# Multi-Objective Bayesian Optimization with Pareto Set Learning

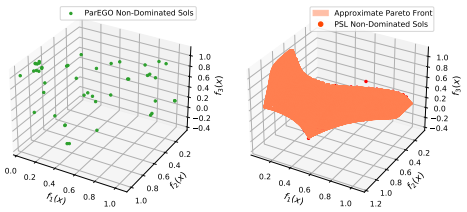
**Anonymous Author(s)**
Affiliation
Address
email

## 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 [34], building a neural network with high accuracy and low latency [24], and improving the quality while minimizing total charge in particle accelerator tuning [66]. 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 [40, 44, 39], typically directly generalized from the single-objective Bayesian optimization (BO) [53, 35, 9, 68, 25], 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 [51]. 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 [49].

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.

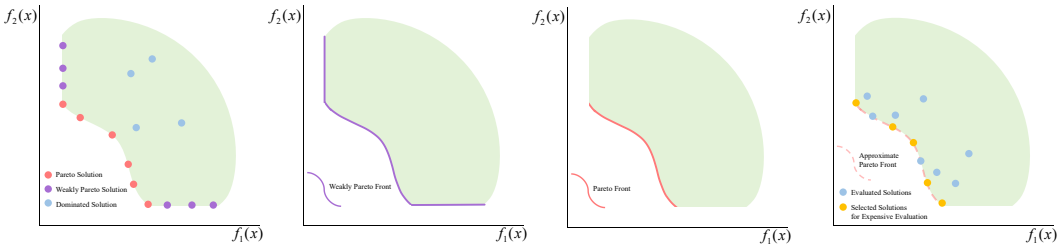
## 2 Related work

**Bayesian Optimization.** Surrogate model-based methods have been widely studied for expensive optimization [43, 36, 58, 70]. 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 [72], high-dimensional optimization [82], batch evaluation [20], scalable optimization [71], and theoretical analysis [38]. Most of them are on single-objective optimization. We refer readers to Garnett [26] 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 [41] and TS-TCH [55] 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 [87] extend the MOEA/D framework [85] to solve a set of surrogate scalarized subproblems simultaneously. SMS-EGO [60] and PAL [92, 93] generalize the upper confidence bound (UCB) to multi-objective optimization. Emmerich et al. [23] and Emmerich and Klinkenberg [22] propose the probability of improvement (PI) and expected improvement (EI) for multi-objective hypervolume. Entropy search methods [29, 31, 33, 81] have also been explored in multi-objective optimization [30, 4, 75]. 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 [73].

The current works have investigated the diverse batch evaluation [47], efficient hypervolume improvement calculation [13], multiple noisy objectives [14], and decision criteria beyond Pareto optimality [49]. Some attempts have been made to incorporate the decision-maker's preference into MOBO [1, 55, 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 [67] learn the geometric structure of the problem in an online manner to accelerate optimization. Wang et al. [80] and Zhao et al. [89] 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 [28, 77] 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 [61, 32, 27]. Pirotta et al. [59] and Parisi et al. [56] 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 [69, 21] and multi-task learning [48, 54, 45]. 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 [51, 32, 86]. The number of Pareto solutions could be infinite ($|\mathcal{M}_{\text{ps}}| = \infty$).

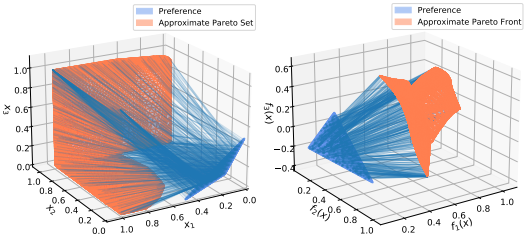
**Bayesian Optimization (BO) [26]** is a model-based method for solving expensive black-box optimization problems. Given a set of already-evaluated solutions $\{\boldsymbol{X}, \boldsymbol{y}\}$, BO builds surrogate models (e.g., Gaussian process) for each objective, and defines acquisition function(s) to leverage the surrogate objective values for navigating the search space. Only promising solutions will be selected for expensive evaluation. We refer interesting reader to [26] for a detailed introduction.

**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 $\boldsymbol{f}(\mathcal{M}_{\text{ps}})$. 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 $\Lambda$ 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

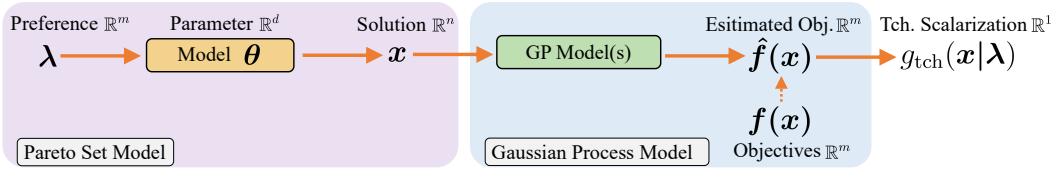



Figure 4: **Pareto Set Learning for Multi-Objective Bayesian Optimization:** **(a)** The Pareto set model learns a parameterized mapping from any valid preference $\boldsymbol{\lambda} \in \Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum \lambda_i = 1\}$ to its corresponding solution $\boldsymbol{x}(\boldsymbol{\lambda}) \in \mathbb{R}^n$. **(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.

further avoided by the augmented Tchebycheff approach [74, 37]. 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 $\rho$ is a sufficiently small positive scalar depends on the problem and current solution location. This form of augmentation has also been used in ParEGO [41]. With the augmentation term, the 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 $\Lambda$ 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 [67]. 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 [47, 49].

**Gaussian Process Model.** A single-objective Gaussian process [62] 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:

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

**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:

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

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:

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

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 posterior mean as the surrogate value. The approximate Pareto front under the posterior mean could provide valuable information to decision-makers. However, for Bayesian optimization, we should take the uncertainty into account to balance exploitation and exploration. Many widely-used criteria, such as expected improvement (EI) [52] and upper confidence bound (UCB) [72], could be a more reasonable choice. In this work, we use the lower confidence bound for minimization problems:

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

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 [84, 83] and MOBO [47, 13]. The max operator in Tchebycheff scalarization is technically only subdifferentiable, but it is known to have good subgradients [83] 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 [46].

### 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. To take all already evaluated solutions $\{\boldsymbol{X}_{t-1}, \boldsymbol{y}_{t-1}\}$ into consideration, we use the hypervolume [91] as the selection criteria. The hypervolume $\text{HV}(\boldsymbol{y}) = \textbf{Vol}(\boldsymbol{S})$ measures the volume of $\boldsymbol{S}$ dominated by a set $\boldsymbol{y}$ in the objective space:

$$\boldsymbol{S} = \{r \in \mathbb{R}^m \mid \exists y \in \boldsymbol{y} \text{ such that } y \prec r \prec r^*\}, \tag{12}$$

where $r^*$ is a reference point that dominated by all $y \in \boldsymbol{y}$. The hypervolume improvement (HVI) of a set $\boldsymbol{X}_B$ with respect to the already evaluated solutions $\{\boldsymbol{X}_{t-1}, \boldsymbol{y}_{t-1}\}$ can be defined as:

$$\text{HVI}(\hat{\boldsymbol{f}}(\boldsymbol{X}_B)) = \text{HV}(\boldsymbol{y}_{t-1} \cup \hat{\boldsymbol{f}}(\boldsymbol{X}_B)) - \text{HV}(\boldsymbol{y}_{t-1}), \tag{13}$$

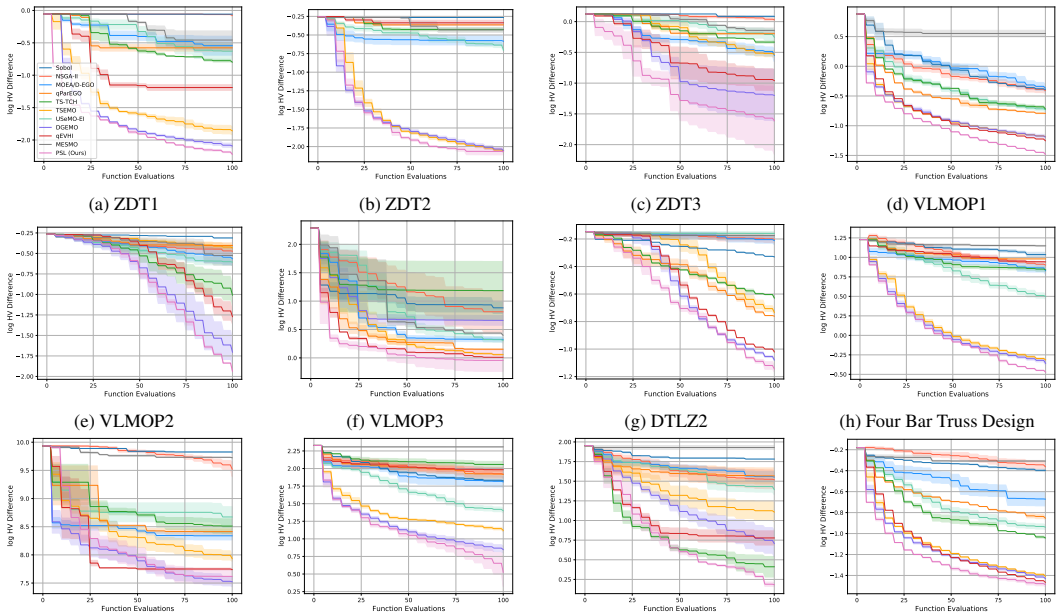
where $\boldsymbol{X}_B = \{\boldsymbol{x}^{(b)}\}_{b=1}^{B}$ are selected solutions, and $\hat{\boldsymbol{f}}(\boldsymbol{X}_B)$ are the surrogate values. In this work, we mainly use the LCB (11) as the surrogate value for Bayesian optimization, and provide an ablation study of different surrogate values in Appendix. F.3.

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 jointly optimize a set of solutions to exactly maximize the hypervolume improvement (13), and therefore sequential greedy selection is typically used [13]. 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 [18], MOEA/D-EGO [87], TSEMO [8],




(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).**

USeMO-EI [5], DGEMO [47] are from DGEMO's open-source codebase[1] based on pymoo[2] [6]. The implementations of scrambled Sobol sequence, TS-TCH [55], MESMO [4], qParEGO [41], and qEHVI [13] and from BoTorch[3] [3]. We implement PSL in Pytorch [57], 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 [90], VLMOP1-3 [79] and DTLZ2 [19] benchmark problems. Then we also conduct experiments on 5 different real-world multi-objective engineering design problems (RE) [76], including 1) four bar truss design [10]; 2) pressure vessel design [42]; 3) disk brake design [63]; 4) gear train design [17] and 5) rocket injector design [78]. 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 [91] 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.

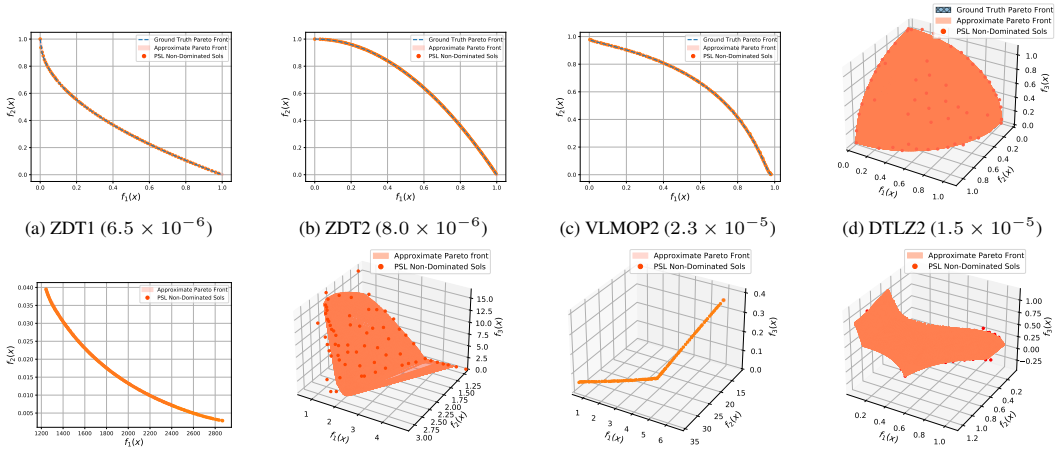
### 5.1 Experimental results and analysis

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

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



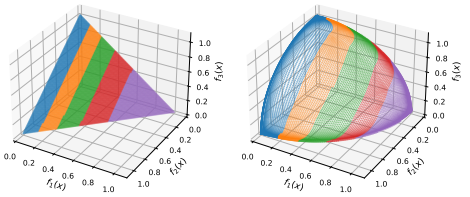

(a) ZDT1 ($6.5 \times 10^{-6}$) (b) ZDT2 ($8.0 \times 10^{-6}$) (c) VLMOP2 ($2.3 \times 10^{-5}$) (d) DTLZ2 ($1.5 \times 10^{-5}$)

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

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.

synthetic/real-world problems during the optimization process. The approximate Pareto fronts for the real-world design problems are from Tanabe and Ishibuchi [76] 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 PSL approach can significantly improve the MOBO performance for expensive optimization problems.

**The Learned Pareto Set.** We present the approximate Pareto set learned by PSL under the posterior mean after 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 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 capture the trade-off among objectives and provide valuable information to support decision-making.



(a) Different Preferences (b) DTLZ2

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

**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 [26] could be crucial for many real-world applications.

## 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.

# 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] S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Multi-objective bayesian optimization over high-dimensional search spaces. *arXiv preprint arXiv:2109.10964*, 2021.

[16] G. De Ath, R. M. Everson, A. A. Rahat, and J. E. Fieldsend. Greed is good: Exploration and exploitation trade-offs in bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(1):1–22, 2021.

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

[18] 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.

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

[20] 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.

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

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

[23] 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.

[24] 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.

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

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

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

[28] 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.

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

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

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

[36] 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.

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

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

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

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

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

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

[47] 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.

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

[49] 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.

[50] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1): 55–61, 2000.

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

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

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

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

[55] 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.

[56] 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.

[57] 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.

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

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

[60] 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.

[61] 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.

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

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

[64] F. Rehbach, M. Zaefferer, B. Naujoks, and T. Bartz-Beielstein. Expected improvement versus predicted value in surrogate-based optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 868–876, 2020.

[65] C. Romero. Extended lexicographic goal programming: a unifying approach. *Omega*, 29(1): 63–71, 2001.

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

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

[68] 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.

[69] 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.

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

[71] 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.

[72] 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.

[73] 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.

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

[75] 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.

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

[77] 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.

[78] 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.

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

[80] 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.

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

[82] 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.

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

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

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

[86] 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.

[87] 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.

[88] R. Zhang and D. Golovin. Random hypervolume scalarizations for provable multi-objective black box optimization. In *International Conference on Machine Learning*, pages 11096–11105. PMLR, 2020.

[89] 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.

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

[91] 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.

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

[93] M. Zuluaga, A. Krause, and M. Püschel. $\varepsilon$-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]

We provide more discussions, algorithm and problem details, and extra experimental results in this appendix:

- More discussions of the proposed algorithm are temporally provided in Section A.
- The limitations and potential future improvement for PSL are discussed in Section B.
- Potential societal impact of this work is discussed in Section C.
- Details of the set model and batch selection algorithm are provided in Section D.
- Details of the benchmark and real-world application problems are given in Section E. (definition of 6 new problems are added.)
- More experimental results and analyses are presented in Section F. (many new results)

# A More Discussion

Since no extra page is allowed for the main paper during the response period, we temporally put all new discussions for our proposed algorithm in this section. Will move part of these discussions into the main paper and reorganize the rest materials once an extra page is allowed.

## A.1 Motivation: Pareto set model over simple scalarization

Our proposed Pareto set learning method has a close relation to the scalarization-based methods. But it can overcome a major disadvantage of the current scalarization methods for expensive optimization.

The scalarization methods randomly select one (e.g., ParEGO) or a batch of scalarized subproblems (e.g., MOEA/D-EGO, TS-TCH, and qParEGO) at each iteration. By optimizing the acquisition function (e.g., EI, LCB, Thompson Sampling etc) on each scalarization, they generate a batch of solutions for expensive evaluation.

A major limitation of all these scalarization methods is that they do not explicitly consider those already-evaluated solutions, neither for choosing the weighted scalarization, nor for selecting the next (batch of) solution(s) for expensive evaluation. Therefore, the selected weighted scalarization(s) might be close to those already-evaluated solutions. In other words, even the obtained solution(s), can maximize the EI/LCB for the selected scalarization, could still be similar to those already-evaluated ones, and indeed not an optimal choice for multi-objective optimization. This limitation leads to inferior performance of these scalarization methods, as reported in [13, 47] and in our experimental results.

To overcome this limitation, our proposed method has a two-stage approach:

- Stage 1: Using the Pareto set model, it first efficiently samples a dense set of candidate solutions to cover the whole approximate Pareto front (of posterior mean, EI, or LCB etc).
- Stage 2: Then it selects a small batch of appropriate solutions from this dense set for expensive evaluation. For selection, we use the HVI criteria to take both the selected and already-evaluated solutions into consideration.

In this way, our proposed method can efficiently explore the whole approximate Pareto front, and choose the most appropriate solutions (on the approximate Pareto front, while far from the selected and already-evaluated ones) for expensive evaluation. The experimental results have validated the efficiency of the proposed method.

## A.2 Hypervolume improvement for batch selection

As discussed above, the current scalarization approach (e.g., (q)ParEGO) randomly selects different scalarization(s) at each iteration, which is to randomly generate a small set of sparse solutions on the approximate Pareto set (with EI, LCB) for evaluation. In other words, the selected solutions are a small random subset of the whole approximate Pareto set (our Stage 1). In our proposed algorithm, we use the HVI to select a small batch of solutions from the dense approximate Pareto set, which can be aware of both selected and already-evaluated solutions and leads to better optimization performance (our Stage 2).

Efficiently finding a batch of solutions to optimize the EI/LCB of HVI could be very challenging. qEHVI [13] and qNEHVI [14] are two promising approaches along this direction. From the viewpoint of optimizing HVI, our proposed method restricts the search procedure only on the approximate Pareto set, which is a low-dimensional (m-1) manifold in the decision space. Therefore, we can use a simple two-stage sample-then-select approach to find the batch of solutions for evaluations. For the optimal situation, the set of solutions that optimize HVI should all be on the Pareto front. However, we agree with the reviewer that if an efficient method exists, directly optimizing the EI/LCB of HVI on the whole search space should be a principled approach for batch selection.

On the other hand, the scalarization-based approach could have a close relationship with the hypervolume [88]. It will be very interesting to study how to better leverage this relation for designing a more efficient algorithm, such as learning the whole Pareto set while inference the location of solutions (on the learned Pareto set) to optimize HVI at a single stage. These will be our future work.

### A.3 Practicality of the approximate Pareto set

In real-world applications, the learned approximate Pareto set could not always be accurate. Therefore, it is risky to only rely on the approximate Pareto set to make a decision (see more discussion in the limitation section). With our proposed method, decision-makers can have both the evaluated solutions and the approximate Pareto set at the same time. The approximate Pareto set can provide extra useful information to better support their decision-making. More specifically,

- It can help the decision-makers to better understand the (approximate) trade-offs they will make when they choose an already evaluated solution as their final solution. The approximate Pareto set and the corresponding surrogate objective values around the chosen solution can provide valuable information to understand what we can gain or lose by adjusting the current chosen solution.
- It allows the decision-maker to explore the whole approximate Pareto set/front easily. If none of the already evaluated solutions can precisely satisfy the decision-maker's preferred trade-off (which could be possible for applications with infinite Pareto sets), the decision-maker can rely on the approximate Pareto set/front to choose the preferred solution for further evaluation.
- When the optimization modeler and the final decision-maker are not the same person, the approximate Pareto set/front provides a much more efficient way for them to communicate and discuss the whole trade-offs among different objectives during or post the optimization process. This demand is common and important for many applications [49].
- Finally, an important application for MOBO is to help domain experts efficiently conduct experiments. The approximate Pareto set/front might contain useful patterns and structures which can help the domain expert obtain more useful information from the experiment. It also provides a way for domain experts to incorporate their knowledge into the optimization process (e.g., choose the most concerned region, and eliminate some uninteresting locations). The proposed Pareto set learning method could be a novel approach to support "bring decision maker back into the loop" for Bayesian optimization [26].

Therefore, the learned approximate Pareto set could be a useful tool to support flexible decision-making for expensive multi-objective optimization.

## B Limitation and potential future work

### B.1 The approximate Pareto set could be inaccurate

It could be very risky, especially in those safety-critical applications, to only rely on an approximate Pareto set to make the final decision. The quality of the approximate Pareto set heavily depends on the quality of the surrogate models. If the evaluation budget is insufficient to build a good set of surrogate models (for each objective), we cannot obtain a good approximate Pareto set. This is also a challenge for the general (multi-objective) Bayesian optimization methods.

One possible method to address this challenge is to leverage the user preference information into the expensive optimization process [1, 55, 2]. In this way, the general MOBO method can spend the

limited evaluation budget mainly on the user-preferred region rather than the whole decision space. Similarly, our proposed method can only build a partial approximate Pareto set with the user-preferred trade-offs. However, the preference-based approach could be affected by the following limitation.

### B.2 Defining the user preference is challenging for black-box optimization

Although some scalarization methods (e.g., Chebyshev scalarization) have good theoretical properties to connect the scalars to their corresponding Pareto solutions, it could be not so easy to define user preferences in terms of scalars, especially in the black-box expensive optimization setting.

In many applications, the decision-makers might even do not know their actual preferences before making their decision. If the approximate Pareto set can be properly learned, instead of asking users to provide their preferences in terms of fixed scalars, our proposed approach can let them interactively explore the whole approximate Pareto set/Pareto front to select their most preferred solutions. In this interactive way, it could be much easier for the decision-makers to accurately express and assign their preferences. However, it could be difficult to precisely obtain user preference if a good approximate Pareto set is not available (e.g., in the early stage of optimization, or without enough budget).

The algorithm proposed in Abdolshah et al. [1] is an elegant and promising way to incorporate preference into the MOBO process via the preference-order constraints, which does not require any prior knowledge on the (approximate) Pareto front. The preference-order constraint also has a close relationship with the lexicographic approach for multi-objective optimization [65], which is a non-scalarizing method with good theoretical property (e.g., connection to weakly Pareto optimal solution). How to incorporate this information into our proposed Pareto set learning model for more flexible preference incorporation could be interesting for future work.

### B.3 Scalability

The scalability with the dimensions of search space is a main challenge for the general MOBO algorithms. It is also very difficult for our proposed PSL method to learn a good enough Pareto set for such large-scale problems. The PSL's performance depends on a set of good surrogate models, which requires a large number of evaluated solutions for the problem with a high-dimensional search space.

Recently, some efficient search region decomposition/management methods have been proposed to better allocate the limited computational budget for problems with high-dimensional search space. These ideas can be naturally generalized to solve expensive multi-objective optimization problems, such as MORBO and LaMOO. Since the search region management methods are algorithm-agnostic (e.g., a meta-algorithm), they can be combined with different (multi-objective) optimization algorithms. Studying how to efficiently combine the MORBO/LaMOO approach with our proposed PSL method will be an important future work to tackle this limitation.

## C Potential societal impact

Our proposed Pareto set learning method mainly has two strengths: (1) it is good for better multi-objective Bayesian optimization in terms of speed and sample efficiency, and (2) it can help decision-makers to navigate the estimated Pareto set and front for better decision-making. These strengths could lead to many positive potential societal impacts. For example, as an efficient MOBO algorithm, it can reduce the cost of obtaining a set of diverse Pareto solutions in many applications such as materials science, engineering design, and recommender systems. The learned approximate Pareto set provides a novel way for decision-makers to easily explore different trade-offs, which could be an important component for a more user-friendly multi-criteria decision system.

On the negative side, as discussed in the limitation section, there is currently no guarantee that the learned approximate Pareto set could be accurate. Solely relying on the approximate Pareto set to make the final decision could be risky. The decision-makers should leverage all the information they have to make the final decision. In addition, the leakage of the learned Pareto set model might unintentionally reveal the problem information and user preference, which should be avoided in real-world applications.

# D Model and algorithm details

## D.1 Pareto set model

**Model Structure.** In this work, we use a multi-layered perceptron (MLP) as the Pareto set model $\boldsymbol{x}(\boldsymbol{\lambda}) = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda})$. For all experiments, the Pareto set model has 2 hidden layers each with 256 hidden units, and the activation is ReLU. The input and output dimension of the set model is the number of objectives $m$ and the dimension of decision variables $n$, respectively. Therefore, the set model $h_{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ has the following structure:

$$\begin{aligned} h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}) :&\text{Input } \boldsymbol{\lambda} \to \text{Linear}(m, 256) \to \text{ReLU} \to \text{Linear}(256, 256) \\ &\to \text{ReLU} \to \text{Linear}(256, 256) \to \text{ReLU} \\ &\to \text{Linear}(256, n) \to \text{Output } \boldsymbol{x}(\boldsymbol{\lambda}), \end{aligned} \tag{14}$$

where the input is an $m$-dimensional preference $\boldsymbol{\lambda} \in \boldsymbol{\Lambda} = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum \lambda_i = 1\}$, the output is a $n$-dimensional decision variables $\boldsymbol{x} \in \mathbb{R}^n$, and $\boldsymbol{\theta} \in \mathbb{R}^d$ represents all learnable parameters for the Pareto set model. We build independent Gaussian process models with Matérn 5/2 kernel for each objective as the surrogate models. The objective to optimize is the augmented Tchebycheff scalarization on the surrogate value with respect to all valid preferences.

**Model Training.** In the proposed Pareto set learning method, we have the following model structure:

$$\boldsymbol{\lambda} \to \textbf{Pareto Set Model } \to \boldsymbol{x}(\boldsymbol{\lambda}) = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \tag{15}$$

$$\to \textbf{GP Model} \to \hat{\boldsymbol{f}}(\boldsymbol{x}) \to \textbf{TCH Scalarization } \hat{g}_{\text{tch_aug}}(\boldsymbol{x}(\boldsymbol{\lambda})), \tag{16}$$

where the preference $\boldsymbol{\lambda}$ goes through the Pareto set model $h_{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ and Gaussian process model, then gets the augmented Tchebycheff scalarized value $\hat{g}_{\text{tch_aug}}(\boldsymbol{x}(\boldsymbol{\lambda}))$ at the end (a detailed version can be found in Figure 4 in the main paper). To train the Pareto set model, we use gradient-based method to optimize the model parameter $\boldsymbol{\theta}$ with respect to the scalarized value $\hat{g}_{\text{tch_aug}}(\boldsymbol{x}(\boldsymbol{\lambda}))$. With the chain rule, we have:

$$\nabla_{\theta} \hat{g}_{\text{tch_aug}}(\boldsymbol{x} = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}) | \boldsymbol{\lambda}) = \frac{\partial \hat{g}_{\text{tch_aug}}}{\partial \boldsymbol{\theta}} = \frac{\partial \hat{g}_{\text{tch_aug}}}{\partial \hat{\boldsymbol{f}}} \frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\theta}}, \tag{17}$$

where we can calculate each term respectively.

- For the first term $\frac{\partial \hat{g}_{\text{tch_aug}}}{\partial \hat{\boldsymbol{f}}}$, the augmented Tchebycheff scalarization $\hat{g}_{\text{tch_aug}}(\boldsymbol{x}|\boldsymbol{\lambda}) = \max_{1 \le i \le m}\{\lambda_i(\hat{f}_i(\boldsymbol{x}) - (z_i^* - \varepsilon))\} + \rho \sum_{i=1}^m \lambda_i \hat{f}_i(\boldsymbol{x})$ has a max operator which is technically only subdifferentiable, but it is known to have good subgradients [83] for surrogate optimization. We simply take the subgradients for this term.
- The term $\frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{x}}$ is the gradient of surrogate values to the input $\boldsymbol{x}$. In this work, we build independent Gaussian process models for each objective with Matérn 5/2 kernel. The gradients of the kernel $\frac{\partial \boldsymbol{k}_i}{\partial \boldsymbol{x}}$, predictive mean $\frac{\partial \hat{\mu}_i}{\partial \boldsymbol{x}}$ and standard deviation $\frac{\partial \hat{\sigma}_i}{\partial \boldsymbol{x}}$ for each objective can be easily calculated. With these terms, the calculation for the surrogate value (e.g., predictive mean, UCB and EI) for each objective $\frac{\partial \hat{f}_i}{\partial \boldsymbol{x}}$ is also straightforward.
- The term $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\theta}} = \frac{\partial h_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}}$ is the gradient of the Pareto set model to the parameter $\boldsymbol{\theta}$. In this work, we use a simple MLP as the Pareto set model, and its gradient be easily obtained by auto-differentiation such as in PyTorch [57].

For **Algorithm 1** in the main paper, we randomly sample $K = 10$ different preferences $\{\boldsymbol{\lambda}_k\}_{k=1}^{K=10} \sim \boldsymbol{\Lambda}$ in batch at each step. Without any prior information (e.g., user's preference), we uniformly sample preference $\boldsymbol{\lambda}$ from $[0, 1]^m$ and then normalize it such that $\sum_{i=1}^m \boldsymbol{\lambda}_i = 1$. The total update step is $T = 500$ for training the Pareto set model at each iteration. The model optimizer is Adam with learning rate $\eta = 1\text{e} - 3$ and no weight decay. The learning process typically only requires less than 10 seconds to finish, which is enough to obtain a good Pareto set approximation. Detailed experimental results on the runtime can be found in Table 3.

We use the same model and training procedure for all experiments, and the only problem-dependent setting is the input/output dimension $m$ and $n$. The proposed Pareto set model and learning approach have robust and promising performances for different expensive multi-objective optimization benchmarks and real-world problems.

### D.2 Batch selection with Pareto set learning

**Algorithm 4** Sampling-based Greedy Batch Selection

**Input:** Evaluated Solutions $\{\boldsymbol{X}_{t-1}, \boldsymbol{y}_{t-1}\}$, Candidate Solutions $\boldsymbol{X}_C = \boldsymbol{X}$, Surrogate Models
Initialize the selected solution set $\boldsymbol{X}_B = \varnothing$ with predicted values $\hat{\boldsymbol{y}}_B = \varnothing$
**for** $t = 1$ to $B$ **do**
  Calculate predicted HVIs for each individual solution in $\boldsymbol{X}_C$ with respective to $\boldsymbol{y}_{t-1} \cup \hat{\boldsymbol{y}}_B$
  Choose the single solution $\boldsymbol{x}_s$ with the best predicted HVI
  $\boldsymbol{X}_B \leftarrow \boldsymbol{X}_B \cup \{\boldsymbol{x}_s\}$, $\hat{\boldsymbol{y}}_B \leftarrow \hat{\boldsymbol{y}}_B \cup \{\hat{\boldsymbol{f}}(\boldsymbol{x}_s)\}$, $\boldsymbol{X}_C \leftarrow \boldsymbol{X}_C \setminus \{\boldsymbol{x}_s\}$
**end for**
**Output:** The Selected Solutions $\boldsymbol{X}_B$.

The MOBO with Pareto Set Learning framework (e.g., **Algorithm 2** in the main paper) is similar to other MOBO methods, and the main difference is on the batch selection with the learned Pareto set (**Algorithm 3**). At each iteration, we first randomly sample $P = 1000$ preferences $\{\boldsymbol{\lambda}^{(p)}\}_{p=1}^{P=1000} \sim \Lambda$ and directly obtain their corresponding solutions $\boldsymbol{X} = \{\boldsymbol{x}(\boldsymbol{\lambda}^{(p)})\}_{p=1}^{P=1000}$ on the current learned Pareto set $\mathcal{M}_{\text{psl}}$. We then use the predicted hypervolume improvement (HVI) with respect to the already evaluated solutions $\{\boldsymbol{X}_{t-1}, \boldsymbol{y}_{t-1}\}$ to select a subset $\boldsymbol{X}_B \subset \boldsymbol{X}$ for expensive evaluation. The surrogate values for the candidate solutions are directly used for the predicted HVI calculation if they are the predicted mean $\hat{\boldsymbol{\mu}}$ or the lower confidence bound $\hat{\boldsymbol{\mu}} - \beta\boldsymbol{\sigma}$. When the surrogate values are expected improvement, although the approximate Pareto set is learned with EI for each scalarization, we simply let $\hat{\boldsymbol{f}}(\boldsymbol{x}_s) = \hat{\boldsymbol{\mu}} - \boldsymbol{\sigma}$ for the predicted HVI calculation. The main reasons for this choice are: 1) we want to avoid the (repeatedly) time-consuming Monte Carlo integration for calculating the expected hypervolume improvement; 2) the LCB (or the posterior mean only) is on the same scale as the value of those already-evaluated solutions, which make the calculation of HVI meaningful.

Since it is computationally intensive to find $B$ solutions to exactly optimize HVI, we select the solution batch $\boldsymbol{X}_B$ in a sequentially greedy manner from $\boldsymbol{X}$. Similar approaches have also been used in the current MOBO methods [47, 15]. The greedy selection approach is presented in **Algorithm 4**. We sequentially sample solutions from the candidate set $\boldsymbol{X}_C$, and add the single solution $\boldsymbol{x}_s$ with the best HVI value into the selected set $\boldsymbol{X}_B$. For all problems, the acquisition search procedure (solution sampling + individual search + batch selection) with batch size $B = 10$ typically costs less than 3 seconds. The detailed results can be found in Table 3.

# E Problem details

Table 2: Problem information and reference point for both synthetic benchmarks and real-world engineering design problems.

| Problem | n | m | Reference Point($\boldsymbol{r}$) |
|---|---|---|---|
| ZDT1 | 6 | 2 | (1.1,1.1) |
| ZDT2 | 6 | 2 | (1.1,1.1) |
| ZDT3 | 6 | 2 | (1.1,1.1) |
| VLMOP1 | 1 | 2 | (4.4,4.4) |
| VLMOP2 | 6 | 2 | (1.1,1.1) |
| VLMOP3 | 2 | 3 | (11,66,1.1) |
| DTLZ2 | 6 | 3 | (1.1,1.1,1.1) |
| Four Bar Truss Design | 4 | 2 | (3175.0065, 0.0400) |
| Pressure Vessel Design | 4 | 2 | (6437.2649, 1417536.7586) |
| Disk Brake Design | 4 | 3 | (5.8374, 3.4412, 27.5) |
| Gear Train Design | 4 | 3 | (6.5241, 61.6, 0.3913) |
| Rocket Injector Design | 4 | 3 | (1.0884, 1.0522, 1.0863) |

## E.1 Synthetic benchmark problems

We test our proposed PSL algorithm on different widely-used synthetic multi-objective optimization benchmark problems, namely ZDT1-3 [90], VLMOP1-3 [79], and DTLZ2 [19]. The input and output dimensions of these algorithms are shown in Table 2. These synthetic problems have known Pareto sets $\mathcal{M}_{\text{ps}}$ and Pareto fronts $\boldsymbol{f}(\mathcal{M}_{\text{ps}})$ with the nadir point:

$$\begin{aligned} &\boldsymbol{z}_{\text{nadir}} = (\max f_1(\boldsymbol{x}_1), \dots, \max f_m(\boldsymbol{x}_m)), \\ &\forall \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m \in \mathcal{M}_{\text{ps}}, \end{aligned} \tag{18}$$

where $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$ are solutions in the Pareto set. We set the reference point $\boldsymbol{r} = 1.1 \times \boldsymbol{z}_{\text{nadir}}$ for each problem.

## E.2 New problems with various shape of Pareto front

Among the synthetic benchmarks, the ZDT problem has a simple Pareto set, which is a line on the boundary of the decision space, and most decision variables of all Pareto optimal solutions share the same values (e.g., 0). These properties could make ZDT problems not appropriate for comparing different MOBO algorithms. We chose the ZDT problem for comparison solely due to it is widely used in multi-objective optimization, and our proposed PSL method is not specially designed for such problems.

In this subsection, to better evaluate our proposed PSL method, we propose 6 new test problems of which the Pareto sets are curves in the search space. The test problems (F1 - F6) are defined in the next page, and the experimental results on these newly proposed problems can be found in Appendix. F.1.

| | |
|---|---|
| F1 | $f_1(\boldsymbol{x}) = (1+\frac{s_1}{\lvert J_1\rvert})\boldsymbol{x}_1, \quad f_2(\boldsymbol{x}) = (1+\frac{s_2}{\lvert J_2\rvert})\left(1-\sqrt{\frac{\boldsymbol{x}_1}{1+\frac{s_2}{\lvert J_2\rvert}}}\right)$<br>where $s_1 = \sum_{j\in J_1}(\boldsymbol{x}_j - (2\boldsymbol{x}_1-1)^2)^2$ and $s_2 = \sum_{j\in J_2}(\boldsymbol{x}_j - (2\boldsymbol{x}_1-1)^2)^2$,<br>$J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$. |
| F2 | $f_1(\boldsymbol{x}) = (1+\frac{s_1}{\lvert J_1\rvert})\boldsymbol{x}_1, \quad f_2(\boldsymbol{x}) = (1+\frac{s_2}{\lvert J_2\rvert})\left(1-\sqrt{\frac{\boldsymbol{x}_1}{1+\frac{s_2}{\lvert J_2\rvert}}}\right)$<br>where $s_1 = \sum_{j\in J_1}(\boldsymbol{x}_j - \boldsymbol{x}_1^{0.5(1.0+\frac{3(j-2)}{n-2})})^2$ and $s_2 = \sum_{j\in J_2}(\boldsymbol{x}_j - \boldsymbol{x}_1^{0.5(1.0+\frac{3(j-2)}{n-2})})^2$,<br>$J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$. |
| F3 | $f_1(\boldsymbol{x}) = (1+\frac{s_1}{\lvert J_1\rvert})\boldsymbol{x}_1, \quad f_2(\boldsymbol{x}) = (1+\frac{s_2}{\lvert J_2\rvert})\left(1-\sqrt{\frac{\boldsymbol{x}_1}{1+\frac{s_2}{\lvert J_2\rvert}}}\right)$<br>where $s_1 = \sum_{j\in J_1}(\boldsymbol{x}_j - \sin(4\pi\boldsymbol{x}_1 + \frac{j\pi}{n}))^2$ and $s_2 = \sum_{j\in J_2}(\boldsymbol{x}_j - \sin(4\pi\boldsymbol{x}_1 + \frac{j\pi}{n}))^2$,<br>$J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$. |
| F4 | $f_1(\boldsymbol{x}) = (1+\frac{s_1}{\lvert J_1\rvert})\boldsymbol{x}_1, \quad f_2(\boldsymbol{x}) = (1+\frac{s_2}{\lvert J_2\rvert})\left(1-\sqrt{\frac{\boldsymbol{x}_1}{1+\frac{s_2}{\lvert J_2\rvert}}}\right)$<br>where $s_1 = \sum_{j\in J_1}(\boldsymbol{x}_j - 0.8\boldsymbol{x}_1\cos(4\pi\boldsymbol{x}_1 + \frac{j\pi}{n}))^2$ and $s_2 = \sum_{j\in J_2}(\boldsymbol{x}_j - 0.8\boldsymbol{x}_1\sin(4\pi\boldsymbol{x}_1 + \frac{j\pi}{n}))^2$,<br>$J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$. |
| F5 | $f_1(\boldsymbol{x}) = (1+\frac{s_1}{\lvert J_1\rvert})\boldsymbol{x}_1, \quad f_2(\boldsymbol{x}) = (1+\frac{s_2}{\lvert J_2\rvert})\left(1-\sqrt{\frac{\boldsymbol{x}_1}{1+\frac{s_2}{\lvert J_2\rvert}}}\right),$<br>where $s_1 = \sum_{j\in J_1}(\boldsymbol{x}_j - 0.8\boldsymbol{x}_1\cos(\frac{4\pi\boldsymbol{x}_1 + \frac{j\pi}{n}}{3}))^2$ and $s_2 = \sum_{j\in J_2}(\boldsymbol{x}_j - 0.8\boldsymbol{x}_1\sin(4\pi\boldsymbol{x}_1 + \frac{j\pi}{n}))^2$,<br>$J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$. |
| F6 | $f_1(\boldsymbol{x}) = (1+\frac{s_1}{\lvert J_1\rvert})\boldsymbol{x}_1, \quad f_2(\boldsymbol{x}) = (1+\frac{s_2}{\lvert J_2\rvert})\left(1-\sqrt{\frac{\boldsymbol{x}_1}{1+\frac{s_2}{\lvert J_2\rvert}}}\right)$<br>where $s_1 = \sum_{j\in J_1}\{\boldsymbol{x}_j - [0.3\boldsymbol{x}_1^2\cos(12\pi\boldsymbol{x}_1 + \frac{4j\pi}{n})) + 0.6\boldsymbol{x}_1]\cos(6\pi\boldsymbol{x}_1 + \frac{j\pi}{n})\}$<br>and $s_2 = \sum_{j\in J_2}\{\boldsymbol{x}_j - [0.3\boldsymbol{x}_1^2\cos(12\pi\boldsymbol{x}_1 + \frac{4j\pi}{n})) + 0.6\boldsymbol{x}_1]\sin(6\pi\boldsymbol{x}_1 + \frac{j\pi}{n})\}$,<br>$J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$. |

### E.3 Real-world application problems

We also conduct experiments on 5 real-world multi-objective engineering design problems [76] that are initially proposed in different communities for different applications:

**Four Bar Truss Design.** This problem is to design a four-bar truss. The two objectives to optimize are its structural volume and joint displacement. The decision variables are the length of the four bars. The details of this problem can be found in Cheng and Li [10].

**Pressure Vessel Design.** This problem is to design a cylindrical pressure vessel. The two objectives to minimize are the total cost (material, forming, and welding) and the violations of three different design constraints. This problem has four decision variables, which are the shell thicknesses, the pressure vessel head, the inner radius, and the length of the cylindrical section. The details of this problem can be found in Kramer [42].

**Disk Brake Design.** This problem is to design a disc brake. The three objectives to minimize are the mass, the minimum stopping time, and the violations of four design constraints. This problem has four decision variables: the inner radius, the outer radius, the engaging force, and the number of friction surfaces. The details of this problem can be found in Ray and Liew [63].

**Gear Train Design.** This problem is to design a gear train with four gears. The three objectives to minimize are the difference between the realized gear ration and the required specification, the maximum size of four gears, and the design constraint violations. The decision variables are the numbers of teeth in each of the four gears. The details of this problem can be found in Deb and Srinivasan [17].

**Rocket Injector Design.** This problem is to design a rocket injector that needs to minimize the maximum temperature of the injector face, the distance from the inlet, and the temperature on the post tip. It has four decision variables, namely, the hydrogen flow angle, the hydrogen area, the oxygen area, and the oxidizer post tip thickness. The details of this problem can be found in Vaidyanathan et al. [78].

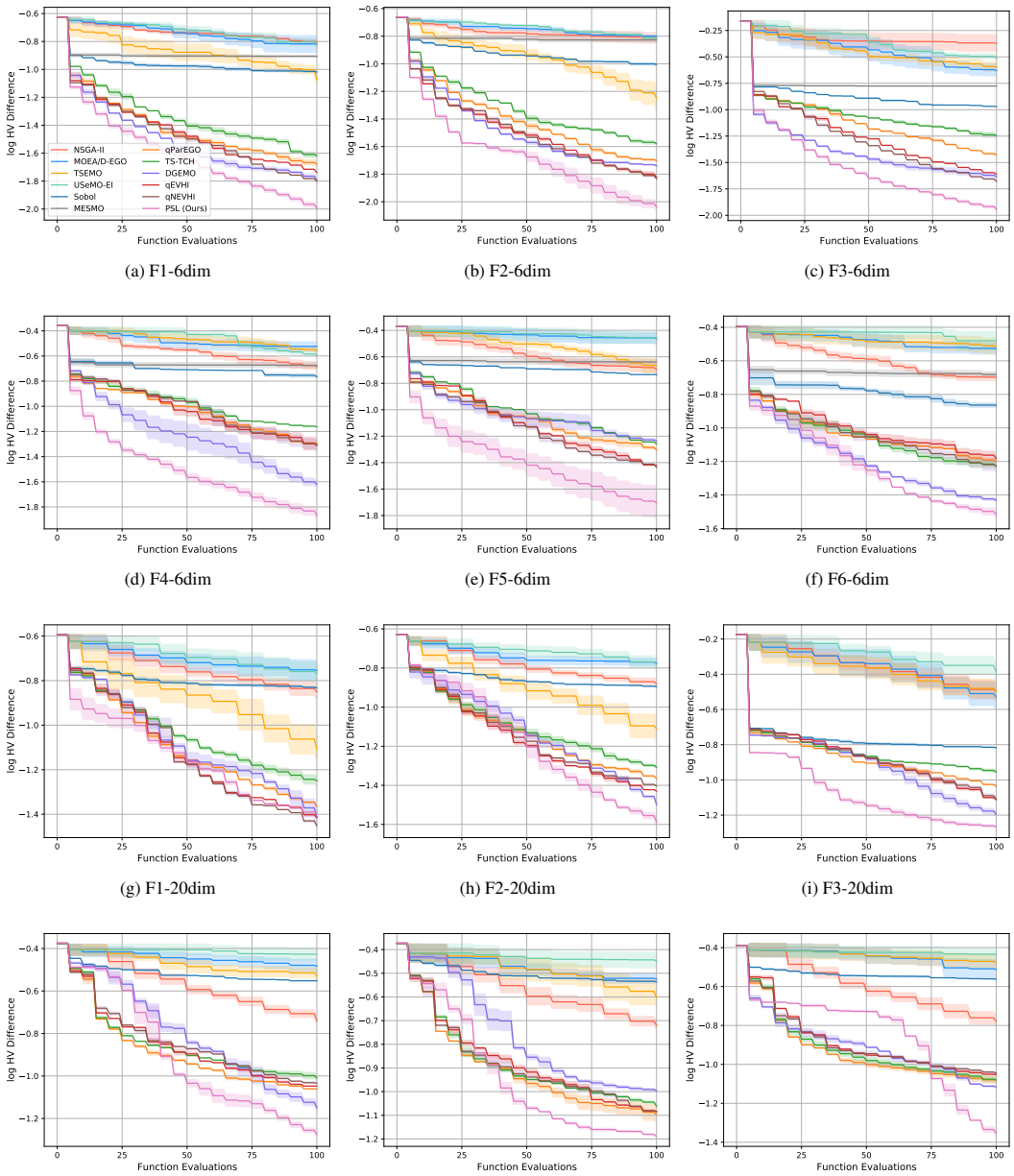
These real-world multi-objective design problems do not have known exact Pareto fronts. We use the approximate Pareto fronts provided by Tanabe and Ishibuchi [76] with a large number of evaluations as our refereed Pareto fronts, which have also been used in other MOBO works. We set the reference point $\boldsymbol{r} = 1.1 \times \hat{\boldsymbol{z}}_{\text{nadir}}$ where $\hat{\boldsymbol{z}}_{\text{nadir}}$ is the nadir point of the approximate Pareto front. The problem information and reference points can be found in Table 2.

### E.4 Experiment settings

For all experiments, we first randomly sample and evaluate $10$ valid solutions with Latin hypercube sampling [50], and set them as the initial solutions $\{\boldsymbol{X}_0, \boldsymbol{y}_0\}$. Then we further run the MOBO algorithm to optimize the given problem with a limited evaluation budget ($100$ for all problems). In the main paper, we report the results with a batch size $5$, and there are only $100/5 = 20$ batched iterations for each algorithm. In section F in this appendix, we also report experimental results with different batch sizes.

# F Additional experiments

## F.1 Results on new challenging problems F1-F6




(a) F1-6dim (b) F2-6dim (c) F3-6dim

(d) F4-6dim (e) F5-6dim (f) F6-6dim

(g) F1-20dim (h) F2-20dim (i) F3-20dim

(j) F4-20dim (k) F5-20dim (l) F6-20dim

Figure 8: The log hypervolume difference w.r.t. the number of expensive evaluation for different algorithms for F1-F6 problems with 6 and 20 dimensional search space. 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).**

We run experiments on more challenging problems proposed in Appendix E.2 to further validate our proposed PSL method. The Pareto sets of these problems are all different complicated space curves in the search space. We have run the experiments with two different settings: 1) 6-dimensional problems with 100 evaluations, and 2) 20-dimensional problems with 200 evaluations. The experimental results in Figure 8 show that our proposed PSL method still has a promising performance on these problems.

### F.2 Run time

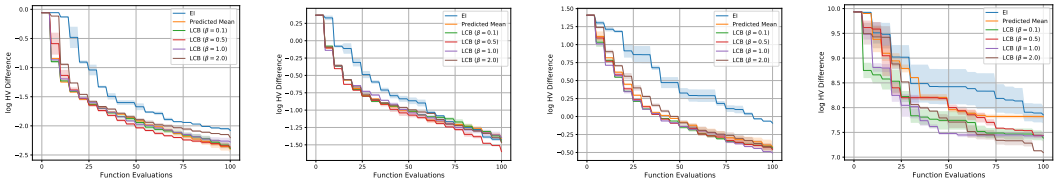
Table 3: Algorithm run time per iteration (in seconds).

| Problem | MOEA/D-EGO | TSEMO | USeMO-EI | DGEMO | qEHVI | PSL(Ours): Model + Selection |
|---|---|---|---|---|---|---|
| ZDT1 | 64.36 | 5.82 | 7.28 | 68.52 | 38.52 | 5.73 + 1.19 = 6.92 |
| ZDT2 | 58.71 | 4.93 | 6.32 | 71.52 | 42.63 | 5.32 + 1.26= 6.58 |
| ZDT3 | 62.38 | 5.28 | 5.97 | 65.17 | 45.19 | 4.98 + 1.21 = 6.19 |
| VLMOP1 | 57.13 | 4.19 | 5.23 | 60.33 | 28.37 | 3.88 + 1.06 = 4.94 |
| VLMOP2 | 61.05 | 4.96 | 6.88 | 68.20 | 39.51 | 4.62 + 1.29 = 5.91 |
| VLMOP3 | 68.72 | 8.20 | 9.03 | 79.82 | 71.25 | 5.61 + 2.49 = 8.10 |
| DTLZ2 | 71.83 | 7.28 | 8.76 | 83.57 | 75.92 | 7.02 + 1.59 = 8.61 |
| Four Bar Truss Design | 63.29 | 3.83 | 5.92 | 62.46 | 42.73 | 6.01 + 0.78 = 6.79 |
| Pressure Vessel Design | 62.61 | 4.61 | 6.59 | 69.73 | 45.32 | 4.98 + 1.33 = 6.31 |
| Disk Brake Design | 72.54 | 7.03 | 8.62 | 75.20 | 88.79 | 5.50 + 2.28 = 7.78 |
| Gear Train Design | 68.48 | 6.92 | 9.31 | 84.52 | 79.55 | 4.24 + 2.13 = 6.37 |
| Rocket Injector Design | 69.43 | 8.72 | 10.03 | 79.31 | 85.30 | 5.09 + 2.26 = 7.35 |

We report the run time for each algorithm with batch size 10 per iteration in Table 3. For our proposed PSL algorithm, we further report the detailed run time for learning the set model and for searching the batch solutions. According to the results, PSL can efficiently learn the set model and select a batch of solutions for evaluation at each iteration with a total time budget of less than 10 seconds. The short PSL batch selection run time is expected, since all solutions are directly sampled from the learned Pareto set with a few further local search steps, rather than optimizing the acquisition function(s) from scratch as in other MOBO methods.

We want to emphasize that the run time for different MOBO algorithms strongly depends on the implementation, which is also highlighted in the current works [13, 47]. The objective evaluations in real-world applications are usually very expensive and involve costly real-world experiments or simulations that take days to run. All the MOBO algorithms are efficient and have negligible run-time overheads in these real-world application scenarios.

### F.3 Performance with different acquisition functions




(a) ZDT1 (b) VLMOP1 (c) Four Bar Truss (d) Pressure Vessel

Figure 9: The log hypervolume difference w.r.t. the number of expensive evaluation for PSL with different acquisition functions.

For the proposed PSL method, We can use different acquisition functions as the surrogate value for the Pareto set learning approach. Here, we compare the performance with 6 different acquisition functions/settings, namely, the Expected Improvement (EI), Predicted Mean, and the Lower Confidence Bound (LCB) with $\beta = 0.1, 0.5, 1.0$ and $2.0$ on four benchmark and real-world application problems.

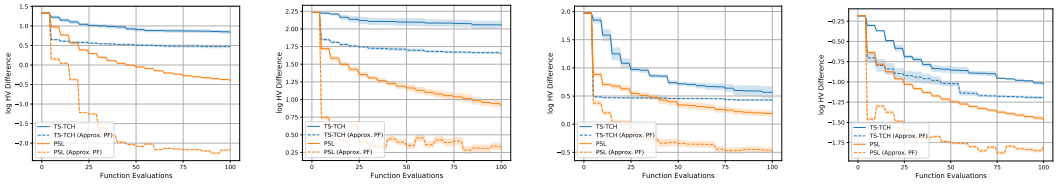
According to the results shown in Figure 9, PSL with all 6 acquisition functions has reasonably good performance. Although PSL with EI performs slightly worse than the other acquisitions, it can still outperform many other MOBO algorithms with the results shown in Figure 5 of the main paper. These results confirm that the learning the approximate Pareto set during optimization could be beneficial for MOBO.

Among the other acquisition functions and settings, no single choice can achieve the best performance for all problems. The predicted mean acquisition can be seen as LCB with $\beta = 0$, which is a pure exploitation strategy without considering the uncertainty (e.g., predicted variance) for exploration.

It might be surprising that this greedy acquisition can still have a pretty good performance. Indeed, this greedy and purely exploitative approach has been recently studied and shown to have promising performance for single-objective BO [64, 16]. DGEMO [47] also uses the predicted mean as an acquisition function for MOBO. The BO textbook [26] briefly discusses this pure exploitative strategy at the end of Section 7.10 (posterior mean acquisition function and lookahead). One limitation of this greedy approach is that it might be stuck at a non-optimal location with no well-fitted surrogate model due to overexploitation [26].

There are many acquisition policies for Bayesian optimization, but finding the most suitable one for a given problem is still an open question [26], which is also the case for our PSL method.

### F.4 Quality of the learned Pareto front



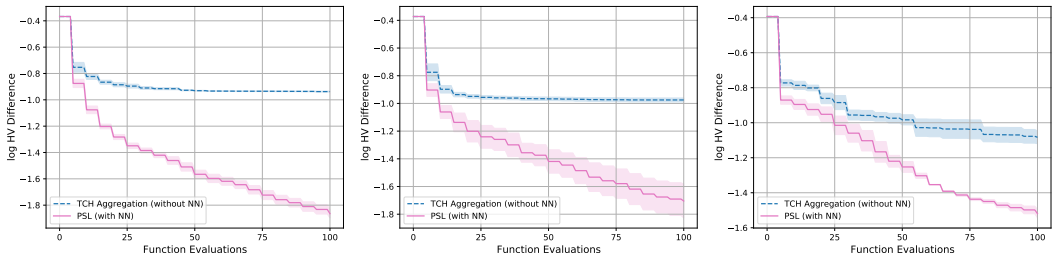

(a) Four Bar Truss (b) Disk Brake (c) Gear Train (d) Rocket Injector

Figure 10: The log hypervolume difference w.r.t. the number of expensive evaluations for PSL and TS-TCH. We report both results for the evaluated solutions (solid lines) and the approximate Pareto front (dashed line).

In the main paper, we have demonstrated that the proposed PSL method can successfully approximate the ground truth Pareto front with a limited evaluation budget as in Figure 6. In this subsection, we further investigate the performance of the learned Pareto fronts during the optimization process. As shown in Figure 10, the learned Pareto fronts have good quality (with 1,000 random samples) during optimization and lead to promising evaluated solutions performance.

In addition, we also compare our Pareto set model with another modeling approach based on evaluated solutions. Specifically, we run the TS-TCH algorithm and record the evaluated solutions with their corresponding preferences during the optimization process. Then we build a model to approximate the Pareto front based on the evaluated and non-dominated solutions at each iteration step. The model we built has an identical structure (2-layer MLP) to the PSL model, but is now trained in a supervised manner. According to the results in Figure 10, our proposed PSL model has better quality than the model trained in the supervised manner with evaluated solutions.

### F.5 Impact of the aggregation function and Pareto set model

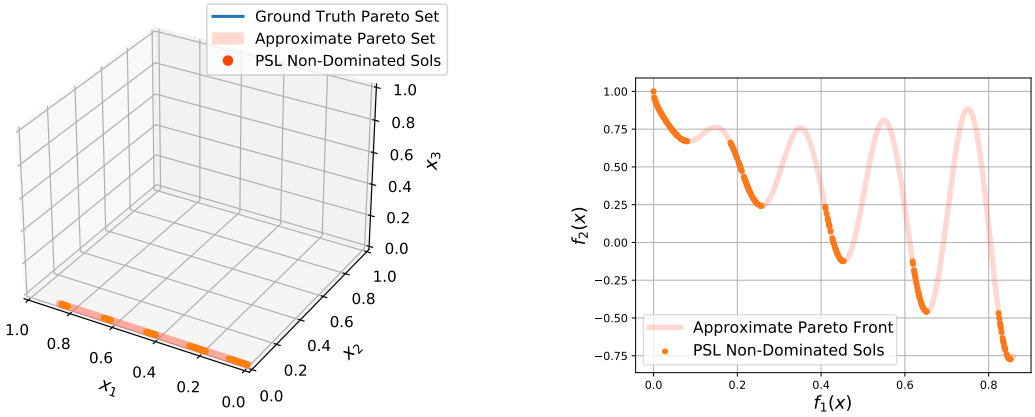



(a) F4 (b) F5 (c) F6

Figure 11: The log hypervolume difference w.r.t. the number of expensive evaluation for different algorithms for F4, F5 and F6.

In this work, we choose the weighted Chebyshev scalarization with an ideal point and epsilon mainly due to its good theoretical property. In this subsection, we conduct an ablation study on our proposed PSL method with and without the Pareto set model. The results in Figure 11 show that simply optimizing the acquisition function for scalarization could lead to a significant performance drop. It confirms that the proposed Pareto set model is important for the overall promising performance.

## F.6 PSL for disconnected Pareto set

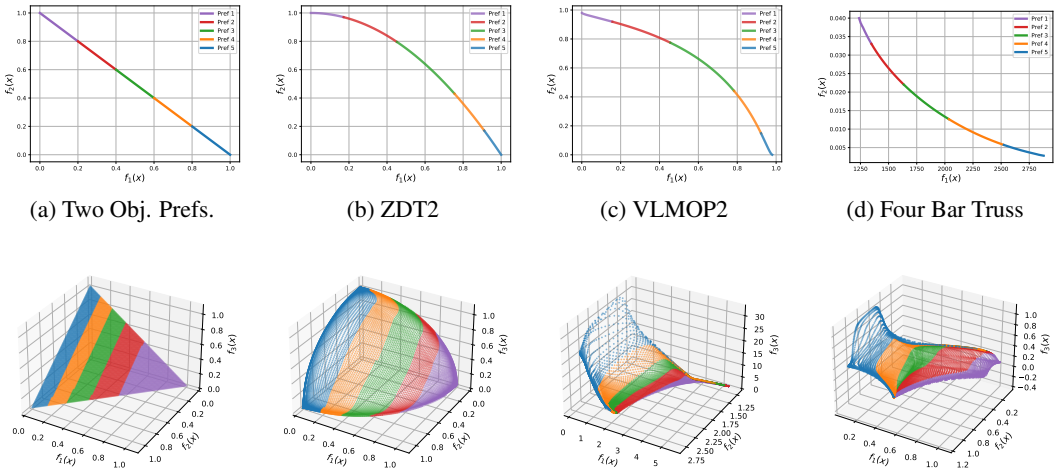



(a) Pareto Set (b) Pareto Front

Figure 12: The learned Pareto set and Pareto front for ZDT3 problem.

We show the approximate Pareto set and Pareto front learned by PSL for the ZDT3 problem in Figure. The ZDT3 problem has a continuous search space, but its Pareto set is disconnected. According to the result, PSL can find an approximate continuous set that contains the disconnected parts of the Pareto set. In this work, we use a simple fully-connected neural network as the Pareto set model, and do not take the disconnected Pareto set into consideration. How to design a more powerful Pareto set model to capture the disconnected patterns of the Pareto set (e.g., maybe with search region decomposition) is worthwhile studying in the future.

## F.7 PSL with different preferences



(a) Two Obj. Prefs. (b) ZDT2 (c) VLMOP2 (d) Four Bar Truss

(e) Three Obj. Prefs. (f) DTLZ2 (g) Disk Brake (h) Rocket Injector

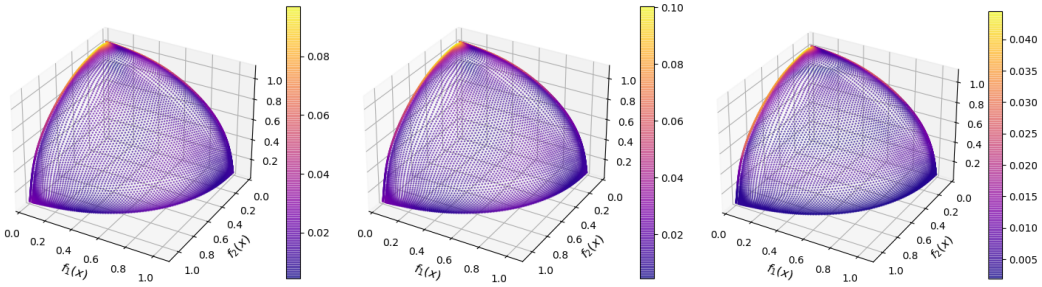
Figure 13: **Learned Pareto Set with Different Preferences:** Our proposed PSL model allows the decision-makers to flexibly explore the approximate Pareto front to obtain any trade-off solution. Suppose the decision-makers have different sets of preferred trade-offs, our model can successfully generate the corresponding parts of the Pareto fronts. The preference assignment is not fixed, and the decision-makers can adjust it and obtain the corresponding approximate solutions in real-time.

Our proposed PSL method enables the decision-makers to easily adjust their preference to explore the whole approximate Pareto set for making decisions (e.g., see Figure 3 and Figure 6 in the main paper). This ability is not supported by other MOBO methods, which is important to flexibly incorporate the decision-maker's preference and prior knowledge into Bayesian optimization [26].

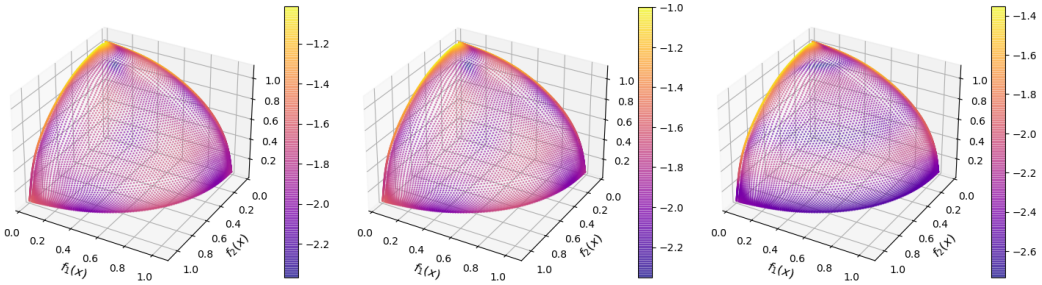
The decision-making process for real-world applications could be subjective. We follow a similar way with other preference-based MOBO to show our method can successfully generate different trade-off solutions. In Figure 13, suppose the decision-makers have different sets of trade-off preferences, our model can generate the corresponding parts of the Pareto fronts for different problems. We uniformly sample $1,000$ and $10,011$ preferences for the two-objective and three-objective problems respectively, and use the learned Pareto set models to generate the corresponding approximate Pareto solutions. We further divide the preferences into 5 exclusive parts, and label them with the corresponding solutions in different colors. As shown in Figure 13, the corresponding connection from the preferences to the solutions is clear for all problems.

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. This ability could be beneficial for the decision-makers to further adjust and assign their most accurate preferences.

### F.8 Standard deviation on the approximated Pareto set



(a) Std for Obj1. (b) Std for Obj2. (c) Std for Obj3.

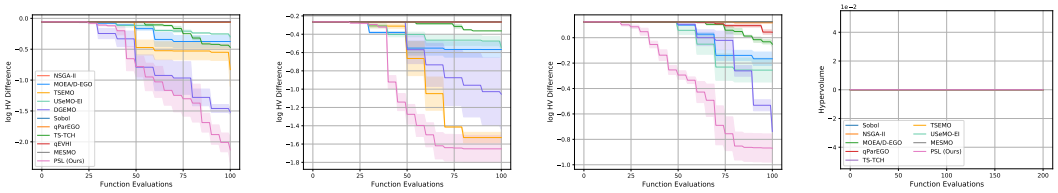


(d) Log Std for Obj1. (e) Log Std for Obj2. (f) Log Std for Obj3.

Figure 14: **Standard deviation and log standard deviation for DTLZ2:** The predicted standard deviations for different objectives on the approximated Pareto front.

In addition to the predicted mean, our PSL method also provides the uncertainty information (e.g., predicted standard deviation) to support decision-making. Figure 14 shows the (log) predicted standard deviations for different objectives on the approximate Pareto front of DTLZ2. According to the results in Figure 14(a)(b)(c), we can observe that the PSL model has a low predicted standard deviations for almost all locations on the approximate Pareto front except the top corner and the top left boundary. From the log standard deviations in Figure 14(d)(e)(f), we find that PSL has a higher overall uncertainty level for objective 1/2 than objective 3, where the uncertainty distributions are quite similar for the first two objectives.

### F.9 Problems with larger number of decision variables and objectives



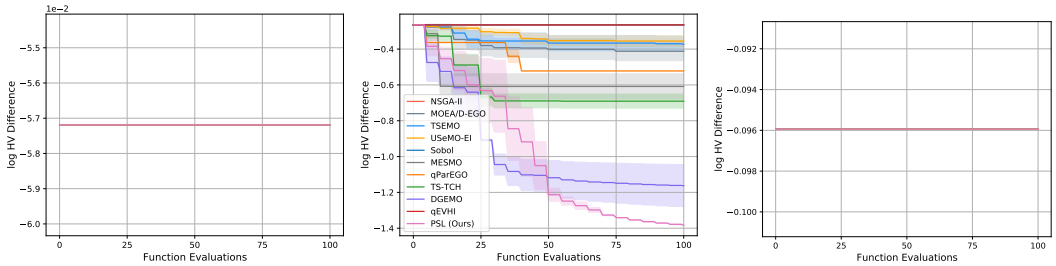

(a) ZDT1 (b) ZDT2 (c) ZDT3 (d) DTLZ2

Figure 15: The log hypervolume difference (and hypervolume) w.r.t. the number of evaluation for 2-objective ZDT1-3 with 20 decision variables and 10-objective DTLZ2 with 50 decision variables.

We also test PSL on problems with a larger number of decision variables and objectives. Figure 15 shows the results on 2-objective ZDT1-3 problems with 20 decision variables and the 10-objective DTLZ2 problem with 50 decision variables. For these problems, all algorithms have 50 randomly generated initial solutions, and we conduct 20 batched evaluations with batch size 10. Therefore, there are total 250 evaluations for each algorithm.

According to the results, PSL still performs well on ZDT problems with 20 decision variables. For the 10-objective DTLZ2 problem, all algorithms have 0 hypervolume since the obtained solutions are far from the Pareto front. We do not include the results for DEGMO and qEHVI. The DEGMO code does not support problems with larger than 3 objectives. The calculation of expected hypervolume improvement could be time-consuming, and the efficient computation methods proposed in qEHVI [13] and qNEHVI [14] make it possible to properly handle problems with 3-4 objective. However, the scalability is still a key limitation for the current MOBO methods. How to efficiently solve problems with a large number of decision variables and objectives could be an important research direction for MOBO.

### F.10 Results on other ZDT and DTLZ problems

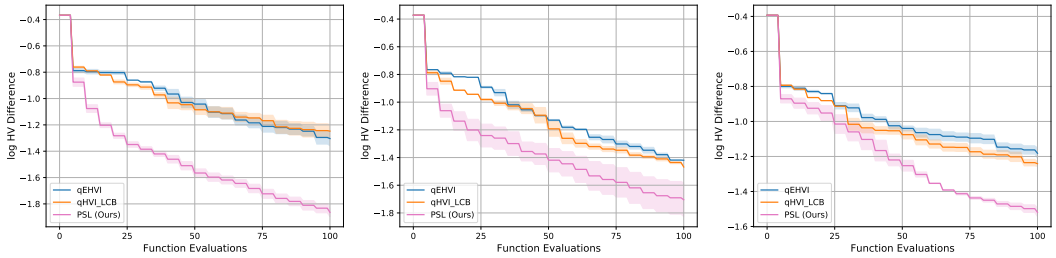



(a) ZDT4 (b) ZDT6 (c) DTLZ1

Figure 16: The log hypervolume difference w.r.t. the number of expensive evaluation for different algorithms for ZDT4, ZDT6 and DTLZ1.

We run experiments on ZDT4, ZDT6, and DTLZ1 in this subsection. The results show that the proposed PSL method can achieve promising results on the ZDT6 problem, while none of the MOBO algorithms can have satisfactory performance for ZDT4 and DTLZ1. Since the obtained solutions are all far from the ground truth Pareto front and the default reference point, all MOBO methods have a zero hypervolume value and hence lead to the same value for hypervolume difference. One possible reason for the poor performance is that the ZDT4 and DTLZ1 problems have a complicated optimization landscape (e.g., with a large number of local optimums). With a limited evaluation budget, the MOBO methods all fail to build a good surrogate model for such problems, and hence have poor performance.

### F.11 Comparisons with HVI-LCB




(a) F4 (b) F5 (c) F6

Figure 17: The log hypervolume difference w.r.t. the number of expensive evaluation for qEHVI, qHVI-LCB, and our proposed PSL on F4, F5, and F6.

In this subsection, we conduct an ablation study on maximizing the hypervolume improvement with LCB on the approximate Pareto set vs. searching across the whole design space. The experimental results on three newly proposed problems (i.e., F4, F5, and F6) with complicated Pareto sets are shown in Figure 17. We also provide the qEHVI results as reference.

Based on these results, it is clear that our proposed PSL method can outperform directly searching the entire search space to maximize HVI with LCB. The qHVI-LCB method performs similarly to qEHVI, which indicates that different acquisitions (for HVI) do not have a significant impact on the optimization performance. In other words, with the learned Pareto set, our proposed PSL method can efficiently conduct the acquisition optimization mainly on the promising low-dimensional (e.g., (m-1)) manifold, and then lead to significantly better Bayesian optimization performance.

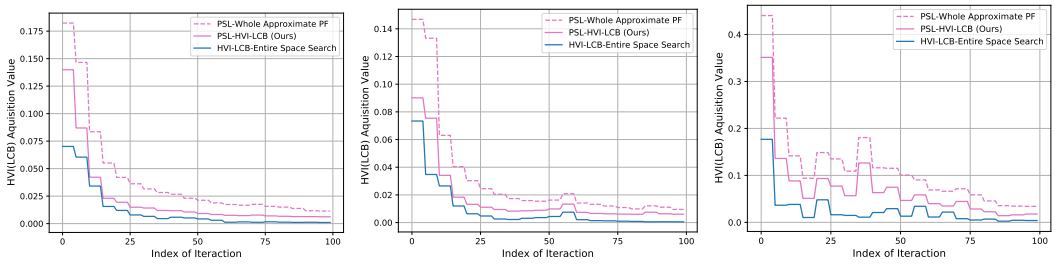
One possible reason for this performance gap could be the difficulty of directly searching the whole design search for maximizing HVI. As shown in recent works [89, 15], the performance of qEHVI can be improved using meta search region management algorithms such as LaMOO [89] and trust region methods [15]. It implies that directly searching the whole design space is not efficient.

These meta-algorithms adaptively decompose the whole search region into different subregions along with the optimization process, while our PSL method directly learns the whole approximate Pareto set at each step. We will add the comparison with those meta-algorithms in the final version (LaMOO [89] is not open-sourced yet, and the code for the trust-region method [15] was just released several days ago). Studying how to efficiently combine PSL with these algorithms could be an important future work as discussed in Appendix B.

Together with the discussion in our initial response, the advantages of PSL can be summarized from the following two viewpoints:

- From the viewpoint of scalarization based methods, PSL proposes a novel two-stage approach to select a small batch of appropriate solutions from the approximate Pareto front, which takes those already-evaluated solutions into consideration. As a result, it significantly outperforms other scalarization-based methods.
- From the viewpoint of hypervolume improvement based methods, in an ideal case, the set of solutions that optimize HVI should all be on the Pareto front. Restricting the search on the approximate Pareto set, PSL leads to an efficient HVI algorithm.

## F.12 The HVI acquisition values during optimization



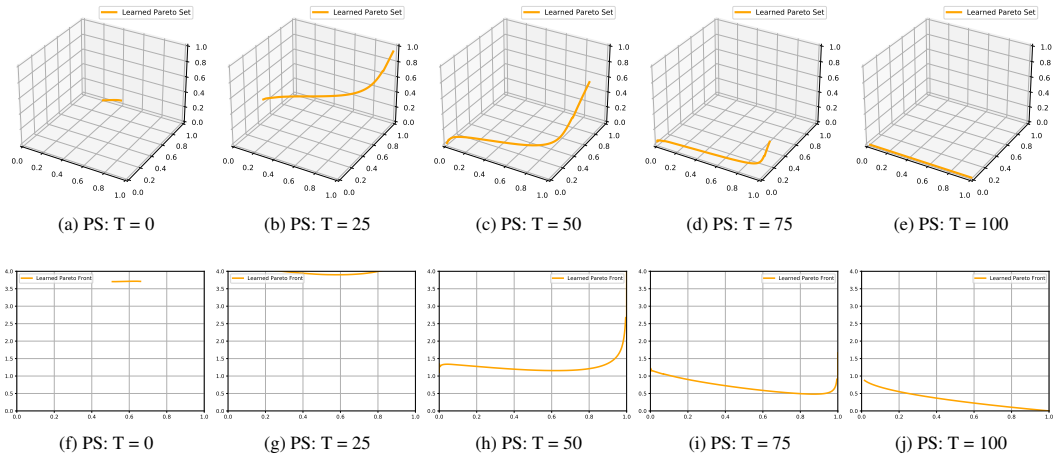

(a) F4 (b) F5 (c) F6

Figure 18: The HVI-LCB acquisition values obtained by different search methods along the PSL's optimization process on F4, F5, and F6.

Following your suggestion, we have added the experiment to compare the HVI-LCB acquisition values obtained by different search methods in Appendix F.11. We run our proposed PSL as the optimization algorithm, and conduct different search methods at each iteration. The compared search methods are: 1) searching across the entire design space, 2) our PLS search on the approximate Pareto front, and 3) the HVI-LCB acquisition values for the whole approximate Pareto front with 1,000 sampled solutions (which can be treated as the upper bound for our method). Based on the results, it is clear that our proposed PSL search can produce better HVI-LCB acquisition values than directly searching the entire design space.

It should be noticed that the reported results are obtained by running PSL, and the search methods have the same surrogate functions and baseline HV (from the same evaluated solutions chosen by PSL) at each iteration. By running qHVI-LCB alone, the inefficient acquisition search at each iteration will lead to the accumulated worse overall performance as shown in Figure.17.

## F.13 The learning process of PSL on ZDT1



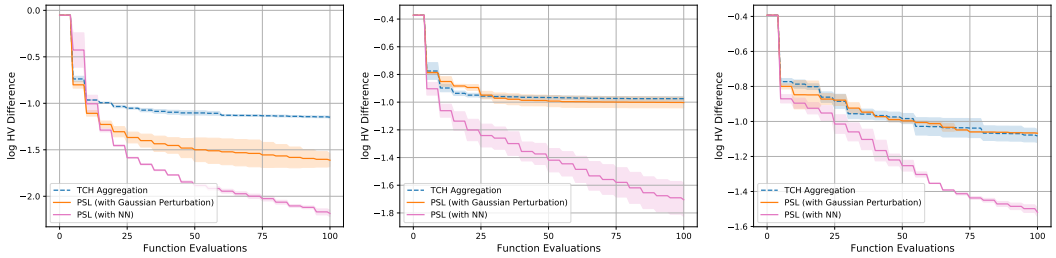

(a) PS: T = 0 (b) PS: T = 25 (c) PS: T = 50 (d) PS: T = 75 (e) PS: T = 100

(f) PS: T = 0 (g) PS: T = 25 (h) PS: T = 50 (i) PS: T = 75 (j) PS: T = 100

Figure 19: The learning process of the approximate Pareto set (upper row) and Pareto front (lower row) by PSL on ZDT1.

Due to the simple Pareto set, the optimization performance on ZDT1 could be significantly affected by the initialization. In the experiment, we did not leverage this property for an unfair comparison on the ZDT problems. PSL randomly initializes the parameters of the Pareto set model at each step, and then uses the gradient-based method to learn the approximate Pareto set. An illustration on how PSL learns the approximate Pareto set/front is shown in Figure 19. In our approach, the initially learned Pareto set is a random curve in the search space far away from the boundary. Then it gradually converges to the Pareto set with the efficient gradient-based update.

### F.14 Gaussian perturbation for generating candidate set




(a) ZDT1 (b) F5 (c) F6

Figure 20: The HVI-LCB acquisition values obtained by different search methods along the PSL's optimization process on ZDT1, F5, and F6.

In this subsection, we conduct an ablation on small perturbing the best points v.s. Pareto set model for generating the candidate set.

Based on the results shown in Figure 20, the Gaussian perturbing method can improve the performance of simple scalarization on problems with simple Pareto set (e.g., ZDT1) but not the problems with complicated Pareto set (e.g., F4-F6). On all problems, our proposed PSL method still achieves the best performance.

DGEMO [47] also has a local search approach to expand the candidate set around the best points (in term of surrogate value) with the first and second derivatives of the GP surrogate model. In our experiments, PSL can outperform DGEMO on most problems, which confirms the importance of the NN for generating the candidate set.

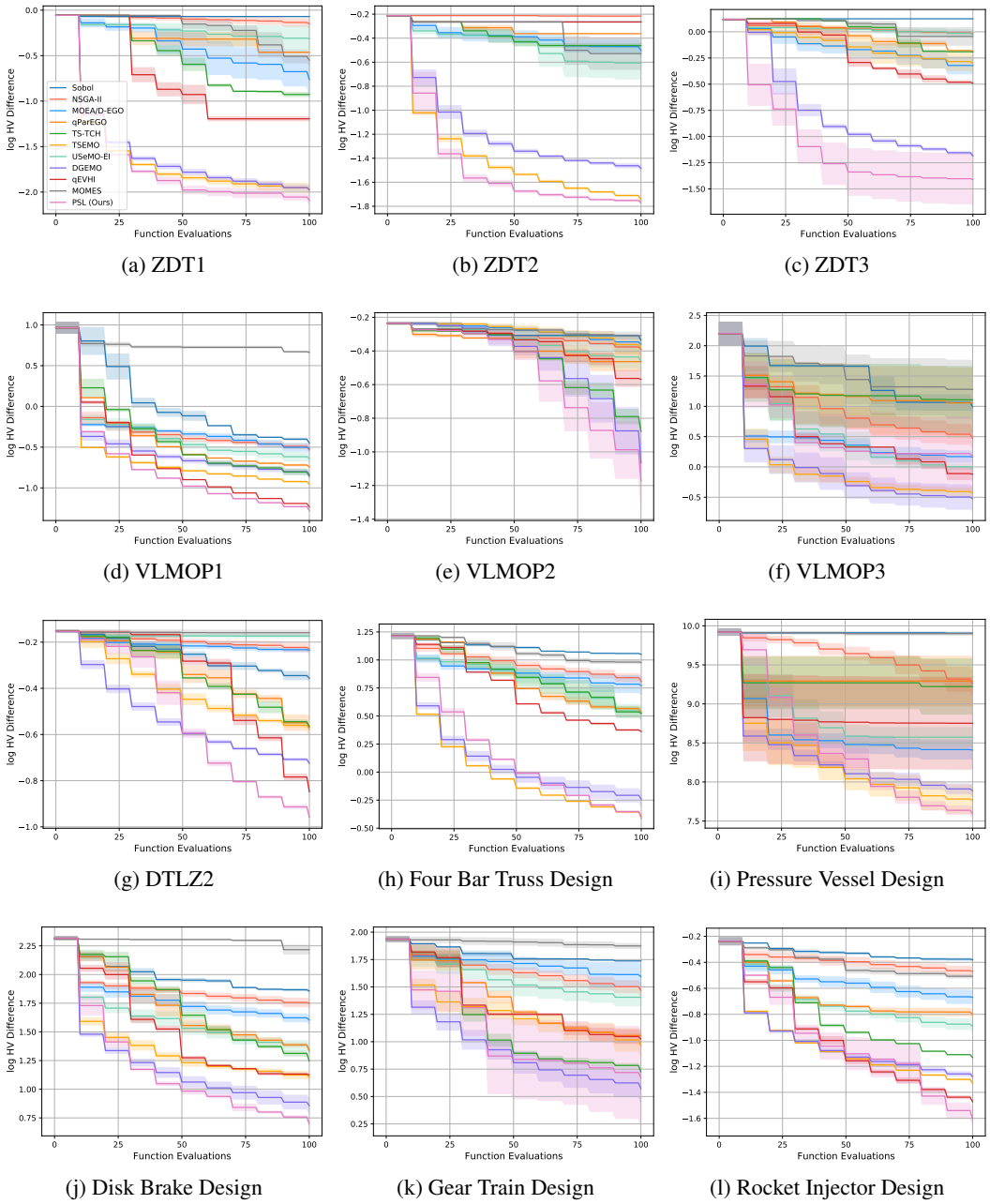
On the other hand, the local search methods might provide complementary candidate solutions to PSL, especially at the early stage of optimization when the approximate Pareto set is inaccurate. We will investigate how to efficiently combine PSL with the local search approaches in future work.

### F.15 Performance with Different Batch Sizes

We conduct additional experiments with different batch size ($B = 10$) as well as sequential evaluation ($B = 1$) for all problems. The total number of evaluations is still $10 + 100 = 110$ for all experiments. In other words, these settings have $10$ and $100$ batched evaluations, respectively. The experimental results are shown in Figure 21 and Figure 22. According to the results, our proposed PSL method consistently has the best overall performance for different batch sizes. It outperforms many other MOBO approaches on a large majority of experiments and has comparable performance with the best algorithm when it is not the best one.

In this work, we focus on the main idea of learning the Pareto set for MOBO problems, and do not aggressively include other powerful methods into our proposed algorithm. Its performance might be further improved by considering the second-order derivatives of GPs and diversity-guided batch selection in DGEMO [47], as well as Thompson sampling and uncertainty maximization in TSEMO [8] and USeMO-EI [5].

These promising algorithm performances, along with the efficient run time, confirm that it is worth building the Pareto set model for expensive multi-objective optimization problems. We believe the Pareto set learning approach could be an efficient off-the-shelf tool for multi-objective Bayesian optimization.



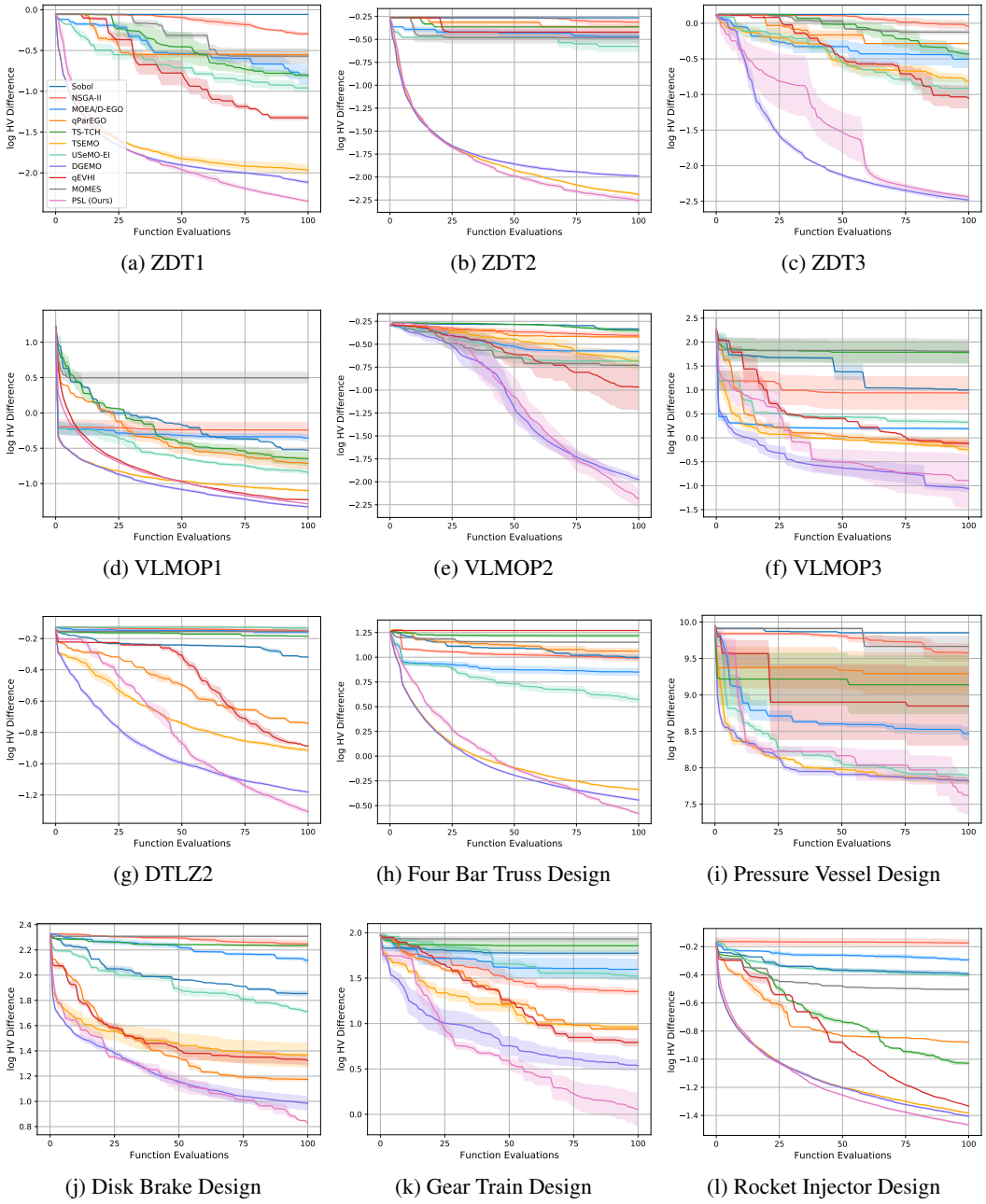

(a) ZDT1 (b) ZDT2 (c) ZDT3

(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 21: **Experimental results with Batch Size 10** on the log hypervolume difference w.r.t. the number of expensive evaluation for different algorithms. The label of each algorithm can be found in subfig(a).




(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 22: **Experimental results with Batch Size 1** on the log hypervolume difference w.r.t. the number of expensive evaluation for different algorithms. The label of each algorithm can be found in subfig(a).

# G Licenses

Table 4 lists the licenses for the codes and problem suite we used in this work.

Table 4: List of licenses for the codes and problem suite we used in this work.

| Resource | Type | Link | License |
|---|---|---|---|
| Botorch [3] | Code | https://botorch.org/ | MIT License |
| DEGMO [47] | Code | https://github.com/yunshengtian/DGEMO | MIT License |
| pymoo [6] | Code | https://pymoo.org/ | Apache License 2.0 |
| reproblems [76] | Problem Suite | https://ryojitanabe.github.io/reproblems/ | None |