Parametric Pareto Set Learning for Expensive Multi-Objective Optimization

Ji Cheng, Bo Xue, Qingfu Zhang*

Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China
The City University of Hong Kong Shenzhen Research Institute, Shenzhen, China
{J.Cheng, boxue4-c}@my.cityu.edu.hk, qingfu.zhang@cityu.edu.hk

Abstract

Parametric multi-objective optimization (PMO) addresses the challenge of solving an infinite family of multi-objective optimization problems, where optimal solutions must adapt to varying parameters. Traditional methods require re-execution for each parameter configuration, leading to prohibitive costs when objective evaluations are computationally expensive. To address this issue, we propose Parametric Pareto Set Learning with multi-objective Bayesian Optimization (PPSL-MOBO), a novel framework that learns a unified mapping from both preferences and parameters to Pareto-optimal solutions. PPSL-MOBO leverages a hypernetwork with Low-Rank Adaptation (LoRA) to efficiently capture parametric variations, while integrating Gaussian process surrogates and hypervolume-based acquisition to minimize expensive function evaluations. We demonstrate PPSL-MOBO's effectiveness on two challenging applications: multi-objective optimization with shared components, where certain design variables must be identical across solution families due to modular constraints, and dynamic multi-objective optimization, where objectives evolve over time. Unlike existing methods that cannot directly solve PMO problems in a unified manner, PPSL-MOBO learns a single model that generalizes across the entire parameter space. By enabling instant inference of Pareto sets for new parameter values without retraining, PPSL-MOBO provides an efficient solution for expensive PMO problems.

Code — https://github.com/jicheng9617/PPSL-MOBO
Extended version — https://arxiv.org/abs/2511.05815

1 Introduction

The ability to make optimal decisions in the face of uncertain or varying parameters is a cornerstone of modern optimization and engineering design (Fiacco 1976; Bank et al. 1982). *Parametric multi-objective optimization* (PMO) seeks to understand how the set of optimal trade-offs, known as the Pareto set (PS), evolves as exogenous parameters change (Milgrom and Shannon 1994; Weaver-Rosen et al. 2020). This paradigm is central to adaptive systems in science and engineering, where it is essential not only to solve for the best trade-offs under given conditions, but also to efficiently track the evolution of these trade-offs as requirements, environments, or constraints shift (Wang et al. 2000; Tsai and Malak Jr 2021).

In this paper, we consider the following PMO problem:

$$\mathcal{F}^*(\boldsymbol{t}) \subset \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left(f_1(\boldsymbol{x},\boldsymbol{t}), f_2(\boldsymbol{x},\boldsymbol{t}), \ldots, f_m(\boldsymbol{x},\boldsymbol{t})\right), \quad (1)$$

where $\boldsymbol{x}$ represents the decision variable vector in the decision space $\mathcal{X} \subset \mathbb{R}^n$, $\boldsymbol{t}$ denotes parameters in the space $\Theta \subset \mathbb{R}^p$, and $\mathcal{F}^*(\boldsymbol{t})$ defines the optimal solutions that simultaneously minimize the m objectives. Applications of PMO span diverse domains: from engineering design where operating conditions vary, to personalized medicine where treatments must adapt to patient characteristics, to autonomous systems that must respond to changing environments in real-time.

However, solving PMO problems presents fundamental challenges, particularly in *expensive* settings where each objective evaluation requires costly simulations, physical experiments, or time-consuming computations. The core challenges include: (i) *Computational intractability*: Traditional methods require re-solving the entire optimization problem for each new parameter value, making real-time adaptation impossible; (ii) *Sample efficiency*: With limited evaluation budgets, we must learn the PS across the entire parameter space without exhaustive sampling; (iii) *Generalization*: The learned model must accurately predict Pareto-optimal solutions for previously unseen parameter values; and (iv) *Exploration-exploitation balance*: We must intelligently allocate evaluations between improving known regions and exploring new parameter territories.

While recent advances in Pareto set learning (PSL) have shown promise in learning mappings from preferences to Pareto-optimal solutions for fixed problems (Navon et al. 2021; Lin et al. 2022), these methods fail to address the parametric nature of many real-world scenarios. Similarly, existing multi-objective Bayesian optimization approaches (Knowles 2006; Emmerich, Giannakoglou, and Naujoks 2006) focus on single-instance problems and cannot leverage the structural relationships across different parameter values. The lack of methods that can efficiently learn and generalize across both preference and parameter spaces represents a critical gap in the literature.

This work proposes *Parametric Pareto Set Learning with Multi-Objective Bayesian Optimization* (PPSL-MOBO), a

*Corresponding author.

novel framework that addresses these challenges through three key innovations. First, we develop a unified neural architecture that learns the continuous mapping from both preferences and parameters to Pareto-optimal solutions, enabling instant adaptation to new scenarios. Second, we integrate surrogate modeling with an information-theoretic acquisition strategy to maximize learning efficiency under tight evaluation budgets. Third, we introduce a systematic approach to balance exploration across the parameter space with exploitation of promising regions, ensuring robust performance across diverse problem instances.

Our main contributions are:

- We proposed a unified model architecture that learns parametric PSs as a continuous function of both preferences and exogenous parameters, enabling real-time multi-objective decision making without retraining.
- We developed a sample-efficient learning framework that integrates flexible surrogate models with intelligent acquisition strategies, dramatically reducing the number of expensive evaluations required.
- We applied our PPSL-MOBO method to modular design scenarios, where certain decision variables are shared across families of solutions. By modeling the PS as a function of shared components, our method enables efficient, real-time design for modular and customizable systems, as validated on real-world-inspired benchmarks.
- We extended PPSL-MOBO to dynamic multi-objective optimization, demonstrating that our approach can efficiently adapt to temporal changes by treating time as a parametric context. Extensive experiments on established dynamic benchmarks verify significant improvements in effectiveness over state-of-the-art methods.

2 Preliminaries

2.1 Multi-Objective Optimization

In the context of expensive continuous multi-objective optimization, the goal is to optimize multiple conflicting objectives simultaneously, which can be formally expressed as:

$$\min_{\boldsymbol{x}\in\mathcal{X}} \ \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})), \tag{2}$$

where $\boldsymbol{x}$ represents a solution in the decision space $\mathcal{X} \subseteq \mathbb{R}^n$. The objective function $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^m$ is vector-valued, with each $f_i(\boldsymbol{x})$ being an individual objective. Each objective $f_i(\boldsymbol{x})$ is assumed to be expensive to evaluate, meaning that function evaluations require significant computational resources or time. In practical scenarios, these objectives often conflict with each other, making it impossible to find a single solution that simultaneously optimizes all objectives.

Definition 1 (Pareto Dominance) *A solution $\boldsymbol{x}_a$ is said to dominate another solution $\boldsymbol{x}_b$, denoted as $\boldsymbol{x}_a \prec \boldsymbol{x}_b$, if and only if $f_i(\boldsymbol{x}_a) \leq f_i(\boldsymbol{x}_b)$ for $i \in \{1, \ldots, m\}$, and $\exists j \in \{1, \ldots, m\}$ such that $f_j(\boldsymbol{x}_a) < f_j(\boldsymbol{x}_b)$. Furthermore, if $f_i(\boldsymbol{x}_a) < f_i(\boldsymbol{x}_b), \forall i \in \{1, \ldots, m\}$, $\boldsymbol{x}_a$ is said to strictly dominate $\boldsymbol{x}_b$, denoted by $\boldsymbol{x}_a \prec_{\text{strict}} \boldsymbol{x}_b$.*

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

Definition 3 (Pareto Set and Pareto Front) *The set of all Pareto optimal solutions is referred to as the Pareto set, denoted by $\mathcal{M}_{\text{ps}} \subseteq \mathcal{X}$. The corresponding image of these solutions in the objective space, $\boldsymbol{f}(\mathcal{M}_{\text{ps}}) = \{\boldsymbol{f}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{M}_{\text{ps}}\}$, is termed the Pareto front. Similarly, we define the weak Pareto set $\mathcal{M}_{\text{weak}}$ and weakly Pareto front $\boldsymbol{f}(\mathcal{M}_{\text{weak}})$.*

Under appropriate regularity conditions, both the PS and Pareto front (PF) typically form $(m-1)$-dimensional manifolds in their respective spaces (Hillermeier 2001; Zhang, Zhou, and Jin 2008). Each solution in the PS represents a unique trade-off among the conflicting objectives, and the cardinality of the PS can be infinite for continuous optimization problems.

2.2 Bayesian Optimization

Due to the expensive nature of objective evaluations, traditional multi-objective optimization algorithms that require numerous function evaluations become impractical. Bayesian Optimization (BO) has emerged as a powerful paradigm for tackling such expensive optimization problems. By constructing surrogate models (typically Gaussian processes) from limited evaluation data and employing acquisition functions to guide the search process, BO can efficiently identify high-quality solutions while minimizing the number of expensive function evaluations required. This model-based approach is particularly well-suited for expensive multi-objective optimization, where each evaluation must be carefully selected to maximize the information gained about the underlying PS and front. We provide a brief introduction as follows, and interested readers can refer to (Garnett 2023).

Gaussian Process (GP) provides a probabilistic framework for modeling unknown functions in Bayesian optimization. For a single objective function, a GP is fully specified by its prior distribution over the function space:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')), \tag{3}$$

where $\mu : \mathcal{X} \to \mathbb{R}$ denotes the mean function (often assumed to be zero), and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ represents the covariance kernel function that encodes assumptions about the function's smoothness and behavior.

Given a dataset of n evaluated solutions $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)}) \mid i = 1, \ldots, n\}$ where $y^{(i)} = f(\boldsymbol{x}^{(i)})$, the GP posterior can be computed in closed form. For any new input $\boldsymbol{x}$, the posterior mean and variance are:

$$\begin{aligned} \hat{\mu}(\boldsymbol{x}) &= \mu(\boldsymbol{x}) + \boldsymbol{k}^T \boldsymbol{K}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}), \\ \hat{\sigma}^2(\boldsymbol{x}) &= k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}^T \boldsymbol{K}^{-1} \boldsymbol{k}, \end{aligned} \tag{4}$$

where $\boldsymbol{k} = [k(\boldsymbol{x}, \boldsymbol{x}^{(1)}), \ldots, k(\boldsymbol{x}, \boldsymbol{x}^{(n)})]^\top$ is the covariance vector between the new point and observed data, $\boldsymbol{K}$ is the kernel matrix of observed points with $\boldsymbol{K}_{ij} = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$, $\boldsymbol{y} = [y^{(1)}, \ldots, y^{(n)}]^\top$ contains the observed function values, and $\boldsymbol{\mu} = [\mu(\boldsymbol{x}^{(1)}), \ldots, \mu(\boldsymbol{x}^{(n)})]^\top$.

For multi-objective optimization with m objectives, independent GP models are typically constructed for each objective f_i, yielding posterior means $\hat{\boldsymbol{\mu}}(\boldsymbol{x}) = [\hat{\mu}_1(\boldsymbol{x}), \ldots, \hat{\mu}_m(\boldsymbol{x})]^\top$ and variances $\hat{\boldsymbol{\sigma}}^2(\boldsymbol{x}) = [\hat{\sigma}_1^2(\boldsymbol{x}), \ldots, \hat{\sigma}_m^2(\boldsymbol{x})]^\top$.

The predictive distribution provided by the Gaussian Process, specifically the posterior mean and variance, forms the foundation for intelligently selecting the next evaluation point in BO. The acquisition function $\alpha : \mathcal{X} \to \mathbb{R}$ serves as a utility measure that guides the selection of the next evaluation point by balancing exploration (sampling uncertain regions) and exploitation (sampling promising regions). The next evaluation point is chosen by optimizing the acquisition function:

$$\boldsymbol{x}^{(n+1)} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \alpha(\boldsymbol{x}; \mathcal{D}). \tag{5}$$

In multi-objective settings, scalar acquisition functions must be extended to handle vector-valued objectives. A particularly effective approach is based on hypervolume improvement (HVI), which measures the increase in dominated volume when adding new solutions. Given a current PF approximation $\mathcal{Y}$ and a set of candidate solutions $\mathcal{X}_+ = \{\boldsymbol{x}^{(i)}\}_{i=1}^b$ with predicted objectives $\hat{\mathcal{Y}}_+ = \{\hat{\boldsymbol{f}}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{X}_+\}$, the hypervolume improvement is:

$$\mathrm{HVI}(\hat{\mathcal{Y}}_+, \mathcal{Y}) = \mathrm{HV}(\hat{\mathcal{Y}}_+ \cup \mathcal{Y}) - \mathrm{HV}(\mathcal{Y}), \tag{6}$$

where $\mathrm{HV}(\cdot)$ denotes the hypervolume indicator computed with respect to a reference point. The next batch of solutions is selected by maximizing this improvement:

$$\mathcal{X}_+ = \arg\max_{\mathcal{X}_+ \subset \mathcal{X}} \mathrm{HVI}(\hat{\mathcal{Y}}_+, \mathcal{Y}). \tag{7}$$

2.3 Pareto Set Learning

Pareto Set Learning, also known as PSL (Lin et al. 2020; Navon et al. 2021; Lin, Yang, and Zhang 2022; Lin et al. 2022), offers a novel paradigm for multi-objective optimization by learning a single parametric model that represents the entire PS. Instead of searching for individual Pareto optimal solutions, PSL aims to capture the continuous mapping from preference vectors to corresponding optimal solutions.

The core idea of PSL is to model the PS as a parametric function:

$$\boldsymbol{x} = h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}), \tag{8}$$

where $\boldsymbol{\lambda} \in \Delta^{m-1}$ represents a preference vector on the $(m-1)$-simplex $\Delta^{m-1} = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m \mid \sum_{i=1}^m \lambda_i = 1\}$, and $h_{\boldsymbol{\theta}} : \Delta^{m-1} \to \mathcal{X}$ is a learnable model parameterized by $\boldsymbol{\theta}$. Each preference vector $\boldsymbol{\lambda}$ encodes a specific trade-off among objectives, and the model $h_{\boldsymbol{\theta}}$ maps these preferences to their corresponding Pareto optimal solutions.

The objective of PSL is to find optimal parameters $\boldsymbol{\theta}^*$ such that for any preference $\boldsymbol{\lambda}$, the output $\boldsymbol{x} = h_{\boldsymbol{\theta}^*}(\boldsymbol{\lambda})$ minimizes an aggregation function:

$$\min_{\boldsymbol{x} \in \mathcal{X}} \; l_{\text{agg}}(\boldsymbol{x} \mid \boldsymbol{\lambda}), \quad \forall \boldsymbol{\lambda} \in \Delta^{m-1}, \tag{9}$$

where $l_{\text{agg}} : \mathcal{X} \times \Delta^{m-1} \to \mathbb{R}$ is a scalarization function that combines multiple objectives into a single value based on the preference vector. This formulation naturally extends decomposition-based approaches (Zhang and Li 2007) by replacing finite populations with a continuous parametric representation.

Given a distribution $P_{\boldsymbol{\lambda}}$ over preferences, the PSL optimization problem becomes:

$$\min_{\boldsymbol{\theta}} \; \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\lambda} \sim P_{\boldsymbol{\lambda}}} \left[l_{\text{agg}}(h_{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \mid \boldsymbol{\lambda}) \right]. \tag{10}$$

The choice of aggregation function l_{agg} is crucial for the quality and coverage of the learned PS. The simplest approach is the weighted sum method, however, this linear scalarization can only identify solutions on the convex hull of the PF, missing important non-convex regions (Boyd and Vandenberghe 2004; Ehrgott 2005).

To overcome this limitation, the Tchebycheff (TCH) scalarization provides superior theoretical guarantees (Miettinen 1999):

$$l_{\text{tch}}(\boldsymbol{x} \mid \boldsymbol{\lambda}) = \max_{i \in [m]} \left\{ \lambda_i (f_i(\boldsymbol{x}) - (z_i^* - \varepsilon)) \right\}, \tag{11}$$

where $z_i^* < \min_{\boldsymbol{x} \in \mathcal{X}} f_i(\boldsymbol{x})$ represents the ideal value for the i-th objective, and $\varepsilon > 0$ is a small constant ensuring numerical stability.

Theorem 1 (Choo and Atkins (1983)) *A feasible solution $\boldsymbol{x} \in \mathcal{X}$ is weakly Pareto optimal if and only if there exists a valid preference vector $\boldsymbol{\lambda} \in \Delta^{m-1}$ such that $\boldsymbol{x}$ is an optimal solution of the Tchebycheff scalarization (11).*

This theorem establishes that TCH scalarization can recover all weakly Pareto optimal solutions, making it a theoretically sound choice for PSL. Recently, the smooth Tchebycheff (STCH) aggregation (Lin et al. 2024) has been proposed to address computational challenges of the non-smooth max operator:

$$l_{\text{stch}}(\boldsymbol{x} \mid \boldsymbol{\lambda}, \nu) = \nu \log \left(\sum_{j=1}^m e^{\left(\frac{\lambda_j (f_j(\boldsymbol{x}) - (z_j^* - \varepsilon))}{\nu} \right)} \right), \tag{12}$$

where $\nu > 0$ is a smoothing parameter. For brevity, we denote $l_{\text{stch}}(\boldsymbol{x} \mid \boldsymbol{\lambda})$ when using a fixed smoothing parameter. This smooth approximation preserves the desirable properties of TCH while offering significant computational advantages, including improved convergence rates and reduced per-iteration complexity due to its differentiability.

The theoretical foundation of STCH is particularly strong, as it maintains the ability to recover the entire PS under appropriate conditions:

Theorem 2 (Lin et al. (2024)) *There exists a smoothing parameter ν^* such that for any $0 < \nu < \nu^*$, every Pareto solution of the multi-objective optimization problem corresponds to an optimal solution of the STCH aggregation with some valid preference vector $\boldsymbol{\lambda} \in \Delta^{m-1}$.*

The smooth nature of STCH not only facilitates gradient-based optimization but also enhances numerical stability, making it particularly well-suited for integration with modern machine learning frameworks. In this work, we adopt STCH as our primary aggregation method, though the proposed framework is sufficiently general to accommodate other scalarization approaches. This flexibility ensures that our methodology can be adapted to various problem characteristics and computational constraints in expensive multi-objective optimization scenarios.

3 Parametric Pareto Set Learning for MOBO

The proposed PPSL-MOBO framework operates through three tightly coupled components: (1) the hypernetwork-LoRA architecture that generates task-specific PS models, (2) a Gaussian process-based training scheme that leverages surrogate models for scalable optimization, and (3) an intelligent data acquisition strategy that maximizes hypervolume improvement in the parametric space. As illustrated in Figure 1, the system forms a closed-loop where newly acquired data continuously refines both the surrogate models and the parametric PS representation. The complete algorithmic workflow can be found in the extended version.

3.1 Model Architecture

Our approach builds upon recent work in PSL (Lin et al. 2022), which utilizes a neural network, $h_{\boldsymbol{\theta}_{\text{ps}}}$, to model the mapping from a preference vector $\boldsymbol{\lambda}$ to a corresponding solution $\boldsymbol{x}$ on the PS:

$$\boldsymbol{x} = h_{\boldsymbol{\theta}_{\text{ps}}}(\boldsymbol{\lambda}). \tag{13}$$

A key innovation of our work is to extend this model to handle parametric objectives, where the PS itself is a function of an external context or task parameter $\boldsymbol{t} \in \mathcal{T}$.

To achieve this, a straightforward approach would be to employ a hypernetwork, $g_{\boldsymbol{\theta}_{\text{hn}}}$, that directly generates the entire set of weights $\boldsymbol{\theta}_{\text{ps}}$ for the PS model based on the input parameter $\boldsymbol{t}$:

$$\boldsymbol{\theta}_{\text{ps}} = g_{\boldsymbol{\theta}_{\text{hn}}}(\boldsymbol{t}), \tag{14}$$

where $\boldsymbol{\theta}_{\text{hn}}$ are the learnable weights of the hypernetwork. However, this naive formulation presents a significant challenge. The task parameter $\boldsymbol{t}$ is typically low-dimensional, while the PS model's weights, $\boldsymbol{\theta}_{\text{ps}}$, can be exceptionally high-dimensional (often numbering in the millions). This vast dimensionality gap makes the hypernetwork difficult to train, prone to overfitting, and unlikely to generalize well across the parameter space $\mathcal{T}$.

To overcome these challenges, we introduce a more sophisticated and parameter-efficient architecture based on Low-Rank Adaptation (LoRA). The central hypothesis is that the PSs for different task parameters $\boldsymbol{t}$ share a significant underlying structure. The LoRA framework elegantly captures this assumption by decomposing the PS model's weights into a large, shared base component and a small, task-specific, low-rank adaptation.

Formally, for each layer ℓ of the PS model, we express its weights $\boldsymbol{\theta}_{\text{ps}}^{\ell}$ as the sum of shared base weights $\boldsymbol{\theta}_0^{\ell}$ and a low-rank matrix product $\boldsymbol{B}^{\ell}(\boldsymbol{t})\boldsymbol{A}^{\ell}(\boldsymbol{t})$:

$$\boldsymbol{\theta}_{\text{ps}}^{\ell}(\boldsymbol{t}) = \boldsymbol{\theta}_0^{\ell} + \boldsymbol{B}^{\ell}(\boldsymbol{t})\boldsymbol{A}^{\ell}(\boldsymbol{t}). \tag{15}$$

Here, $\boldsymbol{\theta}_0^{\ell}$ is a trainable base parameter set shared across all tasks. The matrices $\boldsymbol{B}^{\ell}(\boldsymbol{t}) \in \mathbb{R}^{d^{\ell} \times r}$ and $\boldsymbol{A}^{\ell}(\boldsymbol{t}) \in \mathbb{R}^{r \times k^{\ell}}$ are the low-rank adapters, where the rank r is a small integer (i.e., $r \ll \min(d^{\ell}, k^{\ell})$).

Under this framework, the role of the hypernetwork is no longer to generate the full weights but only the compact low-rank matrices for all layers. Let $\boldsymbol{\theta}_{\text{lora}}(\boldsymbol{t})$ represent the collection of all entries in $\boldsymbol{A}^{\ell}(\boldsymbol{t})$ and $\boldsymbol{B}^{\ell}(\boldsymbol{t})$ across all layers. The hypernetwork task is now redefined as:

$$\boldsymbol{\theta}_{\text{lora}}(\boldsymbol{t}) = g_{\boldsymbol{\theta}_{\text{hn}}}(\boldsymbol{t}). \tag{16}$$

This approach dramatically reduces the output dimensionality of the hypernetwork. For a layer with original weight dimensions $d^{\ell} \times k^{\ell}$, the hypernetwork now only needs to predict $r(d^{\ell} + k^{\ell})$ parameters instead of $d^{\ell}k^{\ell}$. This reduction not only enhances training efficiency and stability but also provides a powerful inductive bias, encouraging the model to learn a shared base structure of the PS manifold while capturing task-specific variations through low-rank modulations.

3.2 Training Framework with Gaussian Process

Our goal is to train the shared base weights $\boldsymbol{\theta}_0$ and the hypernetwork weights $\boldsymbol{\theta}_{\text{hn}}$ such that the model's output, $\boldsymbol{x} = h_{\boldsymbol{\theta}_{\text{ps}}(\boldsymbol{t})}(\boldsymbol{\lambda})$, is a Pareto-optimal solution for any given task parameter $\boldsymbol{t} \in \mathcal{T}$ and preference vector $\boldsymbol{\lambda} \in \Delta^{m-1}$.

Drawing upon the theoretical guarantees of the STCH scalarization (Theorems 1 and 2), this goal can be formulated as finding the optimal solution to the following problem for all valid inputs:

$$\min_{\boldsymbol{x} \in \mathcal{X}} \; l_{\text{stch}}(\boldsymbol{x} \mid \boldsymbol{t}, \boldsymbol{\lambda}), \quad \forall \boldsymbol{\lambda} \in \Delta^{m-1}, \boldsymbol{t} \in \mathcal{T}. \tag{17}$$

To operationalize this, we frame the learning problem as minimizing the expected STCH loss over distributions of tasks and preferences. Let $P_{\boldsymbol{t}}$ be a distribution over the task parameter space $\mathcal{T}$, and let $P_{\boldsymbol{\lambda}}$ be a distribution over the preference simplex Δ^{m-1} (typically uniform to ensure full coverage). The global training objective is to minimize:

$$\min_{\boldsymbol{\theta}_{\text{hn}}, \boldsymbol{\theta}_0} \mathcal{L}(\boldsymbol{\theta}_{\text{hn}}, \boldsymbol{\theta}_0) = \mathbb{E}_{\boldsymbol{t} \sim P_{\boldsymbol{t}}, \boldsymbol{\lambda} \sim P_{\boldsymbol{\lambda}}} \left[l_{\text{stch}} \left(h_{\boldsymbol{\theta}_{\text{ps}}(\boldsymbol{t})}(\boldsymbol{\lambda}) \mid \boldsymbol{\lambda}, \boldsymbol{t} \right) \right]. \tag{18}$$

However, in the context of expensive multi-objective optimization, the true objective functions $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{t})$ are not available for direct, large-scale gradient-based optimization. Therefore, we must rely on a surrogate model to approximate this loss function, which naturally leads us to integrate our PPSL framework with BO.

To make the surrogate models aware of the task parameter $\boldsymbol{t}$, we adopt a direct and effective input augmentation strategy. We define an augmented input space $\mathcal{Z} = \mathcal{X} \times \mathcal{T}$, where each point $\boldsymbol{z}$ is a concatenation of the decision variable $\boldsymbol{x}$ and the task parameter $\boldsymbol{t}$, i.e., $\boldsymbol{z} = [\boldsymbol{x}, \boldsymbol{t}]$. We then construct an independent GP model for each of the m objective functions over this augmented space. For each objective f_i, the GP is defined as:

$$f_i(\boldsymbol{z}) \sim \mathcal{GP}(\mu_i(\boldsymbol{z}), k_i(\boldsymbol{z}, \boldsymbol{z}')). \tag{19}$$

This approach allows the kernel function k_i to directly model the influence of the task parameter $\boldsymbol{t}$ on the objective function, as well as any interaction effects between $\boldsymbol{x}$ and $\boldsymbol{t}$. The GP model can learn different lengthscales for the components of $\boldsymbol{x}$ and $\boldsymbol{t}$, effectively learning how sensitive the objective function is to changes in each part of the input.

The GP hyperparameters (lengthscales, variance) are optimized by maximizing the marginal log-likelihood on the full set of observed data $\mathcal{D} = \{(\boldsymbol{x}_j, \boldsymbol{t}_j, \boldsymbol{y}_j)\}_{j=1}^{N}$. Given a

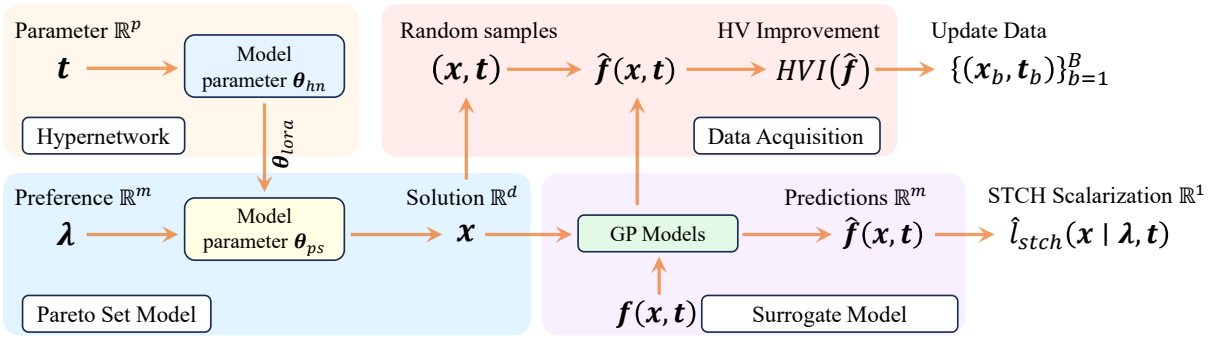

Figure 1: The PPSL-MOBO Framework. A hypernetwork adapts a PS model to a task parameter $\boldsymbol{t}$. The PS model generates approximate Pareto optimal solution $\boldsymbol{x}$ from preference $\boldsymbol{\lambda}$. The entire model is trained using predictions from surrogate models. A Bayesian optimization loop acquires new data by selecting PS-generated candidates that maximize HVI, which in turn refines the surrogate model.

new augmented point $\boldsymbol{z} = [\boldsymbol{x}, \boldsymbol{t}]$, the posterior mean $\hat{\mu}_i(\boldsymbol{z})$ and variance $\hat{\sigma}_i^2(\boldsymbol{z})$ are computed in the standard way.

With the augmented-space GP models in place, we can define a tractable surrogate for the learning objective. To balance exploitation and exploration, we use the Lower Confidence Bound (LCB) as the acquisition signal. The surrogate objective vector $\hat{\boldsymbol{f}}$ for a given pair $(\boldsymbol{x}, \boldsymbol{t})$ is defined as:

$$\hat{\boldsymbol{f}}(\boldsymbol{x}; \boldsymbol{t}) = \hat{\boldsymbol{\mu}}(\boldsymbol{z}) - \beta \hat{\boldsymbol{\sigma}}(\boldsymbol{z}), \tag{20}$$

where β is a hyperparameter controlling the exploration-exploitation trade-off, and $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}$ are the posterior mean and standard deviation from our GPs, respectively.

By substituting the true objectives with this LCB acquisition signal, we arrive at the final, differentiable surrogate loss function that we can optimize via gradient descent:

$$\hat{\mathcal{L}}(\boldsymbol{\theta}_{\text{hn}}, \boldsymbol{\theta}_0) = \mathbb{E}_{\boldsymbol{t} \sim P_{\boldsymbol{t}}, \boldsymbol{\lambda} \sim P_{\boldsymbol{\lambda}}} \left[\hat{l}_{\text{stch}} \left(h_{\boldsymbol{\theta}_{\text{ps}}(\boldsymbol{t})}(\boldsymbol{\lambda}) \mid \boldsymbol{\lambda}, \boldsymbol{t} \right) \right], \tag{21}$$

where $\hat{l}_{\text{stch}}$ is the surrogate STCH loss for a given task $\boldsymbol{t}$ and preference $\boldsymbol{\lambda}$, defined as:

$$\hat{l}_{\text{stch}}(\boldsymbol{x} \mid \boldsymbol{\lambda}, \boldsymbol{t}) = \nu \log \left(\sum_{j=1}^{m} e^{\left(\frac{\lambda_j (\hat{f}_j(\boldsymbol{x}; \boldsymbol{t}) - (z_j^* - \varepsilon))}{\nu} \right)} \right). \tag{22}$$

The expectation in Eq. (21) is approximated using Monte Carlo sampling. The gradients are backpropagated through the surrogate STCH loss, the LCB computation, and the parametric PS model to update the shared base weights $\boldsymbol{\theta}_0$ and the PS hypernetwork weights $\boldsymbol{\theta}_{\text{hn}}$.

3.3 Intelligent Data Acquisition in Parametric Space

The final component of our PPSL-MOBO framework is the data acquisition strategy. At each iteration, we select a small batch of new task-solution pairs $(\boldsymbol{x}, \boldsymbol{t})$ to evaluate.

Generating a Candidate Pool from the Parametric Manifold. First, we leverage our trained parametric PS model, $h_{\boldsymbol{\theta}_{\text{ps}}(\boldsymbol{t})}(\boldsymbol{\lambda})$, to generate a large pool of P high-quality candidate points. We sample task-preference pairs $\{(\boldsymbol{t}_p, \boldsymbol{\lambda}_p)\}_{p=1}^{P}$ and generate corresponding solutions $\boldsymbol{x}_p = h_{\boldsymbol{\theta}_{\text{ps}}(\boldsymbol{t}_p)}(\boldsymbol{\lambda}_p)$, forming a candidate pool $\mathcal{C} = \{(\boldsymbol{x}_p, \boldsymbol{t}_p)\}_{p=1}^{P}$.

Batched Selection via Hypervolume Improvement. From the candidate pool $\mathcal{C}$, we select a batch of B points for expensive evaluation using a sequential greedy strategy based on HVI (Eq. (6)). We iteratively select the point from the candidate pool that offers the largest marginal HVI with respect to the current batch and the archive of already evaluated points $\mathcal{D}$. The HVI is calculated using the LCB value (Eq. (20)) from our augmented-space GP surrogates.

4 Application Study: Multi-Objective Optimization with Shared Components

In this section, we adapt our proposed PPSL-MOBO framework to a practical multi-objective design problem: optimization with shared components. Details can be found in the full version.

4.1 Problem Introduction

In many real-world applications, structural constraints are applied to design variables due to manufacturing limitations, modular design principles, or computational budgets (Guha and Deb 2024; Lin et al. 2025; Zhao et al. 2025). For example, in personalized manufacturing, a key challenge is to generate customized products that cater to individual preferences while maintaining cost efficiency through shared components. Each customer's preference defines a specific trade-off, and the corresponding personalized design can be viewed as a Pareto optimal solution. To minimize manufacturing costs, it is essential that these designs share common modules, ensuring both flexibility and economic feasibility (Garcia and Trinh 2019).

Formally, let $\boldsymbol{x}_{\boldsymbol{s}}$ denote the subvector of decision variables $\boldsymbol{x}$ that holds the shared components, where $\boldsymbol{s} \subset \{1, \ldots, n\}$. Let $\boldsymbol{x}_{\boldsymbol{p}}$ be the remaining variables to be optimized, with $\boldsymbol{p} = \{1, \ldots, n\} \setminus \boldsymbol{s}$. The MOP with shared components can be expressed as finding the PS for a fixed set of shared values $\boldsymbol{\beta}$:

$$\mathcal{F}(\boldsymbol{\beta}) \subset \min_{\boldsymbol{x}_{\boldsymbol{p}}} \boldsymbol{F}(\boldsymbol{x}_{\boldsymbol{p}} | \boldsymbol{x}_{\boldsymbol{s}} = \boldsymbol{\beta}). \tag{23}$$

Problem	Shared Comp.	NSGA-II	qParEGO	qEHVI	PSL-MOBO	PPSL-MOBO
RE21	(x_1)	6.52e-01 (3.57e-03)	6.96e-01 (2.89e-02)	7.32e-01 (1.70e-04)	**7.34e-01** (5.79e-05)	7.33e-01 (2.31e-03)
	(x_2)	6.90e-01 (1.04e-02)	7.53e-01 (5.45e-03)	7.82e-01 (5.59e-05)	**7.84e-01** (3.43e-05)	7.79e-01 (7.91e-03)
	(x_3)	6.14e-01 (2.74e-03)	6.53e-01 (6.11e-03)	6.72e-01 (1.75e-04)	**6.76e-01** (8.41e-05)	6.68e-01 (7.07e-03)
	(x_4)	7.02e-01 (2.24e-03)	7.70e-01 (1.15e-02)	7.99e-01 (2.22e-04)	**8.02e-01** (4.01e-05)	7.99e-01 (2.74e-03)
	(x_1, x_2)	5.92e-01 (5.35e-03)	6.19e-01 (1.77e-03)	**6.23e-01** (1.05e-05)	**6.23e-01** (1.64e-05)	**6.23e-01** (7.84e-04)
	(x_2, x_3)	5.38e-01 (1.38e-03)	5.58e-01 (4.16e-03)	5.66e-01 (3.53e-05)	5.67e-01 (7.38e-05)	**5.69e-01** (6.26e-05)
	(x_3, x_4)	5.36e-01 (1.81e-03)	5.60e-01 (9.90e-04)	5.64e-01 (2.06e-05)	5.65e-01 (1.86e-04)	**5.66e-01** (7.15e-04)
	(x_1, x_2, x_3)	3.96e-01 (1.52e-04)	3.98e-01 (1.21e-03)	**4.00e-01** (5.39e-06)	3.99e-01 (7.76e-05)	**4.00e-01** (6.61e-06)
	(x_2, x_3, x_4)	5.11e-01 (1.43e-03)	5.15e-01 (3.12e-03)	5.18e-01 (1.40e-05)	5.18e-01 (7.88e-05)	**5.19e-01** (2.19e-05)

Table 1: HV values of PPSL-MOBO and baseline approaches on MOPs with shared components.All baseline methods were re-executed with a budget of 100 evaluations per task across the ten tested problems, resulting 1,000 evaluations for each share component configuration. In contrast, PPSL-MOBO was trained only once, using a total of 200 evaluations, and subsequently inferred the PS for each parameterized task.

Problem	qParEGO	qEHVI	PSL-MOBO	PPSL-MOBO Training	PPSL-MOBO Inference
RE21	317.1s	551.3s	1761.3s	364.4s	0.019s
RE33	430.9s	1038.9s	1812.4s	581.8s	0.040s
RE37	349.2s	873.1s	2668.9s	639.0s	0.028s

Table 2: Run time comparison (in seconds) of PPSL-MOBO and baseline model-based methods. All methods are run on the same CPU.

In this context, the shared component values $\boldsymbol{\beta}$ act as the task parameter of our parametric problem. The goal is to learn a model that can generate the entire PS of optimizable variables $\boldsymbol{x_p}$ for any given shared component configuration $\boldsymbol{x_s}$. Using PPSL-MOBO, the parametric PS model is denoted by:

$$\boldsymbol{x_p} = h_{\boldsymbol{\theta}(\boldsymbol{x_s})}(\boldsymbol{\lambda}), \tag{24}$$

where $h(\cdot)$ is a map from the preference space $\boldsymbol{\Delta}^{m-1}$ to the decision space of the non-shared variables $\mathbb{R}^{|\boldsymbol{p}|}$. The parameters of this map, $\boldsymbol{\theta}(\boldsymbol{x_s})$, are generated by the hypernetwork conditioned on the shared component values $\boldsymbol{x_s}$.

This formulation is particularly useful in scenarios where the shared components $\boldsymbol{x_s}$ are determined by external constraints, predefined modules, or customer choices, and the primary design task is to find the optimal configuration of $\boldsymbol{x_p}$ around these fixed components.

4.2 Results Analysis

We evaluate PPSL-MOBO against MOEA and MOBO methods on a real-world multi-objective test problems suite (Tanabe and Ishibuchi 2020; Lin et al. 2022). The experimental results in Tables 1 demonstrate the effectiveness and efficiency of our proposed method. PPSL-MOBO achieves competitive hypervolume performance across different shared component configurations, often matching or outperforming baseline methods that are specifically optimized for individual tasks. The most significant advantage of PPSL-MOBO lies in its computational and sample efficiency. While baseline methods must be re-executed from scratch for each new shared component configuration, PPSL-MOBO requires only a one-time training phase followed by near-instantaneous inference (in milliseconds) for new parameter values. This approach achieves substantial reductions in both total function evaluations and computational time compared to baseline methods, making it highly practical for applications requiring frequent parameter variations or expensive function evaluations.

Figure 2 illustrates the PFs learned for the Rocket Injector Design Problem (RE37) under different shared component configurations. The visualization demonstrates how constraining different design variables affects the shape and position of the achievable PFs. These visualizations confirm that PPSL-MOBO successfully captures the parametric variations in PF geometry across different shared component scenarios, providing decision-makers with comprehensive trade-off information for each design configuration.

5 Application Study: Dynamic Multi-Objective Optimization

In this section, we demonstrate the effectiveness of the proposed PPSL-MOBO framework for Dynamic Multi-Objective Optimization Problems (DMOPs), where the optimization objectives change over time.

5.1 Problem Introduction

Dynamic multi-objective optimization arises in numerous real-world scenarios, such as dynamic portfolio selection (Xu et al. 2006), time-varying process control (Tang and Meng 2021), adaptive scheduling (Abello, Bui, and Michalewicz 2011), and non-stationary machine learning (Kim, McKay, and Moon 2010). In these settings, the parameter at Eq. (1) becomes the time characteristic, inducing time-varying PS and PF to resolve sequentially.

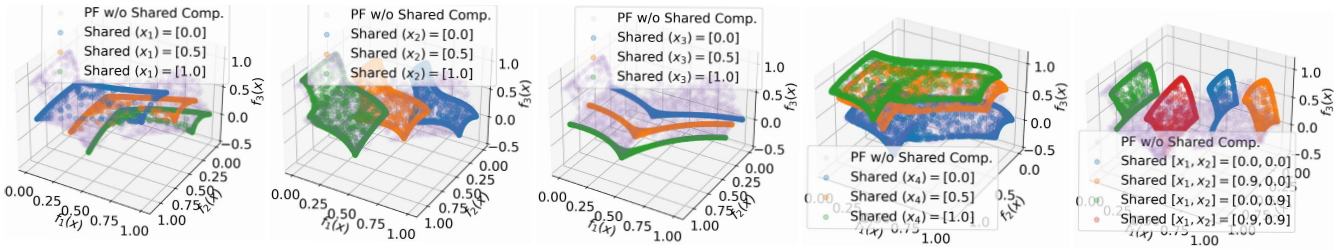

Figure 2: Learned PFs for the Rocket Injector Design Problem (RE37) with Shared Components. We demonstrate PFs for several specific parameter values, while PPSL-MOBO can obtain PS for any parameter in milliseconds. The transparent points denote the PFs for the problem without shared component. The shared components correspond to the decision variable x_1 (hydrogen flow angle), x_2 (hydrogen area), x_3 (oxygen area), and x_4 (oxidizer post tip thickness).

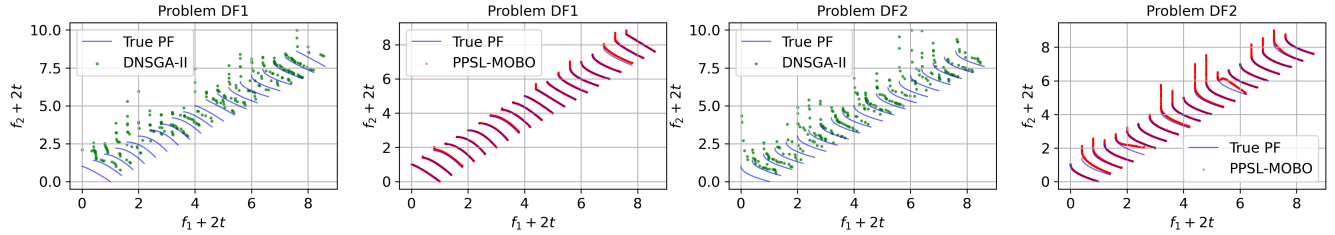

Figure 3: Comparison of generated solutions by DNSGA-II (green) and PPSL-MOBO (red) at each time step on DMOPs.

Traditional approaches to DMOPs often detect environmental changes and restart or re-initialize optimization at each change point, treating each time instance as an independent static problem (Zhou, Jin, and Zhang 2013; Jiang et al. 2022; Zhang et al. 2023). However, this disregards the inherent temporal correlation between solutions at adjacent time steps, potentially leading to inefficiency and suboptimal adaptation. With PPSL-MOBO, we treat the temporal variable t as a *parametric context*, enabling the learning of a *unified parametric PS model* that generalizes across the temporal dimension. This approach not only allows for efficient adaptation to environmental changes but also leverages knowledge transfer across time, capturing the underlying temporal dynamics of the evolving PS and PF.

To align with the algorithmic constraint that only the current time step's function evaluation is accessible, the task distribution P_t in PPSL-MOBO is set to a degenerated distribution, yielding N_t identical parameter samples at each generation. The experimental setup, implementation of PPSL-MOBO, and results are elaborated in Appendix.

5.2 Results and Analysis

Figure 3 illustrates the dynamic tracking capabilities of PPSL-MOBO compared to DNSGA-II (Deb, Rao N, and Karthik 2007) on benchmark problems DF1 and DF2 (Jiang et al. 2018). The visualization highlights the fundamental differences in how these algorithms respond to environmental changes. DNSGA-II, despite being specifically designed for dynamic optimization, struggles to converge to the true Pareto front within the limited two-generation window between environmental changes. Its solutions (depicted as scattered points) remain widely dispersed and fail to adequately approximate the complete Pareto front structure. In contrast, PPSL-MOBO leverages its pre-trained parametric model to instantaneously generate well-distributed Pareto fronts (shown as continuous curves) that closely approximate the true Pareto front at each time step.

6 Conclusion and Future Work

We presented PPSL-MOBO, a novel framework that learns parametric Pareto sets for expensive multi-objective optimization. By combining hypernetwork-LoRA architectures with Bayesian optimization, our approach efficiently maps both preferences and exogenous parameters to Pareto-optimal solutions through a single unified model, eliminating the need for retraining across parameter variations.

Our experiments demonstrate PPSL-MOBO's effectiveness in two challenging domains: (1) In multi-objective optimization with shared components, it achieves competitive performance using only a fraction of function evaluations compared to methods requiring re-execution for each parameter value. (2) In dynamic optimization, by treating time as a parametric context, PPSL-MOBO successfully adapts to changing environments and outperforms state-of-the-art dynamic MOEAs. These results validate that parametric learning of Pareto sets offers a powerful paradigm for expensive optimization scenarios.

Future work includes extending the framework to constrained multi-objective problems, developing theoretical guarantees on sample complexity, and exploring integration with interactive decision-making systems for real-time preference articulation.

Acknowledgments

The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China [GRF Project No. CityU11215723 and CityU11212524].

References

Abello, M. B.; Bui, L. T.; and Michalewicz, Z. 2011. An adaptive approach for solving dynamic scheduling with time-varying number of tasks—Part II. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, 1711–1718. IEEE.

Bank, B.; Guddat, J.; Klatte, D.; Kummer, B.; and Tammer, K. 1982. *Non-linear parametric optimization*, volume 58. Walter de Gruyter GmbH & Co KG.

Boyd, S. P.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge University Press.

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

Deb, K.; Rao N, U. B.; and Karthik, S. 2007. Dynamic multi-objective optimization and decision-making using modified NSGA-II: a case study on hydro-thermal power scheduling. In *Evolutionary Multi-Criterion Optimization*, 803–817. Springer.

Ehrgott, M. 2005. *Multicriteria optimization*. Springer.

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

Fiacco, A. V. 1976. Sensitivity analysis for nonlinear programming using penalty methods. *Mathematical Programming*, 10(1): 287–311.

Garcia, S.; and Trinh, C. T. 2019. Modular design: implementing proven engineering principles in biotechnology. *Biotechnology Advances*, 37(7): 107403.

Garnett, R. 2023. *Bayesian optimization*. Cambridge University Press.

Guha, R.; and Deb, K. 2024. Compromising Pareto-Optimality With Regularity in Platform-Based Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 28(6): 1746–1760.

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

Jiang, S.; Yang, S.; Yao, X.; Tan, K. C.; Kaiser, M.; and Krasnogor, N. 2018. Benchmark functions for the cec'2018 competition on dynamic multiobjective optimization. Technical report, Newcastle University.

Jiang, S.; Zou, J.; Yang, S.; and Yao, X. 2022. Evolutionary dynamic multi-objective optimisation: A survey. *ACM Computing Surveys*, 55(4): 1–47.

Kim, K.; McKay, R. I.; and Moon, B.-R. 2010. Multiobjective evolutionary algorithms for dynamic social network clustering. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, 1179–1186.

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

Lin, X.; Yang, Z.; and Zhang, Q. 2022. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations (ICLR)*.

Lin, X.; Yang, Z.; Zhang, Q.; and Kwong, S. 2020. Controllable Pareto Multi-Task Learning. *arXiv preprint arXiv:2010.06313*.

Lin, X.; Yang, Z.; Zhang, X.; and Zhang, Q. 2022. Pareto Set Learning for Expensive Multi-Objective Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 19231–19247.

Lin, X.; Zhang, X.; Yang, Z.; Liu, F.; Wang, Z.; and Zhang, Q. 2024. Smooth Tchebycheff Scalarization for Multi-Objective Optimization. In *International Conference on Machine Learning (ICML)*.

Lin, X.; Zhang, X.; Yang, Z.; and Zhang, Q. 2025. Dealing With Structure Constraints in Evolutionary Pareto Set Learning. *IEEE Transactions on Evolutionary Computation*, 29(3): 616–630.

Miettinen, K. 1999. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.

Milgrom, P.; and Shannon, C. 1994. Monotone comparative statics. *Econometrica: Journal of the Econometric Society*, 157–180.

Navon, A.; Shamsian, A.; Chechik, G.; and Fetaya, E. 2021. Learning the Pareto Front with Hypernetworks. In *International Conference on Learning Representations (ICLR)*.

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

Tang, L.; and Meng, Y. 2021. Data analytics and optimization for smart industry. *Frontiers of Engineering Management*, 8(2): 157–171.

Tsai, Y.-K.; and Malak Jr, R. J. 2021. A methodology for designing a nonlinear feedback controller via parametric optimization: State-parameterized nonlinear programming control. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 85383, V03AT03A011. American Society of Mechanical Engineers.

Wang, Y.; Seki, H.; Ohyama, S.; Ogawa, M.; Ohshima, M.; et al. 2000. Optimal grade transition control for polymerization reactors. *Computers & Chemical Engineering*, 24(2-7): 1555–1561.

Weaver-Rosen, J. M.; Leal, P. B.; Hartl, D. J.; and Malak Jr, R. J. 2020. Parametric optimization for morphing structures design: application to morphing wings adapting to changing flight conditions. *Structural and Multidisciplinary Optimization*, 62(6): 2995–3007.

Xu, J.; Luh, P. B.; White, F. B.; Ni, E.; and Kasiviswanathan, K. 2006. Power portfolio optimization in deregulated electricity markets with risk management. *IEEE Transactions on Power Systems*, 21(4): 1653–1662.

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

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

Zhang, X.; Yu, G.; Jin, Y.; and Qian, F. 2023. An adaptive Gaussian process based manifold transfer learning to expensive dynamic multi-objective optimization. *Neurocomputing*, 538: 126212.

Zhao, L.; Wang, P.; Shen, J.; Song, B.; and Zhang, Q. 2025. Component-Sharing Preference in Expensive Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 1–1.

Zhou, A.; Jin, Y.; and Zhang, Q. 2013. A population prediction strategy for evolutionary dynamic multiobjective optimization. *IEEE Transactions on Cybernetics*, 44(1): 40–53.