Multi-Objective Bayesian Optimization with Pareto Set Learning

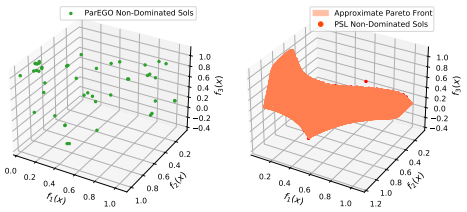
Anonymous Author(s)
Affiliation
Address
email

Abstract

Expensive multi-objective optimization problems can be found in many real-world applications, where their objective function evaluations involve expensive computation and/or physical experiments. It is desirable to obtain an approximate Pareto front with a small evaluation budget. Multi-objective Bayesian optimization (MOBO) has been widely used for finding a finite set of Pareto optimal solutions to this problem. However, it is well-known that the whole Pareto set could have infinite optimal solutions. The structural properties of the Pareto set are not well utilized in existing MOBO methods, and the finite-set approximation may not contain the most preferred solution(s) for decision-makers. This paper develops a simple yet efficient Pareto set learning method to approximate the whole Pareto set for MOBO. We design a simple yet powerful acquisition search method with the learned Pareto set, which naturally supports batch evaluation. In addition, with our proposed model, decision-makers can readily explore any trade-off area in the approximate Pareto set for flexible decision-making. Experimental results on different synthetic and real-world problems demonstrate the effectiveness and efficiency of our proposed method.

1 Introduction

Many real-world applications involve optimizing multiple costly-to-evaluate and potentially competing objectives, such as finding strong yet ductile material [32], building a neural network with high accuracy and low latency [22], and improving the quality while minimizing total charge in particle accelerator tuning [61]. Very often, these objectives could conflict with each other and cannot be optimized simultaneously by a single solution. Instead, there is a set of solutions with different optimal trade-offs among the objectives, called the Pareto set. For a Pareto optimal solution, none of its objective values can be further improved without deteriorating others. In addition, the evaluation of each solution requires time-consuming computation or costly physical experiment, and thus a large number of evaluations are not acceptable. Different multi-objective Bayesian optimization (MOBO) algorithms [38, 42, 37], typically directly generalized from the single-objective Bayesian optimization (BO) [50, 33, 9, 63, 23], have been proposed to find a small set of approximate Pareto solutions with a limited evaluation budget.



(a) ParEGO (b) Pareto Set Learning

Figure 1: **Pareto Set Learning** can approximate the whole Pareto set. With our proposed model, decision-makers can easily explore any trade-off among the objectives to choose their most preferred solutions.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

However, the finite set approximation has undesirable drawbacks. For a nontrivial multi-objective optimization problem, the Pareto set is on a continuous manifold and has infinite solutions with different optimal trade-offs among the objectives [48]. This Pareto set structure could be helpful to better select candidate solutions for expensive evaluation, and hence accelerate the optimization process of MOBO. In addition, a small set of solutions may not contain the one(s) that precisely satisfy the decision-maker's preferences. Finding the most preferred trade-off solution(s) could require several rounds of communication with the decision-makers. These approaches would be extremely time-consuming and undesirable, especially when the optimization modeler and the decision-maker are not in the same team, which is common in real-world MOBO applications [47].

In this work, we propose a novel Pareto set learning (PSL) method to approximate the whole Pareto set for expensive multi-objective optimization problems. Our proposed method can accelerate the multi-objective Bayesian optimization process, and also provide decision-makers with more useful information to support flexible decision-making. We believe it is a principled way to deal with MOBO. This work has the following main contributions:

- We propose a novel set model to map any trade-off preference to its corresponding Pareto solution, along with a surrogate model-based method to approximate the whole Pareto set with a limited evaluation budget.
- We develop a lightweight yet powerful batch acquisition search method on the approximate Pareto set for efficient MOBO, which can outperform other MOBO approaches in both performance and runtime. We also demonstrate that the learned Pareto set can support flexible user-involved decision-making.
- We test our proposed method on both synthetic benchmarks and real-world application problems. The results fully validate the efficiency and usefulness of PSL.

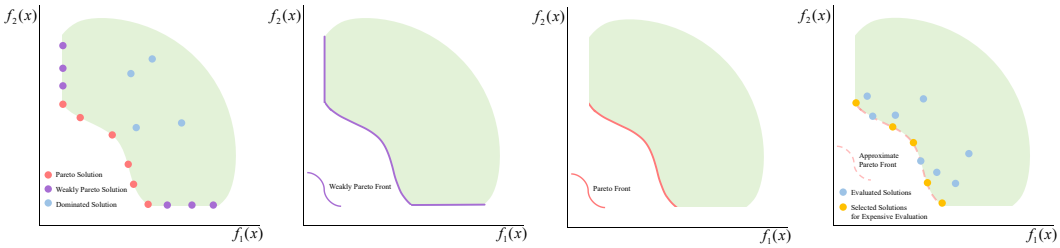
2 Related work

Bayesian Optimization. Surrogate model-based methods have been widely studied for expensive optimization [41, 34, 55, 65]. This method iteratively builds a surrogate model to approximate the black-box objective function, and uses an acquisition function to search optimal solution. Due to its importance in machine learning, much effort has been made on various design issues in Bayesian optimization, such as acquisition functions [67], high-dimensional optimization [77], batch evaluation [18], scalable optimization [66], and theoretical analysis [36]. Most of them are on single-objective optimization. We refer readers to Garnett [24] for a comprehensive introduction.

Multi-Objective Bayesian Optimization. MOBO extends single-objective Bayesian optimization for solving expensive multi-objective optimization problems. The scalarization-based algorithms, such as ParEGO [39] and TS-TCH [52] iteratively scalarize the multi-objective problem into single-objective ones with random preferences, and then apply single-objective BO to solve them. MOEA/D-EGO [82] extend the MOEA/D framework [80] to solve a set of surrogate scalarized subproblems simultaneously. SMS-EGO [57] and PAL [86, 87] generalize the upper confidence bound (UCB) to multi-objective optimization. Emmerich et al. [21] and Emmerich and Klinkenberg [20] propose the probability of improvement (PI) and expectation improvement (EI) for multi-objective hypervolume. Entropy search methods [27, 29, 31, 76] have also been explored in multi-objective optimization [28, 4, 70]. Bradford et al. [8] and Belakaria et al. [5] consider Thompson sampling and uncertainty maximization for multi-objective optimization, respectively. Different algorithms can also be efficiently combined to achieve better overall performance [68].

The current works have investigated the diverse batch evaluation [45], efficient hypervolume improvement calculation [13], multiple noisy objectives [14], and decision criteria beyond Pareto optimality [47]. Some attempts have been made to incorporate the decision-maker's preference into MOBO [1, 52, 2]. They typically need decision-maker's preference before or during optimization, which could be unavailable in many real-world applications.

Structure Learning. In addition to the surrogate objective model, some attempts have been made to explore the problem structure for Bayesian optimization. Sener and Koltun [62] learn the geometric structure of the problem in an online manner to accelerate optimization. Wang et al. [75] and Zhao et al. [83] apply Monte Carlo tree search (MCTS) to divide the search space for efficient modeling



(a) (Weakly) Pareto Sols. (b) Weakly Pareto Front (c) Pareto Front (d) PSL for MOBO

Figure 2: **(Weakly) Pareto solutions and (Weakly) Pareto front. (a)** Examples of Pareto solutions, weakly Pareto solutions, and dominated solutions. The Pareto solutions are also weakly Pareto optimal. The weakly Pareto optimal but not Pareto optimal solutions (e.g., purple points) are dominated but not strictly dominated by at least one Pareto solution. **(b)** The weakly Pareto front $\boldsymbol{f}(\mathcal{M}_{\text{weak}})$ is the image of all weakly Pareto optimal solutions $\mathcal{M}_{\text{weak}}$ in the objective space. **(c)** The Pareto front $\boldsymbol{f}(\mathcal{M}_{\text{ps}})$ is the image of all Pareto optimal solution $\mathcal{M}_{\text{ps}}$ (Pareto set) in the objective space. **(d)** Our proposed method approximates the whole Pareto set and uses it to select solutions for expensive evaluation, which can improve the search efficiency of multi-objective Bayesian optimization.

and searching. Novel latent space modelings [26, 72] have been proposed for optimization problems with complex solution representations.

The idea to approximate the whole Pareto set manifold has been studied in the multi-objective optimization community [58, 30, 25]. Pirotta et al. [56] and Parisi et al. [53] have explored the Pareto manifold approximation method for multi-objective reinforcement learning. Apart from optimization, different approaches have been investigated to incorporate the preference information into deep neural networks for image style transfer [64, 19] and multi-task learning [46, 51, 43]. In this work, we propose to learn a set model which maps all valid trade-off preferences to the approximate Pareto set, and use it for efficient MOBO.

3 Expensive multi-objective optimization

We consider the following expensive continuous multi-objective optimization problem:

$$\min_{\boldsymbol{x}\in\mathcal{X}} \ \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})), \tag{1}$$

where $\boldsymbol{x}$ is a solution in the decision space $\mathcal{X} \subset \mathbb{R}^n$, $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^m$ is an m-dimensional vector-valued objective function, and the evaluation is expensive for all individual objectives $\{f_i(\boldsymbol{x})\}_{i=1}^m$. For a non-trivial problem, no single best solution can optimize all objectives at the same time, and we have to make a trade-off among them. We have the following definitions for multi-objective optimization:

Definition 1 (Pareto Dominance and Strict Dominance) *Let $\boldsymbol{x}^a, \boldsymbol{x}^b \in \mathcal{X}$, $\boldsymbol{x}^a$ is said to dominate $\boldsymbol{x}^b$, denoted as $\boldsymbol{x}^a \prec \boldsymbol{x}^b$, if and only if $f_i(\boldsymbol{x}^a) \leq f_i(\boldsymbol{x}^b), \forall i \in \{1, ..., m\}$ and $\exists j \in \{1, ..., m\}$ such that $f_j(\boldsymbol{x}^a) < f_j(\boldsymbol{x}^{(b)})$. In addition, $\boldsymbol{x}^a$ is said to strictly dominate $\boldsymbol{x}^b$ ($\boldsymbol{x}^a \prec_{strict} \boldsymbol{x}^b$), if and only if $f_i(\boldsymbol{x}^a) < f_i(\boldsymbol{x}^b), \forall i \in \{1, ..., m\}$.*

Definition 2 (Pareto Optimality) *A solution $\boldsymbol{x}^* \in \mathcal{X}$ is Pareto optimal if there is no $\hat{\boldsymbol{x}} \in \mathcal{X}$ such that $\hat{\boldsymbol{x}} \prec \boldsymbol{x}^*$. A solution $\boldsymbol{x}' \in \mathcal{X}$ is weakly Pareto optimal if there is no $\hat{\boldsymbol{x}} \in \mathcal{X}$ such that $\hat{\boldsymbol{x}} \prec_{strict} \boldsymbol{x}'$.*

Definition 3 (Pareto Set/Front) *The set of all Pareto optimal solutions $\mathcal{M}_{ps} \subseteq \mathcal{X}$ is called the Pareto set, and its image in the objective space $\boldsymbol{f}(\mathcal{M}_{ps}) = \{\boldsymbol{f}(\boldsymbol{x}) | \boldsymbol{x} \in \mathcal{M}_{ps}\}$ is called the Pareto front. Similarly, we can define the weakly Pareto set $\mathcal{M}_{weak}$ and weakly Pareto front $\boldsymbol{f}(\mathcal{M}_{weak})$.*

The strict dominance relation is stronger than the Pareto dominance since it requires strictly better values for all objectives. Therefore, the set of weakly Pareto optimal solutions $\mathcal{M}_{\text{weak}}$ (e.g., the solutions that are *not* strictly dominated) would be larger than $\mathcal{M}_{\text{ps}}$, and it is straightforward to see $\mathcal{M}_{\text{ps}} \subseteq \mathcal{M}_{\text{weak}}$. The illustration of (weakly) Pareto solution and Pareto front is shown in Figure 2.

Each Pareto solution $\boldsymbol{x} \in \mathcal{M}_{\text{ps}}$ represents a different optimal trade-off among the objectives for problem (1). Under mild conditions, the Pareto set $\mathcal{M}_{\text{ps}}$ and Pareto front $\boldsymbol{f}(\mathcal{M}_{\text{ps}})$ are both on $(m-1)$-

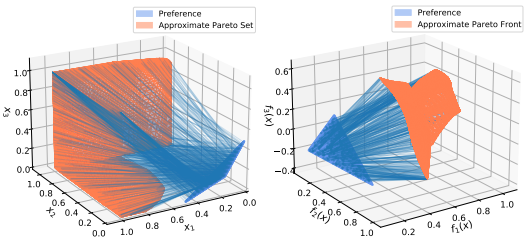
dimensional manifold in the decision space $\mathcal{X} \in \mathbb{R}^n$ and objective space $\mathbb{R}^m$, respectively [48, 30, 81]. The number of Pareto solutions could be infinite ($|\mathcal{M}_{\text{ps}}| = \infty$).

Pareto Set Learning. The current MOBO methods aim to find a small set of finite solutions $\mathcal{S} = \{\bar{\boldsymbol{x}}^{(1)}, \bar{\boldsymbol{x}}^{(2)}, \cdots, \bar{\boldsymbol{x}}^{(|\mathcal{S}|)}\}$ to approximate the Pareto set $\mathcal{M}_{\text{ps}}$. In addition to the evaluated solutions $\mathcal{S}$, our proposed PSL method also learns an estimated Pareto set $\mathcal{M}_{\text{psl}}$ with the predicted Pareto front $\hat{\boldsymbol{f}}(\mathcal{M}_{\text{psl}})$ to approximate the Pareto set $\mathcal{M}_{\text{ps}}$ and Pareto front $\boldsymbol{f}(\mathcal{M}_{\text{ps}})$ as shown in Figure 1(b). The whole approximate Pareto set can be easily explored by adjusting the trade-off preference as illustrated in Figure 3. With the learned Pareto set, we also develop an efficient batched solution selection approach for efficient MOBO.

4 Pareto Set Learning for MOBO

4.1 Pareto set model

As discussed in Section 3, the Pareto set can contain infinite solutions with different trade-offs which are non-dominated with each other. In other words, there is no clear order and rank among the Pareto solutions. A Pareto set model for MOBO should be powerful enough to approximate the whole Pareto set, and convenient enough to easily explore any trade-off solutions. In this work, we propose to build a model that maps any trade-off preferences to their corresponding Pareto solutions with scalarization.



(a) Preference to PS (b) Preference to PF

Figure 3: **Mapping from Preferences to Approximate Pareto Set/Front:** Our proposed PSL method learns the connection from a set of valid (infinite) trade-off preference $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum \lambda_i = 1\}$ to **(a)** the approximate Pareto set $\mathcal{M}_{\text{psl}}$ and hence **(b)** the corresponding predicted Pareto front $\hat{\boldsymbol{f}}(\mathcal{M}_{\text{psl}})$. The whole Pareto set (front) can be easily explored by adjusting the preference. We have resized and rotated the preference simplex for better visualization.

Scalarization. The scalarization method provides a natural connection from a set of well-organized preferences $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum \lambda_i = 1\}$ among the m optimization objective to the Pareto set $\mathcal{M}_{\text{ps}}$. The most simple and straightforward approach is the linear weight-sum scalarization:

$$\min_{\boldsymbol{x} \in \mathcal{X}} g_{\text{ws}}(\boldsymbol{x}|\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathcal{X}} \sum_{i=1}^m \lambda_i f_i(\boldsymbol{x}). \tag{2}$$

However, this method can only find the convex hull of Pareto front $\boldsymbol{f}(\mathcal{M}_{\text{ps}})$ [12]. In this work, we focus on the weighted Tchebycheff approach:

$$\min_{\boldsymbol{x} \in \mathcal{X}} g_{\text{tch}}(\boldsymbol{x}|\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathcal{X}} \max_{1 \le i \le m} \{\lambda_i (f_i(\boldsymbol{x}) - (z_i^* - \varepsilon))\}, \tag{3}$$

where $\boldsymbol{z}^* = (z_1^*, \cdots, z_m^*)$ is the ideal vector for the objective vector $\boldsymbol{f}(\boldsymbol{x})$ (lower-bound for minimization problem), $\varepsilon > 0$ is a small positive scalar, and $u_i = (z_i^* - \varepsilon)$ is called an (unachievable) utopia value for the i-th objective $f_i(\boldsymbol{x})$. This scalarization method has a promising property:

Theorem 1 (Choo and Atkins [11]). *A feasible solution $\boldsymbol{x} \in \mathcal{X}$ is weakly Pareto optimal if and only if there is a weight vector $\boldsymbol{\lambda} > 0$ such that $\boldsymbol{x}$ is an optimal solution of the problem (3).*

In other words, all Pareto solutions $\boldsymbol{x} \in \mathcal{M}_{\text{ps}}$ can be found by solving the Tchebycheff scalarized subproblem (3) with a specific (but unknown) trade-off preference $\boldsymbol{\lambda}$. We let $\mathcal{M}_{\text{tch}}$ be the solution set for problem (3) with all valid preferences Λ and have $\mathcal{M}_{\text{ps}} \subseteq \mathcal{M}_{\text{weak}} = \mathcal{M}_{\text{tch}}$. The weakly Pareto optimal but not Pareto optimal solutions (e.g., $\mathcal{M}_{\text{weak}} \setminus \mathcal{M}_{\text{ps}}$) are dominated (but not strictly dominated) by some Pareto solutions, and are usually not desirable for decision-making. They can be further avoided by the augmented Tchebycheff approach [69, 35]. In this work, we use the following scalarization:

$$g_{\text{tch_aug}}(\boldsymbol{x}|\boldsymbol{\lambda}) = \max_{1 \le i \le m} \{\lambda_i (f_i(\boldsymbol{x}) - (z_i^* - \varepsilon))\} + \rho \sum_{i=1}^m \lambda_i f_i(\boldsymbol{x}), \quad \forall \boldsymbol{\lambda} \in \Lambda,$$

where ρ is a sufficiently small positive scalar depends on the problem and current solution location. This form of augmentation has also been used in ParEGO [39]. With the augmentation term, the

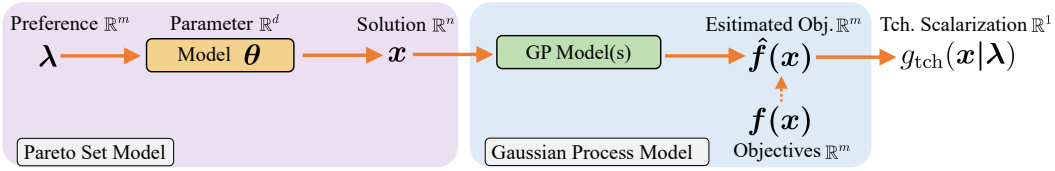


Figure 4: **Pareto Set Learning for Multi-Objective Bayesian Optimization:** **(a)** The Pareto set model learns a parameterized mapping from any valid preference $\boldsymbol{\lambda} \in \Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum \lambda_i = 1\}$ to its corresponding solution $\boldsymbol{x}(\boldsymbol{\lambda}) \in \mathbb{R}^n$. **(b)** We build independent Gaussian process models for each objective function. With these surrogate models, the set model can be efficiently trained to approximate the Pareto set. **(c)** In this work, we use the augmented Tchebycheff scalarization to connect each preference to its corresponding Pareto solution.

weakly dominated solutions will have larger scalarized values than the corresponding Pareto solutions in (4), and will ultimately be eliminated with the optimization process (e.g., $\mathcal{M}_{\text{tch_aug}} = \mathcal{M}_{\text{ps}}$). In this work, we simply set $\rho = 0.001$, dynamically update z_i^* as the current best value for each objective and let $\boldsymbol{\varepsilon} = 0.1|\boldsymbol{z}^*|$. This setting is robust for all problems we considered.

Set Model. With augmented Tchebycheff scalarization, we build a set model for mapping preferences to their solutions:

$$\boldsymbol{x}(\boldsymbol{\lambda}) = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}), \tag{4}$$

where $\boldsymbol{\lambda}$ is any valid preference in $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum \lambda_i = 1\}$, $\boldsymbol{x}(\boldsymbol{\lambda}) \in \mathcal{X}$ is its corresponding Pareto solution, and $h_{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ is the Pareto set model with parameter $\boldsymbol{\theta}$. Since all valid preferences $\boldsymbol{\lambda}$ has $(m-1)$ degree of freedom, the set of all generated solutions $\mathcal{M}_{\text{psl}} = \{\boldsymbol{x} = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}) | \boldsymbol{\lambda} \in \Lambda\}$ is on an $(m-1)$-dimensional manifold in $\mathcal{X} \in \mathbb{R}^n$. In other words, the set model maps the $(m-1)$-dimensional regular preference simplex Λ to the $(m-1)$-dimensional solution set $\mathcal{M}_{\text{psl}}$ with complicated structure.

We want to find the optimal parameters $\boldsymbol{\theta}^*$ such that the generated set $\mathcal{M}_{\text{psl}}$ matches the augmented Tchebycheff scalarization's set $\mathcal{M}_{\text{tch_aug}} = \{\boldsymbol{x}^*(\boldsymbol{\lambda}) | \boldsymbol{\lambda} \in \Lambda\}$, where

$$\boldsymbol{x}^*(\boldsymbol{\lambda}) = h_{\boldsymbol{\theta}^*}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{x} \in \mathcal{X}} g_{\text{tch_aug}}(\boldsymbol{x}|\boldsymbol{\lambda}), \forall \boldsymbol{\lambda} \in \Lambda. \tag{5}$$

An illustration of the learned mapping is shown in Figure 3. Once the connection is learned, we can efficiently navigate the whole approximate Pareto set/front by simply adjusting the preferences among objectives. In this work, we use an MLP neural network as the set learning model for all MOBO problems, which was shown to be powerful enough to capture complicated problem structure [62]. The model details can be found in Appendix.

4.2 Pareto Set Learning with Gaussian Process

Since the evaluation of $\boldsymbol{f}(\boldsymbol{x}(\boldsymbol{\lambda})) = \boldsymbol{f}(h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$ is expensive, we use surrogate models to learn the Pareto set model $h_{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ as shown in Figure 4. Our method is orthogonal to the choice of surrogate models, and we build independent Gaussian process model for each objective as in other MOBO approaches [45, 47].

Gaussian Process Model. A single-objective Gaussian process [59] has a prior distribution defined on the function space:

$$f(\boldsymbol{x}) \sim GP(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x})), \tag{6}$$

where $\mu : \mathcal{X} \to \mathbb{R}$ is the mean function and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the covariance kernel function. With n evaluated solutions $\{\boldsymbol{X}, \boldsymbol{y}\} = \{[\boldsymbol{x}^{(i)}], [f(\boldsymbol{x}^{(i)}] | i = 1, \ldots, n)\}$, the posterior distribution can be updated by maximizing the marginal likelihood based on the data. For a new solution $\boldsymbol{x}^{n+1}$, the posterior mean and variance are:

$$\hat{\mu}(\boldsymbol{x}^{(n+1)}) = \mu(\boldsymbol{x}^{(n+1)}) + \boldsymbol{k}^T \boldsymbol{K}^{-1} \boldsymbol{y}, \quad \hat{\sigma}^2(\boldsymbol{x}^{(n+1)}) = k(\boldsymbol{x}^{(n+1)}, \boldsymbol{x}^{(n+1)}) - \boldsymbol{k}^T \boldsymbol{K}^{-1} \boldsymbol{k}, \tag{7}$$

where $\boldsymbol{k} = k(\boldsymbol{x}, \boldsymbol{X})$ is the kernel vector and $\boldsymbol{K} = k(\boldsymbol{X}, \boldsymbol{X})$ is the kernel matrix (Matérn 5/2 kernel is used for all models). For m independent GP models, we let $\hat{\boldsymbol{\mu}}(\boldsymbol{x}) = [\hat{\mu}_1(\boldsymbol{x}), \cdots, \hat{\mu}_m(\boldsymbol{x})]$

and $\hat{\boldsymbol{\sigma}}^2(\boldsymbol{x}) = [\hat{\sigma}_1^2(\boldsymbol{x}), \cdots, \hat{\sigma}_m^2(\boldsymbol{x})]$ be the predicted mean and variance for the objective vector. Suppose we have a learned Pareto set $\mathcal{M}_{\text{psl}}$, the GP models give us both predicted value $\hat{\boldsymbol{\mu}}(\mathcal{M}_{\text{psl}}) = \{\hat{\boldsymbol{\mu}}(\boldsymbol{x}) | \boldsymbol{x} \in \mathcal{M}_{\text{psl}}\}$ and uncertainty $\hat{\boldsymbol{\sigma}}^2(\mathcal{M}_{\text{psl}}) = \{\hat{\boldsymbol{\sigma}}^2(\boldsymbol{x}) | \boldsymbol{x} \in \mathcal{M}_{\text{psl}}\}$ for the whole approximate Pareto set.

Pareto Set Learning. Now we propose an efficient algorithm to find the optimal parameter $\boldsymbol{\theta}^*$ for the Pareto set model $h_{\boldsymbol{\theta}}(\boldsymbol{\lambda})$. The optimal solution set $\mathcal{M}_{\text{tch_aug}}$ for augmented Tchebycheff scalarization (4) is unknown, hence we need to optimize all solutions generated by our model $\boldsymbol{x}(\boldsymbol{\lambda}) = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ with respect to their corresponding Tchebycheff scalarization subproblems for all valid preferences:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\lambda} \sim \Lambda} g_{\text{tch_aug}}(\boldsymbol{x} = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}) | \boldsymbol{\lambda}). \quad (8)$$

Algorithm 1 PSL with GP Models

1: **Input:** Model $\boldsymbol{x}(\boldsymbol{\lambda}) = \boldsymbol{h}_{\theta}(\boldsymbol{\lambda})$
2: Initialize the parameters $\boldsymbol{\theta}_0$
3: **for** $t = 1$ to T **do**
4: Sample preferences $\{\boldsymbol{\lambda}_k\}_{k=1}^K \sim \Lambda$
5: Update $\boldsymbol{\theta}_t$ with gradient descent in (9)
6: **end for**
7: **Output:** Model $\boldsymbol{x}(\boldsymbol{\lambda}) = h_{\boldsymbol{\theta}_T}(\boldsymbol{\lambda})$

If the model is perfectly learned, the obtained approximate Pareto set $\mathcal{M}_{\text{psl}} = \{\boldsymbol{x} = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}) | \boldsymbol{\lambda} \in \Lambda\}$ should be equal to $\mathcal{M}_{\text{tch_aug}}$. However, it is difficult to directly optimize (8) due to the expectation over infinite preferences ($|\boldsymbol{\Lambda}| = \infty$). We use Monte Carlo sampling and gradient descent to iteratively learn the model with the surrogate model:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \sum_{k=1}^{K} \nabla_{\boldsymbol{\theta}} \hat{g}_{\text{tch_aug}}(\boldsymbol{x} = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_k) | \boldsymbol{\lambda}_k), \quad (9)$$

where we randomly sample $K = 10$ different valid preferences $\{\boldsymbol{\lambda}_1, \cdots, \boldsymbol{\lambda}_K\} \sim \Lambda$ at each iteration in this work. Here $\hat{g}_{\text{tch_aug}}(\cdot)$ is the augmented Tchebycheff scalarization with predicted objective values:

$$\hat{g}_{\text{tch_aug}}(\boldsymbol{x} | \boldsymbol{\lambda}) = \max_{1 \le i \le m} \{\lambda_i (\hat{f}_i(\boldsymbol{x}) - (z_i^* - \varepsilon))\} + \rho \sum_{i=1}^{m} \lambda_i \hat{f}_i(\boldsymbol{x}). \quad (10)$$

One design choice left is the surrogate objective $\hat{\boldsymbol{f}}(\boldsymbol{x})$. If we only want to get the current predictive Pareto front, it is straightforward to use the predictive mean as the surrogate value. However, for Bayesian optimization, we should take the uncertainty into account to balance exploitation and exploration. Many widely-used criteria, such as expectation improvement (EI) [49] and upper confidence bound (UCB) [67], could be a more reasonable choice. In this work, we use the lower confidence bound for minimization problems:

$$\hat{\boldsymbol{f}}(\boldsymbol{x}) = \hat{\boldsymbol{\mu}}(\boldsymbol{x}) - \beta \hat{\boldsymbol{\sigma}}(\boldsymbol{x}). \quad (11)$$

We simply set $\beta = \frac{1}{2}$ and discuss the performance with other surrogate objectives in Appendix.

The expensive objective function $\boldsymbol{f}(\boldsymbol{x})$ is usually black-box and non-differentiable, but we can easily obtain the gradients for the Gaussian process and the set model. Indeed, gradient-based methods have been widely used for optimizing the acquisition function in both BO [79, 78] and MOBO [45, 13]. The max operator in Tchebycheff scalarization is technically only subdifferentiable, but it is known to have good subgradients [78] for surrogate optimization and preserve convexity if the objectives $\{\hat{f}_i(\boldsymbol{x})\}_{i=1}^m$ are all convex [7].

The Pareto set learning algorithm with Gaussian process models is summarized in **Algorithm 1**. We find the simple random initialization and gradient descent are enough to learn a good Pareto set approximation. The overparameterized neural network could be beneficial to overcome potential non-convexity [44].

4.3 Batched selection on approximate Pareto set

In this subsection, we propose a lightweight yet efficient batched acquisition search for MOBO with the learned Pareto set model. The algorithm framework is shown in **Algorithm 2**. The crucial difference with other MOBO approaches is that we build a set model at each iteration for batched solution selection as shown in **Algorithm 1** and **Algorithm 3**. The batched selection procedure contains two closely related steps:

Algorithm 2 MOBO with PSL

Input: Black-box vector-valued function $\boldsymbol{f}(\boldsymbol{x})$
Initial Samples $\{\boldsymbol{X}_0, \boldsymbol{y}_0\}$
for $t = 1$ to T **do**
  Train GPs based on $\{\boldsymbol{X}_{t-1}, \boldsymbol{y}_{t-1}\}$
  Learn set model $\boldsymbol{h}_{\boldsymbol{\theta}_t}(\boldsymbol{\lambda})$ with GPs (**Alg. 1**)
  Select $\{\boldsymbol{x}^{(b)}\}_{b=1}^{B}$ with the set model (**Alg. 3**)
  $\boldsymbol{X}_t \leftarrow \boldsymbol{X}_{t-1} \cup \{\boldsymbol{x}^{(b)}\}_{b=1}^{B}$,
  $\boldsymbol{y}_t \leftarrow \boldsymbol{y}_{t-1} \cup \boldsymbol{f}(\{\boldsymbol{x}^{(b)}\}_{b=1}^{B})$
end for
Output: $\{\boldsymbol{X}_t, \boldsymbol{y}_t\}$ and final set model $h_{\boldsymbol{\theta}_T}(\boldsymbol{\lambda})$

Algorithm 3 Batch Selection with PSL

Input: Model $\boldsymbol{x}(\boldsymbol{\lambda}) = \boldsymbol{h}_\theta(\boldsymbol{\lambda})$, Batch Size B
Sample P preferences $\{\boldsymbol{\lambda}^{(p)}\}_{p=1}^{P} \sim \Lambda$
Generate solutions $\boldsymbol{X} = \{\boldsymbol{x}(\boldsymbol{\lambda}^{(p)})\}_{p=1}^{P}$ on $\mathcal{M}_{\text{psl}}$
Find subset $\{\boldsymbol{x}^{(b)}\}_{b=1}^{B} \subset \boldsymbol{X}$ that has the highest $\text{HVI}(\hat{\boldsymbol{f}}(\{\boldsymbol{x}^{(b)}\}_{b=1}^{B}))$
Output: Batch solutions $\{\boldsymbol{x}^{(b)}\}_{b=1}^{B}$

Batch Sampling on Approximate Pareto Set. Our model naturally supports generating arbitrary number of solutions in batch. If the decision-maker's preferences are available, we can use preference-based sampling in this step. Without any prior knowledge, we uniformly sample P valid preferences $\{\boldsymbol{\lambda}^{(p)}\}_{p=1}^{P}$, and generate the corresponding solutions $\boldsymbol{X} = \{\boldsymbol{x}(\boldsymbol{\lambda}^{(p)})\}_{p=1}^{P}$ on the approximate Pareto set $\mathcal{M}_{\text{psl}}$.

Batch Selection. At each iteration of MOBO, we typically select a small number B (e.g., 5) of solutions $\boldsymbol{X}_B = \{\boldsymbol{x}^{(b)}\}_{b=1}^{B}$ from the sampled solutions $\boldsymbol{X}$ for expensive evaluations. Taking the already evaluated solutions $\{\boldsymbol{X}_{t-1}, \boldsymbol{y}_{t-1}\}$ into consideration, we can use the predicted hypervolume improvement (HVI) for batch selection:

$$\text{HVI}(\hat{\boldsymbol{f}}(\boldsymbol{X}_B)) = \text{HV}(\boldsymbol{y}_{t-1} \cup \hat{\boldsymbol{f}}(\boldsymbol{X}_B)) - \text{HV}(\boldsymbol{y}_{t-1}), \tag{12}$$

where $\boldsymbol{X}_B = \{\boldsymbol{x}^{(b)}\}_{b=1}^{B}$ are selected solutions, and $\hat{\boldsymbol{f}}(\boldsymbol{X}_B)$ are the surrogate values. The hypervolume [85] $\text{HV}(\boldsymbol{y}) = \textbf{Vol}(\boldsymbol{S})$ measures the volume of $\boldsymbol{S}$ dominated by a set $\boldsymbol{y}$ in the objective space:

$$\boldsymbol{S} = \{r \in \mathbb{R}^m \mid \exists y \in \boldsymbol{y} \text{ such that } y \prec r \prec r^*\}, \tag{13}$$

where r^* is a reference point that dominated by all $y \in \boldsymbol{y}$.

A better trade-off set will have a larger hypervolume, and the ground truth Pareto set always has the largest one. We want to select a set of $\boldsymbol{X}_B$ such that their corresponding objective values $\hat{\boldsymbol{f}}(\boldsymbol{X}_B)$ maximize $\text{HVI}(\hat{\boldsymbol{f}}(\boldsymbol{X}_B))$. It would be computationally intensive to find a set to exactly maximize the hypervolume improvement (12). In this work, we select the set $\boldsymbol{X}_B$ with a sequential greedy manner from $\boldsymbol{X}$ where $|\boldsymbol{X}| = P = 1,000$ for all problems. More details can be found in Appendix.

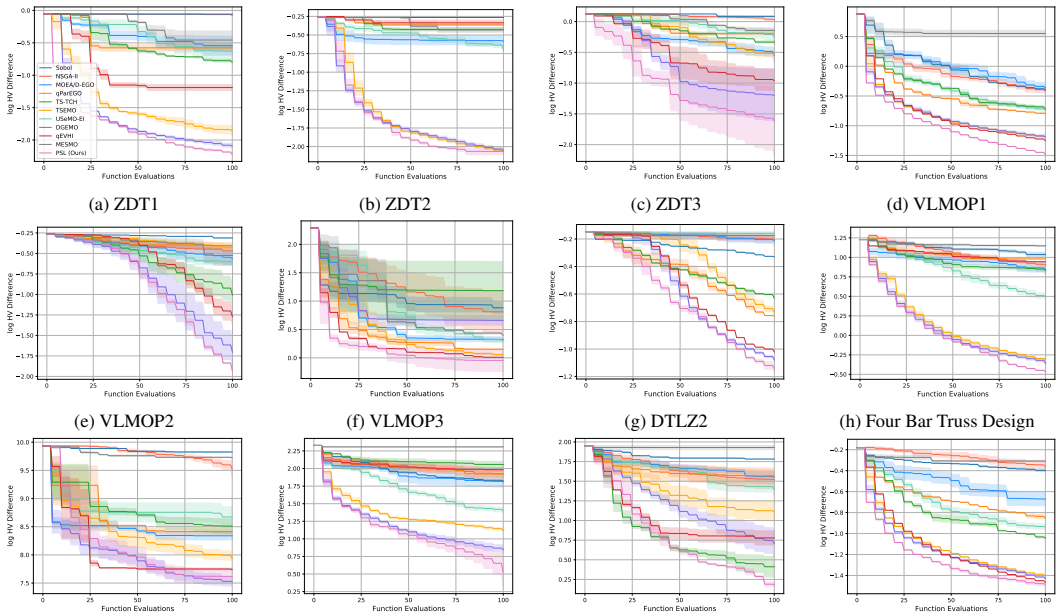
5 Experiments

In this section, we mainly compare the proposed PSL method with other MOBO approaches on the performance of evaluated solutions. We also analyze the quality of the learned Pareto set, which is not supported by other methods.

Baseline Algorithms. We consider different widely-used MOBO methods and two model-free approaches as baselines. The implementations of NSGA-II [16], MOEA/D-EGO [82], TSEMO [8], USeMO-EI [5], DGEMO [45] are from DGEMO's open-source codebase[1] based on pymoo[2] [6]. The implementations of scrambled Sobol sequence, TS-TCH [52], MESMO [4], qParEGO [39], and qEHVI [13] and from BoTorch[3] [3]. We implement PSL in Pytorch [54], and will open-source our code upon publication[4].

Benchmarks and Real-World Problems. The algorithms are first compared on the widely-used ZDT1-3 [84], VLMOP1-3 [74] and DTLZ2 [17] benchmark problems. Then we also conduct experiments on 5 different real-world multi-objective engineering design problems (RE) [71], including

[1] https://github.com/yunshengtian/DGEMO
[2] https://pymoo.org/problems/index.html
[3] https://github:com/pytorch/botorch
[4] To be open-sourced.



(a) ZDT1 (b) ZDT2 (c) ZDT3 (d) VLMOP1

(e) VLMOP2 (f) VLMOP3 (g) DTLZ2 (h) Four Bar Truss Design

(i) Pressure Vessel Design (j) Disk Brake Design (k) Gear Train Design (l) Rocket Injector Design

Figure 5: The log hypervolume difference w.r.t. the number of expensive evaluation for different algorithms. The solid line is the mean value averaged over 10 independent runs for each algorithm, and the shaded region is the standard deviation around the mean value. **The label of each algorithm can be found in subfig(a).**

1) four bar truss design [10]; 2) pressure vessel design [40]; 3) disk brake design [60]; 4) gear train design [15] and 5) rocket injector design [73]. Details of these problems can be found in Appendix.

Experiment Setting. For each experiment, we randomly generate 10 initial solutions for expensive evaluations, and then conduct MOBO with 20 batched evaluations with batch size 5. Therefore, there are total 110 expensive evaluations. For an experiment, all algorithms are independently run 10 times with different random seeds. We use the hypervolume indicator [85] as the metric to compare the quality of evaluated solutions chosen by different MOBO algorithms, which is monotonic to the Pareto dominance relation. The ground truth Pareto front will always have the best (highest) hypervolume. Detailed experimental setting can be found in Appendix.

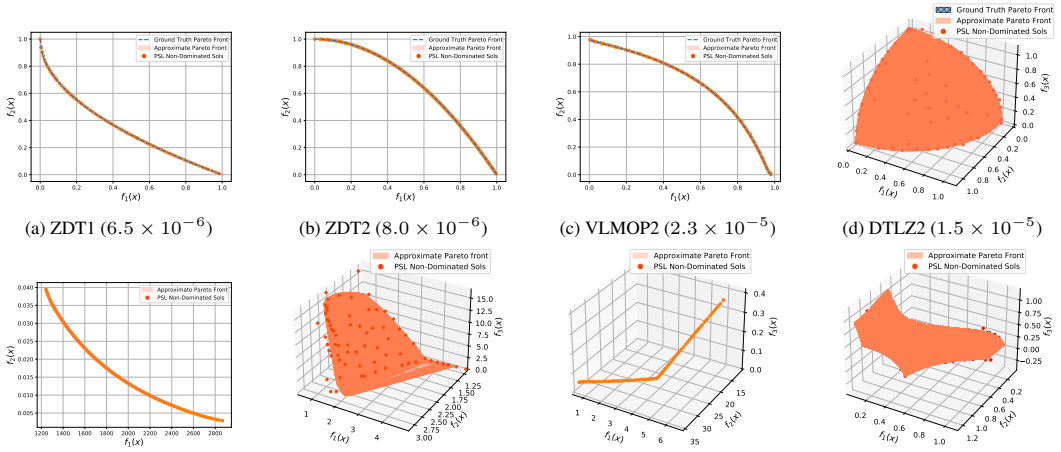
5.1 Experimental results and analysis

Table 1: Algorithm runtime per iteration (in seconds).

Problem	#objs	MOEA/D-EGO	TSEMO	USeMO-EI	DGEMO	qEHVI	PSL(Ours): Model + Selection
ZDT1	2	64.36	5.82	7.28	68.52	38.52	5.73 + 1.19 = 6.92
DTLZ2	3	71.83	7.28	8.76	83.57	75.92	7.02 + 1.59 = 8.61

MOBO Performance. We compare PSL with other MOBO methods on the performance of evaluated solutions. Figure 5 shows the log hypervolume difference to the true/approximate Pareto front for the synthetic/real-world problems during the optimization process. The approximate Pareto fronts for the real-world design problems are from Tanabe and Ishibuchi [71] with a large number of evaluations, which are also used in other MOBO works. In most experiments, our proposed PSL method converges faster and achieves better final results than other MOBO methods. As shown in Table 1, PSL has a shorter or comparable total runtime per iteration with other MOBO methods, which can be ignored in the expensive optimization problems (might take days). The promising results validate that the Pareto set learning approach can significantly improve the MOBO performance for expensive optimization problems.

The Learned Pareto Set. We present the approximate Pareto set learned by our proposed PSL method at the end of optimization in Figure 6, which is not supported by other MOBO methods. According to the results, PSL can successfully learn the Pareto sets for different benchmarks and



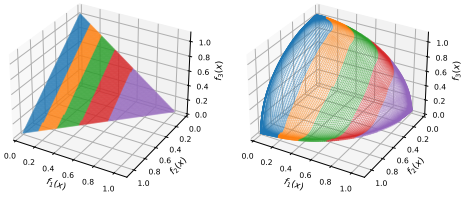
(a) ZDT1 (6.5×10^{-6}) (b) ZDT2 (8.0×10^{-6}) (c) VLMOP2 (2.3×10^{-5}) (d) DTLZ2 (1.5×10^{-5})

(e) Four Bar Truss (2×10^{-4}) (f) Disk Brake (7.7×10^{-3}) (g) Gear Train (2.4×10^{-4}) (h) Rocket Injector (5×10^{-4})

Figure 6: **The Learned Pareto Fronts (Relative Hypervolume Difference) by PSL:** Our learned Pareto fronts can match the ground truth Pareto fronts for the benchmarks, and have small relative hypervolume differences to the approximate Pareto fronts for real-world design problems. The learned Pareto front can well represent the optimal trade-offs among different objectives and provide valuable information to support flexible decision-making.

real-world application problems with different shapes of Pareto fronts. For benchmark problems, PSL can match the ground truth Pareto front with a small evaluation budget. For real-world applications, the approximate Pareto fronts can clearly capture the trade-off among different objectives and provide valuable information to support decision-making.

Flexible Trade-off Adjustment. With our model, the decision-makers can easily explore the whole approximate Pareto set by themselves to select the most preferred trade-off solution(s) as shown in Figure 7. No time-consuming communication between the optimization modeler and the decision-maker is required. By directly exploring the approximate Pareto front in an interactive manner, the decision-makers can observe and understand the connection between preferences and corresponding solutions in real-time. It is also beneficial for decision-makers to further adjust and assign their most accurate preferences. The ability to incorporate user's knowledge into decision making [24] could be crucial for many real-world applications.



(a) Different Preferences (b) DTLZ2

Figure 7: Different trade-off preferences and their corresponding solutions on the approximate Pareto set.

More Experimental results and Analysis can be found in Appendix due to the page limit.

6 Conclusion, limitation and future work

Conclusion. In this work, we have proposed a novel Pareto set learning method for expensive multi-objective optimization with a limited evaluation budget. The benefits of this approach are two-fold. First of all, by learning and utilizing the approximate Pareto set, it can serve as an efficient MOBO method that outperforms different existing approaches. Secondly, it also allows the decision-makers to easily explore any trade-off region on the approximate Pareto set, which supports flexible and interactive decision-making. We believe the proposed Pareto set learning method could provide a novel way to solve expensive multi-objective optimization problems.

Limitation and future work. PSL shares the same limitation with current MOBO methods that it could be challenging to solve problems with a large number of objectives or decision variables. In future work, it could be interesting to investigate how to make PSL more scalable to those problems.

References

[1] M. Abdolshah, A. Shilton, S. Rana, S. Gupta, and S. Venkatesh. Multi-objective bayesian optimisation with preferences over objectives. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[2] R. Astudillo and P. Frazier. Multi-attribute bayesian optimization with interactive preference learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[3] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] S. Belakaria and A. Deshwal. Max-value entropy search for multi-objective bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[5] S. Belakaria, A. Deshwal, N. K. Jayakodi, and J. R. Doppa. Uncertainty-aware search framework for multi-objective bayesian optimization. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[6] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[7] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

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

[9] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[10] F. Cheng and X. Li. Generalized center method for multiobjective engineering optimization. *Engineering Optimization*, 31(5):641–661, 1999.

[11] E. U. Choo and D. Atkins. Proper efficiency in nonconvex multicriteria programming. *Mathematics of Operations Research*, 8(3):467–470, 1983.

[12] I. Das and J. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1): 63–69, 1997.

[13] S. Daulton, M. Balandat, and E. Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[14] S. Daulton, M. Balandat, and E. Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[15] K. Deb and A. Srinivasan. Innovization: Innovating design principles through optimization. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2006.

[16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[17] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *IEEE Congress on Evolutionary Computation (CEC)*, 2002.

[18] T. Desautels, A. Krause, and J. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1): 3873–3923, 2014.

[19] A. Dosovitskiy and J. Djolonga. You only train once: Loss-conditional training of deep networks. *International Conference on Learning Representations (ICLR)*, 2019.

[20] M. Emmerich and J. Klinkenberg. The computation of the expected improvement in dominated hypervolume of pareto front approximations. *Rapport technique, Leiden University*, 34, 2008.

[21] M. Emmerich, K. Giannakoglou, and B. Naujoks. Single-and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[22] D. Eriksson, P. I.-J. Chuang, S. Daulton, A. Aly, A. Babu, A. Shrivastava, P. Xia, S. Zhao, G. Venkatesh, and M. Balandat. Latency-aware neural architecture search with multi-objective bayesian optimization. *arXiv preprint arXiv:2106.11890*, 2021.

[23] P. I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[24] R. Garnett. *Bayesian Optimization*. Cambridge University Press, 2022. in preparation.

[25] I. Giagkiozis and P. J. Fleming. Pareto front estimation for decision making. *Evolutionary computation*, 22(4):651–678, 2014.

[26] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[27] P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

[28] D. Hernández-Lobato, J. Hernandez-Lobato, A. Shah, and R. Adams. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning (ICML)*, 2016.

[29] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[30] C. Hillermeier. Generalized homotopy approach to multiobjective optimization. *Journal of Optimization Theory and Applications*, 110(3):557–583, 2001.

[31] M. W. Hoffman and Z. Ghahramani. Output-space predictive entropy search for flexible global optimization. In *NIPS workshop on Bayesian Optimization*, pages 1–5, 2015.

[32] K. M. Jablonka, G. M. Jothiappan, S. Wang, B. Smit, and B. Yoo. Bias free multiobjective active learning for materials design and discovery. *Nature Communications*, 12(1):1–10, 2021.

[33] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.

[34] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[35] I. Kaliszewski. A modified weighted tchebycheff metric for multiple objective programming. *Computers & operations research*, 14(4):315–323, 1987.

[36] K. Kawaguchi, L. P. Kaelbling, and T. Lozano-Pérez. Bayesian optimization with exponential convergence. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[37] A. J. Keane. Statistical improvement criteria for use in multiobjective design optimization. *AIAA journal*, 44(4):879–891, 2006.

[38] N. Khan, D. E. Goldberg, and M. Pelikan. Multi-objective bayesian optimization algorithm. In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 684–684. Citeseer, 2002.

[39] J. 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.

[40] S. Kramer. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design*, 116: 405, 1994.

[41] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

[42] M. Laumanns and J. Ocenasek. Bayesian optimization algorithms for multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature (PPSN)*, 2002.

[43] X. Lin, Z. Yang, Q. Zhang, and S. Kwong. Controllable pareto multi-task learning. *arXiv preprint arXiv:2010.06313*, 2020.

[44] D. Lopez-Paz and L. Sagun. Easing non-convex optimization with neural networks. In *International Conference on Learning Representations (ICLR) Workshops*, 2018.

[45] M. K. Lukovic, Y. Tian, and W. Matusik. Diversity-guided multi-objective bayesian optimization with batch evaluations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[46] P. Ma, T. Du, and W. Matusik. Efficient continuous pareto exploration in multi-task learning. *International Conference on Machine Learning (ICML)*, 2020.

[47] G. Malkomes, B. Cheng, E. H. Lee, and M. Mccourt. Beyond the pareto efficient frontier: Constraint active search for multiobjective experimental design. In *International Conference on Machine Learning (ICML)*, 2021.

[48] K. Miettinen. *Nonlinear multiobjective optimization*. Springer Science & Business Media, 1998.

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

[50] J. Mockus. *Bayesian approach to global optimization: theory and applications*. Kluwer Academic Publishers., 1989.

[51] A. Navon, A. Shamsian, G. Chechik, and E. Fetaya. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2021.

[52] B. Paria, K. Kandasamy, and B. Póczos. A flexible framework for multi-objective bayesian optimization using random scalarizations. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2020.

[53] S. Parisi, M. Pirotta, and M. Restelli. Multi-objective reinforcement learning through continuous pareto manifold approximation. *Journal of Artificial Intelligence Research (JAIR)*, 57:187–227, 2016.

[54] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[55] M. Pelikan, D. E. Goldberg, E. Cantú-Paz, et al. Boa: The bayesian optimization algorithm. In *Genetic and Evolutionary Computation Conference (GECCO)*, 1999.

[56] M. Pirotta, S. Parisi, and M. Restelli. Multi-objective reinforcement learning with continuous pareto frontier approximation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[57] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted s-metric selection. In *International Conference on Parallel Problem Solving from Nature (PPSN)*, 2008.

[58] J. Rakowska, R. T. Haftka, and L. T. Watson. Tracing the efficient curve for multi-objective control-structure optimization. *Computing Systems in Engineering*, 2(5-6):461–471, 1991.

[59] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[60] T. Ray and K. Liew. A swarm metaphor for multiobjective design optimization. *Engineering optimization*, 34(2):141–153, 2002.

[61] R. Roussel, A. Hanuka, and A. Edelen. Multiobjective bayesian optimization for online accelerator tuning. *Physical Review Accelerators and Beams*, 24(6):062801, 2021.

[62] O. Sener and V. Koltun. Learning to guide random search. In *International Conference on Learning Representations (ICLR)*, 2020.

[63] B. Shahriari, K. Swersky, Z. Wang, R. 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.

[64] A. Shoshan, R. Mechrez, and L. Zelnik-Manor. Dynamic-net: Tuning the objective without re-training for synthesis tasks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[65] J. Snoek, H. Larochelle, and R. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[66] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning (ICML)*, 2015.

[67] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *International Conference on Machine Learning (ICML)*, 2010.

[68] I. Steponavičė, R. J. Hyndman, K. Smith-Miles, and L. Villanova. Dynamic algorithm selection for pareto optimal set approximation. *Journal of Global Optimization*, 67(1):263–282, 2017.

[69] R. E. Steuer and E.-U. Choo. An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26(3):326–344, 1983.

[70] S. Suzuki, S. Takeno, T. Tamura, K. Shitara, and M. Karasuyama. Multi-objective bayesian optimization using pareto-frontier entropy. In *International Conference on Machine Learning (ICML)*, 2020.

[71] R. Tanabe and H. Ishibuchi. An easy-to-use real-world multi-objective optimization problem suite. *Applied Soft Computing*, 89:106078, 2020.

[72] A. Tripp, E. Daxberger, and J. M. Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[73] R. Vaidyanathan, K. Tucker, N. Papila, and W. Shyy. Cfd-based design optimization for single element rocket injector. In *Aerospace Sciences Meeting and Exhibit*, page 296, 2003.

[74] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm test suites. In *ACM Symposium on Applied Computing (SAC)*, 1999.

[75] L. Wang, R. Fonseca, and Y. Tian. Learning search space partition for black-box optimization using monte carlo tree search. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[76] Z. Wang and S. Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning (ICML)*, 2017.

[77] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. De Freitas. Bayesian optimization in high dimensions via random embeddings. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2013.

[78] J. T. Wilson, R. Moriconi, F. Hutter, and M. P. Deisenroth. The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*, 2017.

[79] J. Wu and P. Frazier. The parallel knowledge gradient method for batch bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[80] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[81] Q. Zhang, A. Zhou, and Y. Jin. Rm-meda: A regularity model-based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 12(1):41–63, 2008.

[82] Q. Zhang, W. Liu, E. Tsang, and B. Virginas. Expensive multiobjective optimization by moea/d with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2010.

[83] Y. Zhao, L. Wang, K. Yang, T. Zhang, T. Guo, and Y. Tian. Multi-objective optimization by learning space partitions. In *International Conference on Learning Representations (ICLR)*, 2022.

[84] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[85] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, 2007.

[86] M. Zuluaga, G. Sergent, A. Krause, and M. Püschel. Active learning for multi-objective optimization. In *International Conference on Machine Learning (ICML)*, 2013.

[87] M. Zuluaga, A. Krause, and M. Püschel. ε-pal: an active learning approach to the multi-objective optimization problem. *The Journal of Machine Learning Research*, 17(1):3619–3650, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not find any potential negative societal impacts of our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] Our code will be released upon publication.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Yes. See Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]