO on a different setting with large scale datasets using neural network as surrogate model. However, to the best of our knowledge, none of the previous methods consider diversity in both design and performance space in the batch selection, which can be crucial for many real-world problems where the Pareto-optimal solutions are distributed in diverse regions of the design space.

3 Preliminaries

3.1 Multi-objective optimization

We consider a multi-objective optimization problem over a set of continuous input variables $\mathcal{X} \subset \mathbb{R}^d$, called *design space*. The goal is to simultaneously minimize $m \geq 2$ objective functions $f_1, ..., f_m : \mathcal{X} \to \mathbb{R}$. We denote the vector of all objectives as $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_m(\mathbf{x}))$, where $\mathbf{x} \in \mathcal{X}$ is a vector of input variables. The *performance space* is then an m-dimensional image $\mathbf{f}(\mathcal{X}) \subset \mathbb{R}^m$.

Objectives are often conflicting, resulting in a set of optimal solutions, rather than a single best solution. These optimal solutions are referred to as *Pareto set* $\mathcal{P}_s \subseteq \mathcal{X}$ in the design space, and the corresponding images in performance space are *Pareto front* $\mathcal{P}_f = \mathbf{f}(\mathcal{P}_s) \subset \mathbb{R}^m$. More formally, a point $\mathbf{x}^* \in \mathcal{P}_s$ is considered Pareto-optimal if there is no other point $\mathbf{x} \in \mathcal{X}$ such that $f_i(\mathbf{x}^*) \geq f_i(\mathbf{x})$ for all i and $f_i(\mathbf{x}^*) > f_i(\mathbf{x})$ for at least one i.

To measure the quality of an approximated Pareto front, *hypervolume indicator* [55] is the most commonly used metric in MOO [37]. Let $\mathcal{P}_f$ be a Pareto front approximation in an m-dimensional performance space and given a reference point $\boldsymbol{r} \in \mathbb{R}^m$, the hypervolume $\mathcal{H}(\mathcal{P}_f)$ is defined as

$$\mathcal{H}(\mathcal{P}_f) = \int_{\mathbb{R}^m} \mathbb{1}_{H(\mathcal{P}_f)}(z) dz \tag{1}$$

where $H(\mathcal{P}_f) = \{\boldsymbol{z} \in Z \mid \exists 1 \leq i \leq |\mathcal{P}_f| : \boldsymbol{r} \preceq \boldsymbol{z} \preceq \mathcal{P}_f(i)\}$. $\mathcal{P}_f(i)$ is the i-th solution in $\mathcal{P}_f$, $\preceq$ is the relation operator of objective dominance, and $\mathbb{1}_{H(\mathcal{P}_f)}$ is a Dirac delta function that equals 1 if $z \in H(P_f)$ and 0 otherwise. The higher the hypervolume, the better $\mathcal{P}_f$ approximates the true Pareto front. To determine how much the hypervolume would increase if a set of new points $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subset \mathbb{R}^m$ is added to the current Pareto front approximation $\mathcal{P}_f$, we can use the *hypervolume improvement* that is defined as

$$\mathrm{HVI}(P, \mathcal{P}_f) = \mathcal{H}(\mathcal{P}_f \cup P) - \mathcal{H}(\mathcal{P}_f). \tag{2}$$

3.2 Bayesian optimization

A variety of optimization problems search to find a global optimal solution of a black-box function $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ that is expensive to evaluate. Neither analytical form nor derivatives of f are known, and a limited number of function evaluations are available. In this scenario, Bayesian optimization (BO) has proven to be a powerful tool. The key advantage of BO lies in the selection strategy of the next points to evaluate that balances the trade-off between exploration of unknown regions vs. exploitation of the best-performing ones. A good selection strategy allows BO to achieve great results in a small number of algorithm iterations and function evaluations. In each iteration of the BO algorithm, the first step is to train a statistical model on all available evaluated points, called *surrogate model*. Typically a Gaussian process is used to model the unknown objective function f and provide the model's prediction and uncertainty. The second step of the algorithm is the optimization of an *acquisition function* on the surrogate model that selects a point to evaluate next. The most popular acquisition functions for single-objective optimization problems include *expected improvement* (EI) [33, 20], *probability of improvement* (PI) [24], and *upper confidence bound* (UCB) [43]. In the case of multi-objective optimization, a surrogate model is usually trained for each objective independently,

and an acquisition function is adapted to overlook multiple models. Examples of such acquisition function are *expected hypervolume improvement* (EHI) [12] and *predictive entropy search* (PES) [17].

4 Proposed method

In this section, we present our multi-objective Bayesian optimization approach that proposes batches of samples to evaluate in each iteration.

Let $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$ be a vector of m black-box objectives of form $f_j : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \mathbb{R}^d$. The input to our method is a small set of initial samples $X_0 \subset \mathcal{X}$, usually drawn by *Latin Hypercube Sampling* (LHS) [31], and the corresponding evaluations $Y_0 = \{\mathbf{f}(\mathbf{x}_i), \forall \mathbf{x}_i \in X_0\} \subset \mathbb{R}^m$. Our approach consists of three main components: building a surrogate model G_j for each objective f_j (Section 4.1), approximation of the Pareto set $\mathcal{P}_s$ and Pareto front $\mathcal{P}_f$ (Section 4.2), and finally a selection of next set of samples to evaluate (Section 4.3). We iterate over these three components to efficiently improve the Pareto-optimal solution. The approach is summarized in Algorithm 1.

Algorithm 1 DGEMO

Inputs: Design space $\mathcal{X}$; black-box objectives $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_m(\mathbf{x}))$; number of iterations n; number of initial samples k; batch size b.
Output: Pareto set $\mathcal{P}_s$ and Pareto front $\mathcal{P}_f$.
$X_0 \leftarrow \{\mathbf{x}_1, \dots, \mathbf{x}_k\}, Y_0 \leftarrow \{\mathbf{f}(\mathbf{x}_1), \dots, \mathbf{f}(\mathbf{x}_k)\}$ // initial samples drawn from LHS
for $i \leftarrow 0$ **to** n **do**
  Train surrogate models $G_j^{(i)}$ on X_i, Y_i for each objective f_j, $j \in \{1, \dots, m\}$
  Define acquisition function $\tilde{f}_j^{(i)}$ from each $G_j^{(i)}$, $\tilde{\mathbf{f}}^{(i)}(\mathbf{x}) \leftarrow (\tilde{f}_1^{(i)}(\mathbf{x}), \dots, \tilde{f}_m^{(i)}(\mathbf{x}))$
  Approximate Pareto set $\mathcal{P}_s^{(i)}$ and Pareto front $\mathcal{P}_f^{(i)}$ over $\tilde{\mathbf{f}}^{(i)}$
  Split points from $\mathcal{P}_s^{(i)}$ into diversity regions $\mathcal{D}_1^{(i)}, \dots, \mathcal{D}_r^{(i)}$
  Select points $\mathbf{x}_1^{(i)}, \dots, \mathbf{x}_b^{(i)}$ to evaluate from $\mathcal{D}_1^{(i)}, \dots, \mathcal{D}_r^{(i)}$
  Evaluate and update $Y_{i+1} \leftarrow Y_i \cup \{\mathbf{f}^{(i)}(\mathbf{x}_1^{(i)}), \dots, \mathbf{f}^{(i)}(\mathbf{x}_b^{(i)})\}$, $X_{i+1} \leftarrow X_i \cup \{\mathbf{x}_1^{(i)}, \dots, \mathbf{x}_b^{(i)}\}$
Compute Pareto front $\mathcal{P}_f$ from points in Y_n and corresponding Pareto set $\mathcal{P}_s$

4.1 Surrogate model

We use Gaussian process (GP) as a surrogate model to model each objective independently. First, the prior of GP model is characterized by the *mean function* $m : \mathcal{X} \to \mathbb{R}$ and the *kernel function* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. For a given input variable $\mathbf{x}$, the prior distribution of GP can be stated as $f(\mathbf{x}) \sim \mathcal{N}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}))$. GP prior can be used to incorporate any prior beliefs about the objective functions if available, without depending on the input data, by choosing different mean function and kernel function. Without the loss of generality, we use mean function $m(\mathbf{x}) = 0$ and experiment with *Matern kernel functions* [36] that can capture a large variety of function properties. Secondly, we train a GP posterior to refine the prior model by maximizing the log marginal likelihood $\log p(\mathbf{y}|\mathbf{x}, \theta)$ on the available dataset $\{X, Y\}$, where θ denotes the parameters of the kernel function (e.g. $\theta = l$ for Matern kernel). Finally, the posterior distribution of GP is given in the form $f(\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}), \Sigma(\mathbf{x}))$, where the mean function is $\mu(\mathbf{x}) = m(\mathbf{x}) + \boldsymbol{k}\boldsymbol{K}^{-1}Y$, and covariance function $\Sigma(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \boldsymbol{k}\boldsymbol{K}^{-1}\boldsymbol{k}^T$, with $\boldsymbol{k} = k(\mathbf{x}, X)$ and $\boldsymbol{K} = k(X, X)$. To better explore the Pareto front based on this surrogate model, our Pareto front approximation algorithm (Section 4.2) makes use of the Jacobian and Hessian of the GP prediction $\mu(\mathbf{x}), \Sigma(\mathbf{x})$ w.r.t. $\mathbf{x}$. We provide the corresponding derivations in Appendix A.

4.2 Pareto front approximation

After obtaining a surrogate model G_j for each objective function f_j, we simply use the mean function of GP posterior as an acquisition function $\tilde{f}_j = \mu_j$. The next step of our algorithm then is to compute the Pareto front over all $\tilde{f}_j$. A core contribution to the efficacy of our algorithm comes from the Pareto front approximation approach, based on the method proposed in [39]. In typical MOO methods, discovering an individual point that minimizes the set of functions $\{f_1(\mathbf{x}), \dots, f_m(\mathbf{x})\}$, s.t. $\mathbf{x} \in \mathcal{X}$ can be challenging. However, once a single solution is discovered, locally searching around this point is easier. By solving a dual problem based on *KKT conditions* [19], the approach of [39] leverages this fact and derives a first-order approximation of the Pareto front. They efficiently discover large

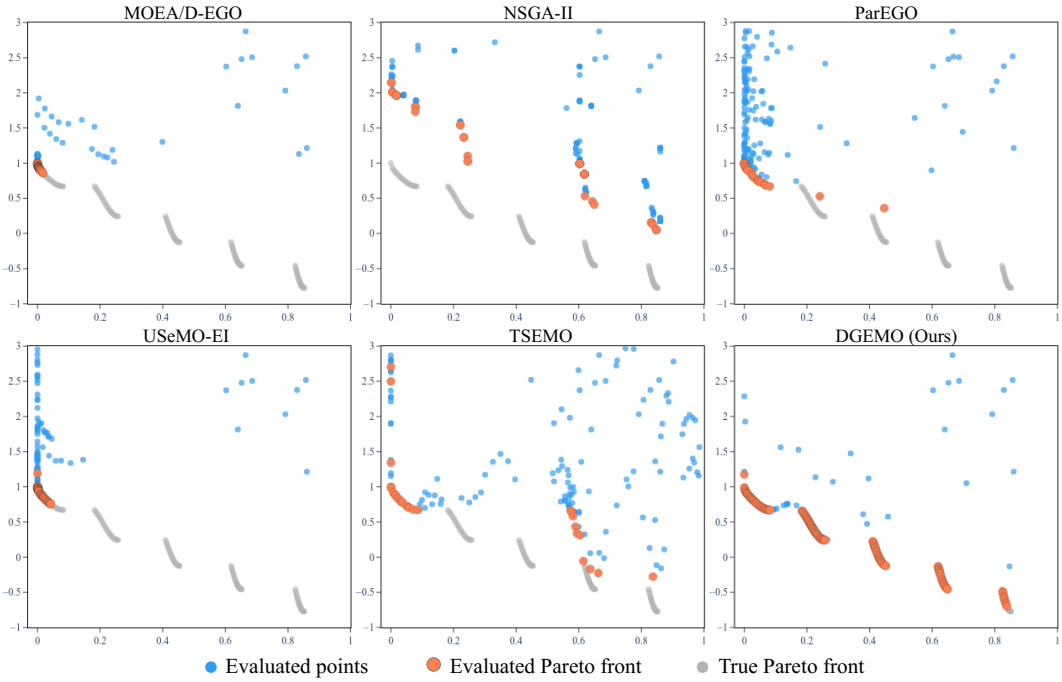


Figure 1: Comparison of evaluated points (blue and orange) performed by different algorithms on standard ZDT3 problem. Note that DGEMO quickly discovers all regions of interest and focuses most evaluations in these regions, covering almost the entire ground truth Pareto front (gray) in only 20 iterations of the algorithm with batch size 10.

piecewise continuous regions of the Pareto front, rather than a sparse set of points obtained by random mutations from popular evolutionary algorithms. For more details, we refer the reader to [39].

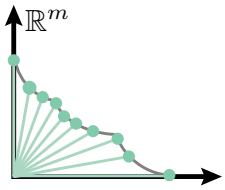
In summary, the Pareto front approximation implements an iterative procedure consisting of three main steps. First, a *stochastic sampling* scheme is used to generate a set of random samples $\mathbf{x}_i \in \mathcal{X}$ perturbed from the best samples found so far, to avoid local minima and balance between exploration and exploitation of the design space neighborhoods. The second step implements a *local optimization* procedure that drives each sample $\mathbf{x}_i$ towards a local Pareto-optimal solution $\mathbf{x}_i^*$ with L-BFGS solver [27]. Different optimization directions are explored to cover diverse regions of the Pareto front. Finally, a *first-order approximation* of the Pareto front is extracted around $\mathbf{x}_i^*$, resulting in a dense set of solutions located on a $(m-1)$-dimensional manifold that approximates a continuous region on the Pareto front. The last step delivers the key advantage of this approach: once a single point in a Pareto set is found, it can be used to efficiently discover large Pareto-optimal regions in its neighborhood. In our experiments, we found that only 10 iterations of this procedure are enough to produce a high-quality approximation. One example that presents the resulting Pareto front approximation of a benchmark problem is shown in Figure 1. Note that due to the limited budget of evaluations, other algorithms waste most of the samples trying to expand a single region and struggle to escape the local minima, while ours manages to quickly discover all regions of the true Pareto front.

4.3 Batch selection strategy

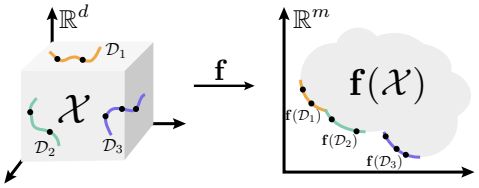
The last part of our algorithm proposes a novel strategy for selecting a batch of samples to evaluate in the next iteration. We first present the approach for defining *diversity regions*; we then describe the selection strategy and corresponding algorithm.

Diversity regions Starting from the Pareto front approximation, the goal is to group the optimal points based on their design properties and performance, resulting in several *diversity regions*. Standard MOO methods, such as NSGA-II, tend to output a Pareto front as a sparse set of points uniformly scattered around performance space, without considering the design space diversity. On the contrary, in our method, the Pareto front representation obtained from Section 4.2 has compact regions and better space coverage. From this representation, we can easily trace the neighboring points and their

correlation in both design and performance space. To solve the grouping problem, we rely on a data structure called *performance buffer*, implemented by [39]. The performance buffer sorts out the best performing points from the Pareto front approximation by storing them into an $(m-1)$-dimensional array discretized with hyperspherical coordinates. Assuming that all objectives are positive, any point that lies on the Pareto front will intersect a positive ray traced from the origin (see inset). The best performing points are selected as the points closest to the origin and sorted based on their hyperspherical coordinates. The buffer stores the performance and design space locations of each selected point, as well as the corresponding linear subspace of the design space extracted with the first-order approximation step. This way, the buffer keeps both the optimality information from the performance space and the closeness information from the design space. This data structure is compatible with a graph-cut method, which then extracts a sparse subset of optimal points grouped into k linear subspaces $\mathcal{D}_1, \dots \mathcal{D}_k$. The points are grouped based on the quality of their performance and their neighborhoods in the design space. We use these linear subspaces as our diversity regions.



Selection strategy Our strategy involves two criteria: diversity measure and hypervolume improvement. Our diversity measure incorporates the knowledge from both design and performance space, and aims to evenly distribute selected samples among diversity regions (see inset). This measure then enforces the exploration of different regions of the Pareto front, while also considering diversity in the design space. It is especially important in the initial iterations when the uncertainty in the model is high and hypervolume improvement is often inaccurate. It further prevents the optimization from falling into local minima and overly exploiting one high-performing area, while neglecting other potentially promising regions.



Our selection strategy aims to maximize the hypervolume improvement while enforcing the samples to be taken from all diverse regions as uniformly as possible. These requirements can be written in the following form:

$$\arg\max_{X_B} \mathrm{HVI}(Y_B, \mathcal{P}_f) \quad \text{s.t.} \max_{1 \le i \le |\mathcal{D}|} \delta_i(X_B) - \min_{1 \le i \le |\mathcal{D}|} \delta_i(X_B) \le 1 \tag{3}$$

where $X_B = \{\mathbf{x}_1, \dots, \mathbf{x}_b\}$ is a set of b samples in a batch, $Y_B = \{\tilde{\mathbf{f}}(\mathbf{x}_1), \dots, \tilde{\mathbf{f}}(\mathbf{x}_b)\}$, $\mathcal{P}_f$ is the current Pareto front, functions $\delta_i(.)$ are defined for each region $\mathcal{D}_i \in \mathcal{D}$ as a number of elements $\mathbf{x}_j$ from X_B that belong to the region $\mathcal{D}_i$.

Selection algorithm The optimization problem defined in Equation 3 can be solved combinatorially. Nevertheless, it would be extremely computationally expensive. Instead, we implement a greedy approach. We select a point $\mathbf{x}_1$ with the largest hypervolume improvement, add $\mathbf{x}_1$ to the batch, and add $\tilde{\mathbf{f}}(\mathbf{x}_1)$ to the current Pareto front $\mathcal{P}_f$. We then search for the next point with the largest hypervolume improvement, which does not belong to the same region as $\mathbf{x}_1$. We repeat this process while avoiding all the regions from which the points were already selected until all regions are covered. If more points are still needed for the batch, we reset the counter of covered regions and repeat the same process. This way we obtain evenly distributed points around regions, while also aiming to maximize the hypervolume improvement. The complete algorithm is presented in Algorithm 2.

5 Experiments and results

We now compare our algorithm to the relevant state-of-the-art methods on both synthetic test functions and real-world benchmark problems.

Benchmark problems First, we conduct experiments on 13 synthetic multi-objective test functions including ZDT1-3 [54], DTLZ1-6 [10], OKA1-2 [34], VLMOP2-3 [48], which are widely used in previous literature. The experiments include problems with 2 and 3 objectives, and the number of design variables varying from 2 to 7. Second, we adopt 7 real-world engineering design problems

Algorithm 2 Batch Selection Algorithm

Inputs: Current Pareto front $\mathcal{P}_f$; batch size b; candidate points split into regions $\mathcal{D}_1, \ldots, \mathcal{D}_r$; surrogate objectives $\tilde{\mathbf{f}}(\mathbf{x}) = (\tilde{f}_1(\mathbf{x}), ..., \tilde{f}_m(\mathbf{x}))$.
Output: Batch of selected samples X_B.
$S \leftarrow \{1, 2, ..., r\}$ // set of available regions to choose
for $i \leftarrow 1$ **to** b **do**
  $h_{max} \leftarrow -\infty, s_{max} \leftarrow 0, \mathbf{x}_{max} \leftarrow null$ // record the sample with maximal HVI
  for each $s \in S$ **do**
    for each $\mathbf{x} \in \mathcal{D}_s$ **do**
      $h \leftarrow \mathrm{HVI}(\tilde{\mathbf{f}}(\mathbf{x}), \mathcal{P}_f)$
      if $h > h_{max}$ **then** $h_{max} \leftarrow h, s_{max} \leftarrow s, \mathbf{x}_{max} \leftarrow \mathbf{x}$
  $X_B \leftarrow X_B \cup \{\mathbf{x}_{max}\}$, $\mathcal{P}_f \leftarrow \mathcal{P}_f \cup \{\tilde{\mathbf{f}}(\mathbf{x}_{max})\}$ // aggregate solutions
  $\mathcal{D}_{s_{max}} \leftarrow \mathcal{D}_{s_{max}} - \{\mathbf{x}_{max}\}$, $S \leftarrow S - \{s_{max}\}$
  if $S = \emptyset$ **then** $S \leftarrow \{1, 2, ..., r\}$ // reset the counter of visited regions

presented in RE problem suite [46], which are: four bar truss design, reinforced concrete beam design, hatch cover design, welded beam design, disc brake design, gear train design, and rocket injector design. We use names RE1-7 to refer to these problems in plots and further text. More detailed description of the problems can be found in Appendix B.

Algorithms and implementation We compare our algorithm to several baseline algorithms described in Section 2: NSGA-II [9], ParEGO [22], MOEA/D-EGO [52], TSEMO [6], and USeMO-EI [2]. We extend ParEGO and USeMO-EI from their original sequential selection to batch selection for batch evaluation scenario; see Appendix B for details of the extension. We implement and compare all algorithms in our Python codebase, built upon pymoo [5], a state-of-the-art Python framework for multi-objective optimization. Our code will be released open-source with reproducibility guarantee. The performance of our implementation is of high-quality and even surpasses the original authors' implementation on some baseline algorithms, see Appendix B for comparison.

Comparison metric To compare the performance of different algorithms, we mainly use *hypervolume indicator* – the most popular comparison metric for multi-objective optimization problems [37]. This indicator measures the quality of the set of discovered solutions. We monitor the increase of the hypervolume indicator (Eq. 1) over the number of evaluations performed. For comparison fairness, we use the same reference point and initial set of samples for every algorithm. See Appendix B for more details on initial samples and reference points used for each problem. Results using other comparison metrics are available in Appendix D.

5.1 Results and discussion

The selected test problems serve to validate the quality of the method when dealing with functions that are concave, convex, disconnected, multimodal, and of diverse design and performance space complexity. Our approach shows consistent top performance, compared to other algorithms that oscillate on different types of problems. DGEMO also proves to be particularly good on the synthetic functions mimicking the real-world data (OKA, VLMOP) and the actual real-world problems (RE). Figure 2 presents the comparison of the hypervolume indicator w.r.t. evaluation number. For every algorithm, we run every experiment with 10 different random seeds and the same 50 initial samples. In these experiments, we use a batch of 10 samples in each iteration and ran 20 iterations in total. All hyperparameters used are presented in Appendix B. The analysis on computational complexity can be found in Appendix C. We also conduct experiments using different batch sizes, including 1, 2, 4, 5 and 20, and our algorithm still consistently outperforms others. See Appendix E for more results. The Pareto front approximation approach presented in Section 4.2 enables our method to work well even with the small batch size (including a sequential case with batch size 1). However, while other methods are unstable and can vary the performance quality when the batch size increases, our method always exhibits superior and stable results thanks to the diversity-guided selection strategy.

In experimental design problems, aside from searching for an optimal set of solutions, it is often desirable to obtain a reliable prediction of the performance of untested samples. Hence, the quality of the surrogate model and the Pareto front approximation can be an important criterion in choosing

the algorithm to work with. In Table 1 we show the average prediction error of all evaluated points produced with our approach and other MOBO baseline algorithms (NSGA-II does not have a surrogate model so comparison is infeasible). The prediction error is calculated by $\|\mathbf{f}(\mathbf{x}) - \tilde{\mathbf{f}}(\mathbf{x})\|$ as the Euclidean distance between the surrogate prediction and the actual performance. It turns out that besides the superior performance of the hypervolume metric, DGEMO also has the lowest prediction error among the majority of the problems.

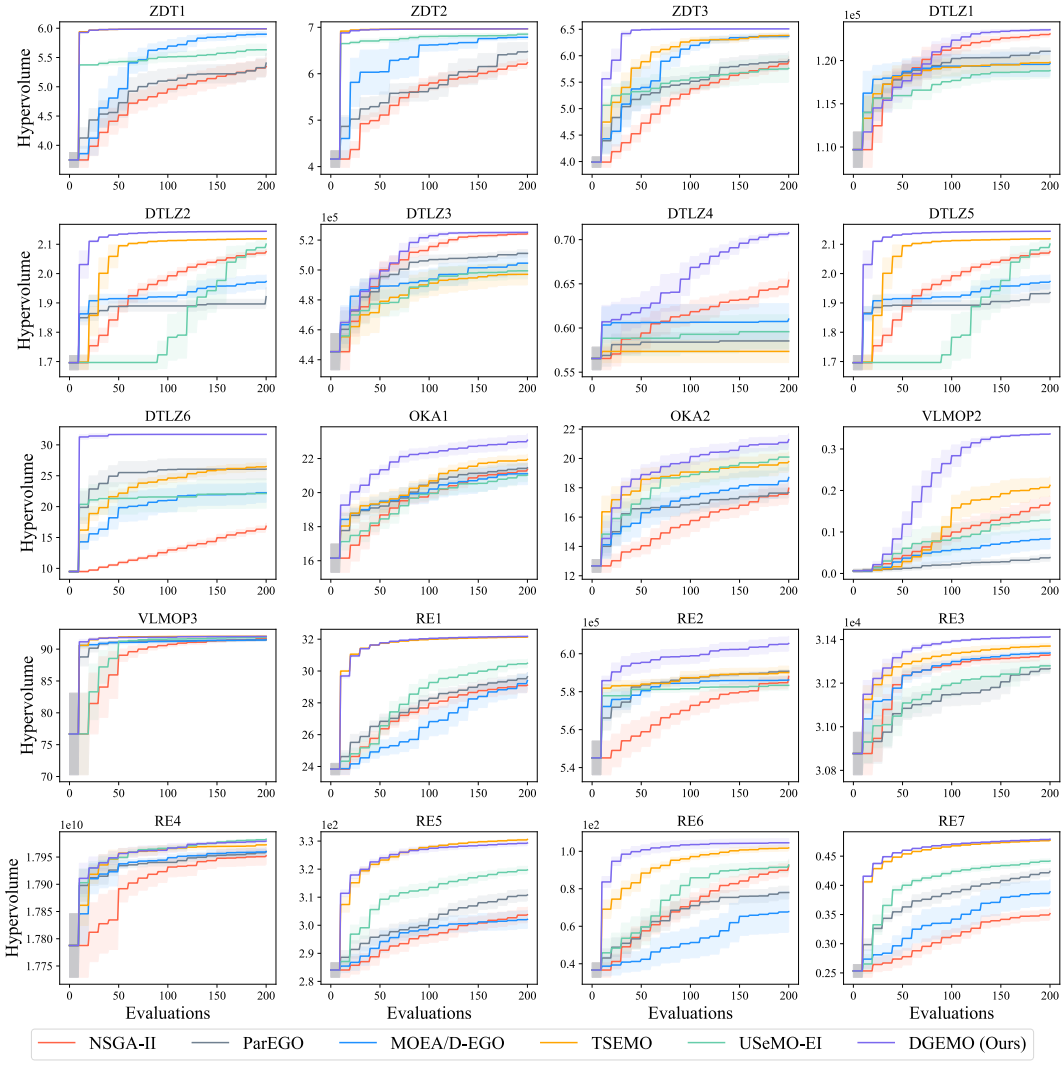


Figure 2: Comparison of different algorithms including our DGEMO on synthetic test functions and real-world problems. The hypervolume indicator is shown w.r.t. the number of function evaluations. Every experiment has batch size 10 and 20 algorithm iterations in total. The curve is averaged over 10 different random seeds and the variance is shown as a shaded region.

Given a trained surrogate model, unlike the others, our approach makes use of the Jacobian and Hessian information of this model in Pareto front approximation. We observe that DGEMO needs fewer iterations to stabilize the prediction, and greatly improves the hypervolume quality already in the first few iterations. Experiments that have a budget of less than 20 iterations would highly benefit from this property of our approach. In addition, DGEMO explores and provides additional information on diverse regions of Pareto-optimal solutions based on their properties and performance. Examining these regions individually may offer more comprehensive understanding of the problem and a wider range of solutions. One such example is presented in Appendix F, where we analyze and discuss the discovered regions on the rocket injector design problem (RE7).

Table 1: Averaged surrogate prediction error of points proposed by different algorithms, including DGEMO, on all benchmark problems. The scale of the error varies across different problems due to the problem definition, as some objectives have extremely high but valid values.

Method	ZDT1	ZDT2	ZDT3	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	OKA1
ParEGO	7.87	2.29	34.9	1.76e+04	6.95	3.83e+04	**30.1**	6.93	304	679
MOEA/D-EGO	76	8.11	106	2.09e+04	5.75	4.09e+04	124	5.75	6.37e+03	3.63e+03
TSEMO	65.9	5.48	331	1.78e+05	41.6	4.12e+05	607	41.6	1.43e+03	544
USeMO-EI	352	1.39	420	2.25e+04	22.3	5.54e+04	92.2	22.3	3.08e+04	1.55e+03
DGEMO (Ours)	**5.25**	**0.933**	**11.6**	**8.24e+03**	**1.59**	**1.5e+04**	71.6	**1.59**	**275**	**233**

Method	OKA2	VLMOP2	VLMOP3	RE1	RE2	RE3	RE4	RE5	RE6	RE7
ParEGO	**58.8**	**5.4**	61.6	5.93	2.86e+16	98.6	**1.95e+08**	575	292	0.597
MOEA/D-EGO	297	9.22	69.4	10.2	1.23e+15	264	1.19e+09	284	1.64e+03	1.26
TSEMO	641	31.9	284	153	1.19e+17	1.32e+03	1.47e+10	229	1.33e+03	0.734
USeMO-EI	1.21e+03	17.5	115	16.4	7.74e+15	386	6.72e+09	**114**	976	1.14
DGEMO (Ours)	140	6.26	**21**	**3.09**	**3.25e+11**	**92.7**	2.13e+08	1.22e+03	**288**	**0.343**

6 Conclusion and future work

We introduced a novel multi-objective Bayesian optimization method that allows evaluation of batches of samples in each iteration. The selection of batch samples takes into consideration the diversity of samples in both design and performance space, making it suitable for physical experiments and many real-world problems. The key advantage of our approach lies in the Pareto front approximation algorithm and the batch selection strategy that are, to the best of our knowledge, different from any previously published work. We performed extensive tests on both synthetic test functions and real-world problems. We found that our approach significantly outperforms other methods in most of the test problems, and in others performs equally well as the best-scoring state-of-the-art algorithm.

While Gaussian processes are good at handling a certain amount of noise in the data, examining the effects of noise on the system in more detail is an interesting avenue for future research. Currently, our method works with continuous design variables. Extending the method to support discrete variables would be a valuable addition for some real-world applications. Integrating DGEMO to an automated experimental lab setup to control the development and discovery of best performing experiments is an exciting new research direction.

Broader Impact

Bayesian optimization has successfully been used for a wide range of applications, including parameter tuning [4, 42], recommender systems and advertising [23, 40, 7], robotics [28, 30], resource allocation [18, 49], environmental monitoring [29], clinical drug trials [47, 50], experimental design [44, 14]. This list is not exhaustive; many more examples can be found and new problems could exploit Bayesian optimization methods. We present a novel approach for multi-objective Bayesian optimization that aims to enable parallelized evaluations and increased time efficiency of the optimization process. Our approach is not limited to specific applications. In general, any problem that requires sample-efficient optimization of multiple black-box functions that are expensive to evaluate could benefit from using our approach. The proposed method exhibits improved performance over the current state-of-the-art methods and offers a new perspective on the diversity of explored solutions. In particular, the extracted diversity regions of Pareto-optimal solutions may provide important material for further investigation of the problem. Examining the shared properties among samples in each region could deliver a better understanding of the interaction between design variables (base ingredients) and their contribution to optimal performance. Our method may be especially useful for applications in areas such as materials science, chemistry, and pharmaceutical industry, where the experiments are long (e.g., days or weeks), but can easily be carried out in parallel.

An important benefit of our work may be found in sustainability. The method specifically cares about preserving the resources used in experiments, such as energy consumption, chemical ingredients, and time. The reduced number of evaluations may significantly lower the testing waste. We hope that our approach will initiate more research in this direction and attract more attention to saving resources as much as possible alongside with rapid technological advances.

We plan to release an open-source codebase with a user interface for users with limited prior exposure to machine learning or multi-objective optimization algorithms. We hope that this codebase will perpetuate research in various scientific disciplines and enable the growth of interdisciplinary work.

On the negative side, we cannot guarantee that our approach will be used solely for a good cause with no negative use cases. Our contribution is mainly improving the efficiency of the experiments and saving more resources; however, we cannot prevent the abuse of this technology.

The failure in our system would mean an inability to improve the set of Pareto-optimal solutions received from the initial set of samples. This inability may be caused by inadequate input data, such as too limited design space, or evaluations with intractable oscillations in noise. The failure can be detected after only a few iterations and experiments could be terminated. At this point, we cannot think of any further disadvantages of using our approach.

We see further opportunities in the development of automated experimental design systems in research and technology. Our approach can be integrated as a *"brain"* of an autonomous production and testing system, to drive the design of experiments and evaluations without human supervision and hand-picking of the design samples to evaluate. The automation process may cause some job loss of repetitive tasks. However, it may also result in increased productivity of researchers and accelerate the development of important products, such as drugs, chemicals, and new materials. It may provide new job opportunities for algorithm engineers and data scientists to build more intelligent optimization workflows according to the specific needs of different problems.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their helpful comments in revising the paper. This work is supported by National Science Foundation (grant No. 1815372). M. K. Luković would like to acknowledge support from the Schmidt Science Fellowship.

References

[1] Peter M Attia, Aditya Grover, Norman Jin, Kristen A Severson, Todor M Markov, Yang-Hung Liao, Michael H Chen, Bryan Cheong, Nicholas Perkins, Zi Yang, et al. Closed-loop optimization of fast-charging protocols for batteries with machine learning. *Nature*, 578(7795):397–402, 2020.

[2] Syrine Belakaria and Aryan Deshwal. Uncertainty-aware search framework for multi-objective bayesian optimization. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[3] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 7823–7833, 2019.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2546–2554, Red Hook, NY, USA, 2011. Curran Associates Inc.

[5] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, pages 1–1, 2020.

[6] Eric Bradford, Artur M Schweidtmann, and Alexei Lapkin. Efficient multiobjective optimization employing gaussian processes, spectral sampling and a genetic algorithm. *Journal of global optimization*, 71(2):407–438, 2018.

[7] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2249–2257. Curran Associates, Inc., 2011.

[8] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[10] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*, pages 105–145. Springer, 2005.

[11] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15:3873–3923, 2014.

[12] Michael Emmerich. The computation of the expected improvement in dominated hypervolume of pareto front approximations.

[13] Adrian Gambier and Essameddin Badreddin. Multi-objective optimal control: An overview. In *2007 IEEE International Conference on Control Applications*, pages 170–175. IEEE, 2007.

[14] R. Garnett, M. A. Osborne, and S. J. Roberts. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, page 209–219, New York, NY, USA, 2010. Association for Computing Machinery.

[15] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In *Artificial intelligence and statistics*, pages 648–657, 2016.

[16] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design. *arXiv preprint arXiv:1709.05501*, 2017.

[17] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in neural information processing systems*, pages 918–926, 2014.

[18] R. Hickish, D.I. Fletcher, and R.F. Harrison. Investigating bayesian optimization for rail network optimization. *International Journal of Rail Transportation*, October 2019. © 2019 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

[19] Claus Hillermeier. Generalized homotopy approach to multiobjective optimization. *Journal of Optimization Theory and Applications*, 110:557–583, 2001.

[20] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[21] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 4206–4214, 2016.

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

[23] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. Controlled experiments on the web: Survey and practical guide. *Data Min. Knowl. Discov.*, 18(1):140–181, February 2009.

[24] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. 1964.

[25] Xi Lin, Qingfu Zhang, and Sam Kwong. An efficient batch expensive multi-objective evolutionary algorithm based on decomposition. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1343–1349. IEEE, 2017.

[26] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qingfu Zhang, and Sam Kwong. A batched scalable multi-objective bayesian optimization algorithm. *arXiv preprint arXiv:1811.01323*, 2018.

[27] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[28] Daniel Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 944–949, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[29] Roman Marchant and Fabio Ramos. Bayesian optimisation for intelligent environmental monitoring. pages 2242–2249, 10 2012.

[30] Ruben Martinez-Cantin, Nando Freitas, Eric Brochu, Jose Castellanos, and Arnaud Doucet. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Auton. Robots*, 27:93–103, 08 2009.

[31] Michael D McKay, Richard J Beckman, and William J Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[32] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[33] Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

[34] Tatsuya Okabe, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. On test functions for evolutionary multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 792–802. Springer, 2004.

[35] Victor Picheny. Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, 2015.

[36] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[37] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)*, pages 1–11. IEEE, 2015.

[38] Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. In *Advances in Neural Information Processing Systems*, pages 1583–1591, 2014.

[39] Adriana Schulz, Harrison Wang, Eitan Grinspun, Justin Solomon, and Wojciech Matusik. Interactive exploration of design trade-offs. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[40] Steven L. Scott. A modern bayesian look at the multi-armed bandit. *Appl. Stoch. Model. Bus. Ind.*, 26(6):639–658, November 2010.

[41] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[42] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[43] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 1015–1022, 2010.

[44] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1015–1022, Madison, WI, USA, 2010. Omnipress.

[45] Raj Subbu, Piero P Bonissone, Neil Eklund, Srinivas Bollapragada, and Kete Chalermkraivuth. Multiobjective financial portfolio design: A hybrid evolutionary approach. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1722–1729. IEEE, 2005.

[46] Ryoji Tanabe and Hisao Ishibuchi. An easy-to-use real-world multi-objective optimization problem suite. *Applied Soft Computing*, page 106078, 2020.

[47] William R Thompson. On the Likelihood That one Unknown Probability Exceeds Another in View of the Evidence of two Samples. *Biometrika*, 25(3-4):285–294, 12 1933.

[48] David A Van Veldhuizen and Gary B Lamont. Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM symposium on Applied computing*, pages 351–357, 1999.

[49] J. Wu, W.Y. Zhang, S. Zhang, Y.N. Liu, and X.H. Meng. A matrix-based bayesian approach for manufacturing resource allocation planning in supply chain management. *International Journal of Production Research*, 51(5):1451–1463, 2013.

[50] Zhenning Yu, Viswanathan Ramakrishnan, and Caitlyn Meinzer. Simulation optimization for bayesian multi-arm multi-stage clinical trial with binary endpoints. *Journal of Biopharmaceutical Statistics*, 29:1–12, 02 2019.

[51] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[52] Qingfu Zhang, Wudong Liu, Edward Tsang, and Botond Virginas. Expensive multiobjective optimization by moea/d with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2009.

[53] Yichi Zhang, Daniel W Apley, and Wei Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific Reports*, 10(1):1–13, 2020.

[54] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

[55] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[56] Marcela Zuluaga, Guillaume Sergent, Andreas Krause, and Markus Püschel. Active learning for multi-objective optimization. In *International Conference on Machine Learning*, pages 462–470, 2013.