META PARETO FLOW LEARNING WITH ORDINARY DIFFERENTIAL EQUATION RESIDUAL SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Pareto set learning (PSL) amortizes multiobjective optimization by learning a continuous mapping from objective preferences to solutions. However, existing PSL methods mainly model task-specific Pareto sets, whose terminal solution manifolds can be geometrically misaligned across related tasks, making shared transferable structure difficult to capture from these mappings alone. We propose *Pareto Flow Learning* (PFL), which formulates each preference-conditioned scalarized problem as a continuous-time Pareto gradient flow and trains a neural model to approximate the resulting flows through ordinary differential equation (ODE) residual supervision. The learned model represents optimization dynamics in addition to Pareto-optimal solutions, providing complementary process-level information for transfer. We further extend PFL to meta-learning by learning shared structure across source-task Pareto flows and adapting the resulting *meta Pareto flow model* (mPFM) to unseen targets. Integrating mPFM with multiobjective Bayesian optimization yields mPFM-MOBO for search under limited evaluation budgets. Experiments on synthetic benchmarks and engineering design problems show improved early-stage convergence and competitive or better final performance on most target tasks, including under distribution shift. Ablation studies further indicate that Pareto-flow ODE supervision provides useful transferable information beyond terminal Pareto-set mappings alone.

1 INTRODUCTION

Multiobjective optimization lies at the core of many machine learning (Sener & Koltun, 2018; Dong et al., 2025; Lin et al., 2019), engineering design (Min et al., 2017), and scientific discovery (MacLeod et al., 2022; Low et al., 2024) problems, where multiple, often conflicting objectives should be optimized simultaneously. Since no single solution can generally optimize all objectives at once, the goal is instead to identify a set of Pareto optimal solutions that characterize different trade-offs among the objectives. Recently, Pareto set learning (PSL) (Lin et al., 2022b;a; Chengyu et al.; Xu et al., 2025) has emerged as a learning-based paradigm for multiobjective optimization. Rather than explicitly searching for a finite collection of (near-)Pareto optimal solutions, PSL learns a continuous mapping from objective preferences to corresponding optimal solutions, thereby representing the Pareto set as a continuous manifold in the decision space. Once learned, such a mapping can generate solutions for a wide range of preferences without requiring a separate optimization process for each preference. This amortized formulation makes PSL particularly attractive when diverse Pareto optimal solutions need to be generated efficiently.

However, many practical applications involve a family of related multiobjective optimization tasks induced by different environments or system configurations, rather than a single isolated task (Liu et al., 2024b). This motivates a *meta-learning* perspective, where knowledge acquired from previously solved source tasks is reused to facilitate Pareto set learning on an unseen target task. A central challenge, however, is to identify transferable structure across tasks. Existing PSL methods typically represent each task through a preference-to-solution mapping (Liu et al., 2025; Wei et al., 2024), which compactly parameterizes its terminal Pareto set. Even for related tasks, these Pareto sets may be geometrically misaligned in the decision space due to task-dependent shifts, transformations, or variations in the objective functions. Consequently, similarity between tasks can be difficult to capture directly from their terminal Pareto set mappings alone, limiting the shared structure available for meta-learning. This motivates a representation that relates tasks not only through where their

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Pareto optimal solutions lie, but also through how preference-conditioned solutions evolve toward them.

To complement the solution-level representation learned by conventional PSL, we consider the optimization process itself as an additional source of transferable information. Specifically, we view multiobjective optimization through the lens of continuous-time gradient flows (Ren et al., 2024), where, for a given preference over the objectives, a trajectory evolves from an initial candidate toward a preference-specific Pareto optimal region. We refer to this preference-conditioned dynamical representation as the *Pareto gradient flow*. Compared with a direct preference-to-solution mapping, such a flow characterizes not only the terminal solution but also intermediate optimization states, and can therefore encode information about the local optimization landscape encountered along the trajectory. Based on this perspective, we propose *Pareto Flow Learning* (PFL), which parameterizes preference-conditioned optimization trajectories with a neural model and trains it using residuals derived from the corresponding Pareto gradient-flow ordinary differential equations (ODE). This follows the general principle of residual-based supervision, as exemplified by physics-informed neural networks (Raissi et al., 2019; Chiu et al., 2022; Wong et al., 2026a;c;b), while applying it to optimization dynamics rather than physical systems. Building on this representation, we further develop a *meta* PFL framework that learns shared structure across the Pareto flows of related source tasks and adapts the resulting model to an unseen target task. We finally integrate the learned *meta Pareto flow model* (mPFM) into multiobjective Bayesian optimization, where it provides transferred optimization guidance under limited target task evaluation budgets.

- We formulate PFM as the approximation of preference-conditioned gradient-flow trajectories and introduce ODE residual supervision to learn both the intermediate optimization dynamics and their terminal Pareto optimal solutions.
- We develop a meta Pareto flow model that captures shared gradient-flow structures across a distribution of source multiobjective optimization tasks and supports efficient adaptation to unseen target tasks.
- We integrate the learned meta Pareto flow into multiobjective Bayesian optimization, where the adapted model serves as a task-aware candidate generator for sample-efficient optimization under limited evaluation budgets.
- We conduct extensive experiments on benchmark multiobjective optimization problems, demonstrating that ODE residual supervision improves the learning and transfer of preference-conditioned optimization dynamics, thereby enhancing efficiency on unseen target tasks.

2 PRELIMINARIES

A multiobjective optimization problem (Deb, 2011; Branke, 2008) can be formulated, without loss of generality, as:

$$\min \ \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})], \quad \text{s.t. } \mathbf{x} \in \mathcal{X}, \tag{1}$$

where $f_i(\mathbf{x}), (i \in \{1, \ldots, m\})$ is the ith objective function, m is the number of objectives, $\mathbf{x}$ represents a candidate solution (decision vector), $\mathcal{X} = \{\mathbf{x} = (x_1, \ldots, x_d) | L_j \leq x_j \leq U_j,\ j = 1, \ldots, d\}$ is the decision space of all possible solutions, d is the dimensionality of the decision vector, and L_j and U_j are the lower and upper bounds of the jth decision variable.

Given the formulation in equation 1, we revisit definitions of some basic concepts in multiobjective optimization below:

Pareto Dominance. Solution $\mathbf{x}_a$ Pareto dominates $\mathbf{x}_b$, if $\forall i \in \{1, 2, \ldots, m\}$, $f_i(\mathbf{x}_a) \leq f_i(\mathbf{x}_b)$ and $\exists i' \in \{1, 2, \ldots, m\}$ such that $f_{i'}(\mathbf{x}_a) < f_{i'}(\mathbf{x}_b)$.

Pareto Optimal Solution. Solution $\mathbf{x}_a$ is said to be Pareto optimal if there exists no other solution in $\mathcal{X}$ that Pareto dominates $\mathbf{x}_a$.

Pareto Set. The set of all Pareto optimal solutions constitutes the Pareto set in decision space.

Pareto Front. The image of the Pareto set in the objective space is referred to as the Pareto front.

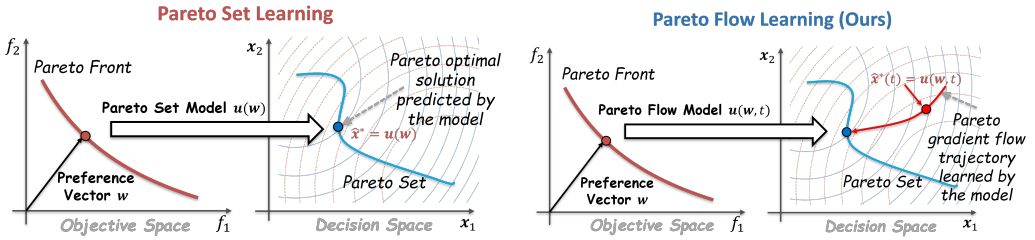


Figure 1: Comparison between PSL and the proposed PFL. PSL directly maps each preference vector to a Pareto optimal solution, whereas PFL captures both the terminal Pareto optimal solutions and the continuous optimization trajectories induced by preference-conditioned gradient flows.

3 Pareto Flow Learning via ODE Residual Supervision

In this section, we introduce the PFL with ODE residual supervision. We first formulate multi-objective optimization as a family of gradient-flow ODEs, referred to as *Pareto gradient flows*. A neural network is then used to parameterize the corresponding solution trajectories and is trained by minimizing the ODE residual and initial-condition losses. As illustrated in Figure 1, the resulting model provides a continuous-time representation of the optimization dynamics, whose terminal states approximate the Pareto set. The details are presented below.

Pareto Gradient Flow Based on Augmented Tchebycheff Scalarization. Decomposition-based approaches (Zhang & Li, 2007; Liu et al., 2013) are widely used for multiobjective optimization. Their main idea is to transform a multiobjective optimization problem into a set of single-objective subproblems through scalarization functions. Solving these subproblems with different preference vectors yields a collection of solutions that can be used to approximate the Pareto front and the Pareto set. In this work, we adopt the augmented Tchebycheff scalarization:

$$f^{\mathrm{tch}}(\mathbf{x};\mathbf{w}) = \max_{i\in\{1,\dots,m\}} \{w_i f_i(\mathbf{x})\} + \eta \sum_{i=1}^{m} w_i f_i(\mathbf{x}), \tag{2}$$

where $\eta > 0$ is the augmentation term, and $\mathbf{w}$ is a preference vector selected from $\mathcal{W} = \{\mathbf{w} = (w_1,\dots,w_m) \,|\, w_i \geq 0,\ \sum_{i=1}^{m} w_i = 1\}$.

Under the augmented Tchebycheff scalarization, each preference vector $\mathbf{w}$ defines a scalar subproblem. We describe the optimization dynamics of this subproblem through a continuous-time Pareto gradient flow (Santambrogio, 2017; Wendin & Altafini, 2026). Specifically, given an initial point $\mathbf{x}_0$, $u(\mathbf{w},t)$ is defined as the solution of the following ODE:

$$\frac{\partial u(\mathbf{w},t)}{\partial t} = -\nabla_{\mathbf{u}} f^{\mathrm{tch}}\big(u(\mathbf{w},t);\mathbf{w}\big), \qquad \text{s.t. } u(\mathbf{w},t_0) = \mathbf{x}_0. \tag{3}$$

Here, t is an artificial time variable that parameterizes the optimization trajectory, t_0 is an artificially set initial time, and $u(\mathbf{w},t)$ denotes the state reached at time t when starting from $\mathbf{x}$ under the gradient flow induced by the scalarized objective $f^{\mathrm{tch}}(\cdot;\mathbf{w})$.

Training of the Pareto Flow Model. The gradient flow equation 3 characterizes the continuous-time optimization trajectory induced by the scalarized objective. Therefore, for a sufficiently large terminal time $t_{\max}$, the terminal state of the trajectory can be used as an approximation to the optimal solution associated with the corresponding preference vector. We parameterize the solution trajectory by a neural network $\hat{u}_\theta(\mathbf{w},t)$, which takes a preference vector $\mathbf{w}$ and a time variable t as inputs. The network is trained by enforcing both the initial condition and the ODE residual of the Pareto gradient flow. This gives the following loss function:

$$\begin{aligned}\mathcal{L}_{\mathrm{PFM}} = {} & \lambda_{\mathrm{IC}} \mathbb{E}_{\mathbf{w}\sim\mathcal{W}} \left[\|\hat{u}_\theta(\mathbf{w},t_0) - \mathbf{x}_0\|_2^2 \right] \\ & + \lambda_{\mathrm{DE}} \mathbb{E}_{\mathbf{w}\sim\mathcal{W},\, t\sim p(t)} \left[\left\| \frac{\partial \hat{u}_\theta(\mathbf{w},t)}{\partial t} + \nabla_{\mathbf{u}} f^{\mathrm{tch}}\big(\hat{u}_\theta(\mathbf{w},t);\mathbf{w}\big) \right\|_2^2 \right].\end{aligned} \tag{4}$$

Here, $p(t)$ is chosen as a uniform distribution over $[t_0, t_{\max}]$, where t_0 and $t_{\max}$ denote the prescribed initial and terminal times, respectively. The coefficients λ_{IC} and λ_{DE} balance the initial-

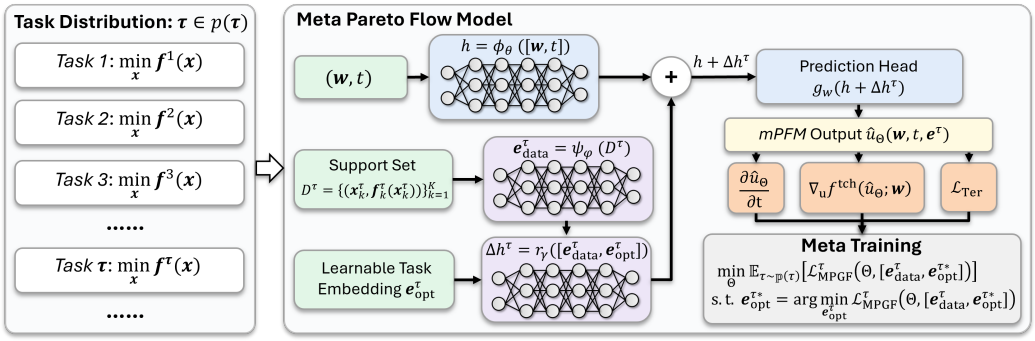


Figure 2: Architecture of the meta Pareto flow model.

condition loss and the ODE residual loss. After training, the terminal prediction $\hat{u}_\theta(\mathbf{w}, t_{\max})$ provides an approximate optimal solution of the scalarized subproblem associated with $\mathbf{w}$. Varying $\mathbf{w} \in \mathcal{W}$ yields a set of candidate solutions that can be used to approximate the Pareto set.

Although equation 4 defines the loss function of the Pareto flow model, directly optimizing this loss can be difficult in practice. The gradient field of the preference conditioned objective may vary sharply over the decision space, causing the ODE residual to be sensitive to local variations in the objective landscape and leading to unstable or high-variance training signals. We therefore adopt a stabilized implementation: the temporal derivative is approximated by finite differences, and the objective gradient is estimated through evolution strategies (Schwefel, 1993; Salimans et al., 2017). These approximations preserve the gradient-flow training principle while producing a smoother residual, thereby improving training stability. Further details are provided in **Appendix B**.

4 Meta Pareto Flow Learning

We further extend PFL to the meta-learning setting by learning transferable Pareto gradient-flow representations across related multiobjective optimization tasks. Specifically, the proposed *meta Pareto flow learning* framework learns shared structure from the Pareto gradient flows of multiple source tasks and adapts the resulting model to an unseen target task. By incorporating ODE residual supervision during meta-training and adaptation, the framework exploits process-level information along the optimization trajectories in addition to terminal Pareto-set information, providing additional guidance for learning the target-task Pareto set.

Architecture of the Meta Pareto Flow Model. Given a preference vector $\mathbf{w} \in \mathcal{W}$ and an artificial time variable $t \in [t_0, t_{\max}]$, the mPFM predicts a decision vector

$$\hat{\mathbf{x}} = \hat{u}_\Theta(\mathbf{w}, t, \mathbf{e}^\tau), \tag{5}$$

where $\mathbf{e}^\tau = [\mathbf{e}^\tau_{\text{opt}}, \mathbf{e}^\tau_{\text{data}}]$ is a task-dependent embedding with the aim of parameterizing the optimization task τ as a vector, and Θ denotes the shared network parameters. The model first maps the concatenated input $(\mathbf{w}, t)$ to a latent representation through a backbone:

$$\mathbf{h} = \phi_\theta([\mathbf{w}, t]). \tag{6}$$

This backbone is shared across tasks and represents the task-invariant component of the learned gradient flow dynamics. To condition the model on a specific task τ, we use a support set $\mathcal{D}^\tau = \{(\mathbf{x}^\tau_k, \mathbf{f}^\tau(\mathbf{x}^\tau_k))\}_{k=1}^K$ and encode it with a permutation-invariant task encoder:

$$\mathbf{e}^\tau_{\text{data}} = \psi_\varphi(\mathcal{D}^\tau). \tag{7}$$

This embedding summarizes the observed task data. In addition, mPFM maintains a learnable task-specific embedding $\mathbf{e}^\tau_{\text{opt}}$, which provides lightweight adaptation parameters for each task. The two embeddings are concatenated and transformed into a residual modulation of the backbone representation:

$$\Delta\mathbf{h}^\tau = r_\gamma([\mathbf{e}^\tau_{\text{data}}, \mathbf{e}^\tau_{\text{opt}}]), \qquad \mathbf{h}^\tau = \mathbf{h} + \Delta\mathbf{h}^\tau. \tag{8}$$

A prediction head then maps the task-conditioned representation to the decision space:

$$\hat{u}_\Theta(\mathbf{w}, t, \mathbf{e}^\tau) = g_\omega(\mathbf{h}^\tau). \tag{9}$$

This design separates shared and task-specific components. The backbone and prediction head model the common structure of Pareto gradient flow trajectories, while the task embeddings modulate the latent representation for a particular task. The residual form also allows the model to start from a shared predictor and introduce task-dependent corrections through adaptation.

Training of the Meta Pareto Flow Model. We train the mPFM over a task distribution $p(\tau)$, where each task τ defines a multiobjective function $\mathbf{f}^\tau : \mathcal{X} \to \mathbb{R}^m$. The goal is to learn shared parameters that are useful across tasks, while adapting only a low-dimensional task embedding for each individual task. We formulate this as a bilevel optimization problem:

$$\begin{aligned} \min_{\Theta} \quad & \mathbb{E}_{\tau\sim p(\tau)} \left[\mathcal{L}^\tau_{\mathrm{mPFM}} \left(\hat{u}_\Theta \left(\mathbf{w}, t, [\mathbf{e}^\tau_{\mathrm{data}}, \mathbf{e}^{\tau *}_{\mathrm{opt}}] \right) \right) \right] \\ \text{s.t.} \quad & \mathbf{e}^{\tau *}_{\mathrm{opt}} = \arg\min_{\mathbf{e}^\tau_{\mathrm{opt}}} \mathcal{L}^\tau_{\mathrm{mPFM}} \left(\hat{u}_\Theta \left(\mathbf{w}, t, [\mathbf{e}^\tau_{\mathrm{data}}, \mathbf{e}^\tau_{\mathrm{opt}}] \right) \right), \end{aligned} \tag{10}$$

where $\Theta = \{\theta, \varphi, \gamma, \omega\}$ contains the shared parameters of the backbone, task encoder, residual calculator, and prediction head. The lower-level problem adapts the task-specific embedding $\mathbf{e}^\tau_{\mathrm{opt}}$ for a sampled task, while the upper-level problem updates the model using the post-adaptation loss.

For each task, $\mathcal{L}^\tau_{\mathrm{mPFM}}$ is calculated as

$$\mathcal{L}^\tau_{\mathrm{mPFM}} = \lambda_{\mathrm{PFM}} \mathcal{L}^\tau_{\mathrm{PFM}} + \lambda_{\mathrm{Ter}} \mathcal{L}^\tau_{\mathrm{Ter}} + \lambda_{\mathrm{Reg}} \left\| \mathbf{e}^\tau_{\mathrm{opt}} \right\|_2^2, \tag{11}$$

where $\mathcal{L}^\tau_{\mathrm{PFM}}$ is the Pareto gradient-flow loss defined in equation 4. The terminal loss $\mathcal{L}^\tau_{\mathrm{Ter}} = \mathbb{E}_{\mathbf{w}\sim\mathcal{W}} \left[f^{\mathrm{tch}} \left(\hat{u}_\Theta \left(\mathbf{w}, t_{\max}, \mathbf{e}^\tau \right); \mathbf{w} \right) \right]$ evaluates the quality of the terminal prediction at $t_{\max}$ through the scalarized objective, encouraging the learned trajectory to end at a high-quality solution for the corresponding preference vector. The final term regularizes the task-specific embedding and limits excessive task-dependent deviation from the shared model.

In practice, the inner problem in equation 10 is approximated by a fixed number of gradient steps. For each sampled task, we first evaluate a small support set and compute $\mathbf{e}^\tau_{\mathrm{data}}$ using the task encoder. We then initialize $\mathbf{e}^\tau_{\mathrm{opt}}$ and update it with the task-level Pareto gradient flow loss while keeping the shared parameters fixed. After this adaptation, the resulting embedding $\mathbf{e}^{\tau *}_{\mathrm{opt}}$ is used to compute the post-adaptation loss, and the shared parameters Θ are updated over a minibatch of tasks. When finetuning the mPFM for a target multiobjective optimization task, the shared parameters are kept fixed. Given a small set of observations from the target task, the task encoder first produces the data-dependent embedding $\mathbf{e}_{\mathrm{data}}$, and the task-specific embedding $\mathbf{e}_{\mathrm{opt}}$ is then adapted using the same inner-loop objective as in meta-training. After adaptation, the terminal predictions $\hat{u}_\Theta \left(\mathbf{w}, t_{\max}; [\mathbf{e}_{\mathrm{data}}, \mathbf{e}_{\mathrm{opt}}] \right), \mathbf{w} \in \mathcal{W}$, are used as candidate solutions for the scalarized subproblems associated with different preference vectors.

5 Multiobjective Bayesian Optimization Driven by Meta Pareto Flow Model

We now describe how the learned mPFM is used within multiobjective Bayesian optimization. The resulting algorithm is named as mPFM-MOBO, which is detailed in **Algorithm 1**. Let $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ denote the evaluated dataset on the target task, where $\mathbf{y}^{(n)} = \mathbf{f}(\mathbf{x}^{(n)})$, and $\mathbf{f}(\mathbf{x})$ is the objective function vector of the target task. At each Bayesian optimization iteration, we fit a group of Gaussian process (GP) surrogate (Binois & Wycoff, 2022) models to $\mathcal{D}$, with one independent GP for each objective. The surrogate provides a predictive mean and uncertainty for any candidate solution. We employ the lower confidence bound (LCB) as the acquisition (Srinivas et al., 2012; Lin et al., 2022b):

$$\mathrm{LCB}_i(\mathbf{x}) = \boldsymbol{\mu}_i(\mathbf{x}) - \beta \boldsymbol{\sigma}_i(\mathbf{x}), \tag{12}$$

where $\boldsymbol{\mu}_i(\mathbf{x})$ and $\boldsymbol{\sigma}_i(\mathbf{x})$ are the GP predictive mean and standard deviation for the ith objective, and β controls the exploration strength. The mPFM is then adapted to the current target task. Given the observed data $\mathcal{D}$, the task encoder computes the data embedding $\mathbf{e}_{\mathrm{data}} = \psi_\varphi(\mathcal{D})$. Starting from the pretrained mPFM parameters, we keep the shared modules fixed and optimize the task-specific embedding $\mathbf{e}_{\mathrm{opt}}$ using the GP surrogate. The task embedding is obtained by minimizing the scalarized LCB objective:

$$\mathbf{e}^*_{\mathrm{opt}} = \arg\min_{\mathbf{e}_{\mathrm{opt}}} \mathbb{E}_{\mathbf{w}\sim\mathcal{W}} \left[f^{\mathrm{lcb-tch}} \left(\hat{u}_\Theta \left(\mathbf{w}, t_{\max}, [\mathbf{e}_{\mathrm{data}}, \mathbf{e}_{\mathrm{opt}}] \right); \mathbf{w} \right) \right], \tag{13}$$

Algorithm 1 mPFM-MOBO

Require: Initial dataset size N_0, evaluation budget N, batch size q, pretrained mPFM parameters Θ

Initialize $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{f}(\mathbf{x}^{(n)}))\}_{n=1}^{N_0}$ with randomly sampled solutions
while $|\mathcal{D}| < N$ **do**
  Fit independent GP surrogate models for the objectives according to $\mathcal{D}$
  Compute $\mathbf{e}_{\text{data}} = \psi_{\varphi}(\mathcal{D})$ using the task encoder
  Restore the pretrained mPFM parameters Θ and freeze the shared modules
  Obtain $\mathbf{e}^*_{\text{opt}}$ via equation 13
  Sample preference vectors $\{\mathbf{w}_j\}_{j=1}^{N_{\text{pool}}}$ from $\mathcal{W}$
  Generate candidate pool $\mathcal{X}_{\text{pool}} = \{\hat{u}_{\Theta}(\mathbf{w}_j, t_{\max}; [\mathbf{e}_{\text{data}}, \mathbf{e}^*_{\text{opt}}])\}_{j=1}^{N_{\text{pool}}}$
  Compute LCB values for all $\mathbf{x} \in \mathcal{X}_{\text{pool}}$
  Select a batch $\mathcal{B}$ of q candidates by greedy hypervolume maximization
  Evaluate $\mathbf{f}(\mathbf{x})$ for each $\mathbf{x} \in \mathcal{B}$
  Update $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}, \mathbf{f}(\mathbf{x})) : \mathbf{x} \in \mathcal{B}\}$
end while
return Evaluated solutions and objective values

where

$$f^{\text{lcb}-\text{tch}}(\mathbf{x}) = \max_{i \in \{1,\dots,m\}} \{w_i \text{LCB}_i(\mathbf{x})\} + \eta \sum_{i=1}^{m} w_i \text{LCB}_i(\mathbf{x}). \tag{14}$$

This step adapts the pretrained gradient flow model toward regions that are promising under the current surrogate model, while preserving the shared dynamics learned during meta-training. After adaptation, we sample a set of preference vectors $\{\mathbf{w}_j\}_{j=1}^{N_{\text{pool}}}$ and use mPFM to construct a candidate pool

$$\mathcal{X}_{\text{pool}} = \left\{\hat{u}_{\Theta}\left(\mathbf{w}_j, t_{\max}, [\mathbf{e}_{\text{data}}, \mathbf{e}^*_{\text{opt}}]\right)\right\}_{j=1}^{N_{\text{pool}}}. \tag{15}$$

The GP models evaluates the LCB values of all candidates in the pool. To select a batch for expensive evaluation, we use a greedy hypervolume criterion employed in (Lin et al., 2022b). The selected candidates are then evaluated by the true objective function and appended to the dataset. In this optimization process, mPFM-MOBO uses mPFM as a task-adaptive candidate generator. The GP surrogate provides uncertainty-aware objective estimates, while the pretrained meta Pareto gradient flow model restricts the acquisition search to a structured set of candidates parameterized by preference vectors. Compared with optimizing the acquisition function directly in the full decision space, this reduces the candidate search to sampling and adapting within the learned preference-conditioned solution manifold, thus enhancing the efficiency of the optimization.

Note that, the target task adaptation in equation 13 is intentionally different from that used during meta-training. During meta-training, we assume that the source objectives are either directly accessible or supported by sufficient source-domain data to provide a reasonably accurate representation of the objective landscape, making the Pareto-flow loss a suitable supervision signal for adapting the task-specific embedding. In the target task optimization process, however, only a limited number of target evaluations are available, and the GP surrogate may not yet faithfully represent the target landscape, especially its local gradient field. Using the same flow-based adaptation in this regime can therefore introduce unreliable guidance. We instead keep the shared mPFM fixed and adapt only $\mathbf{e}_{\text{opt}}$ using the LCB objective in equation 13, so that the meta-learned flow is retained as a transferable prior while the surrogate provides target specific search guidance.

6 EXPERIMENTS

In this section, we evaluate the performance of mPFM-MOBO on synthetic multiobjective benchmark problems and engineering design benchmark problems. Our experiments are designed to answer the following questions: (i) whether the learned meta Pareto gradient flow improves sample efficiency in multiobjective Bayesian optimization; (ii) whether the learned optimization dynamics generalizable to both in-distribution and out-of-distribution target tasks; and (iii) whether the pro-

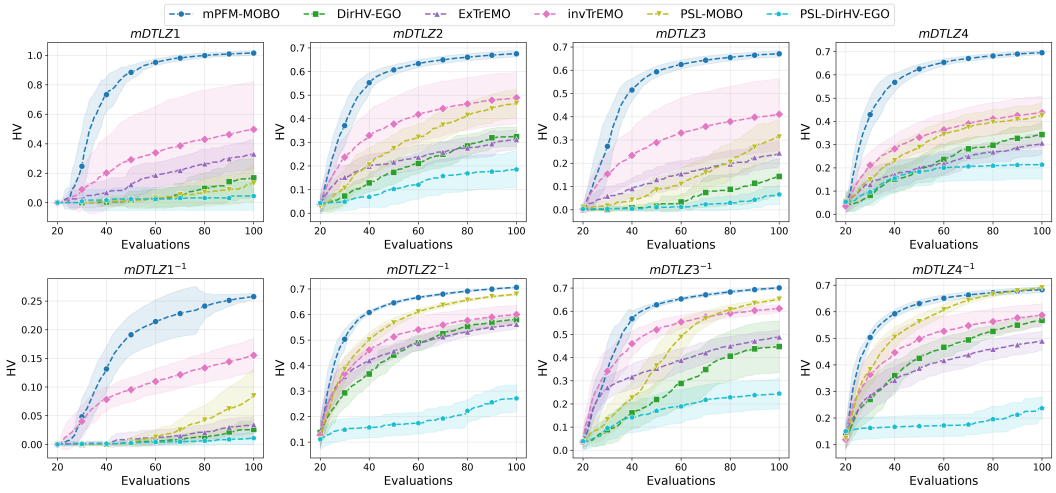


Figure 3: Comparison of HV convergence trends averaged over 20 independent runs of mPFM-MOBO, DirHV-EGO, ExTrEMO, invTrEMO, PSL-MOBO, and PSL-DirHV-EGO on in-distribution target synthetic tasks. Shaded areas represent one standard deviation on either side of the mean.

posed method shows practical potential on real-world optimization tasks. We first summarize the general experimental setup below, with additional implementation details provided in **Appendix C**.

Baselines. We compare mPFM-MOBO with five representative baselines: PSL-MOBO (Lin et al., 2022b), DirHV-EGO (Zhao & Zhang, 2023), PSL-DirHV-EI (Zhang et al., 2024), ExTrEMO (Liu et al., 2024a), and invTrEMO (Liu et al., 2025). PSL-MOBO and PSL-DirHV-EGO are Pareto-set-learning-enhanced multiobjective Bayesian optimization methods, where a preference-conditioned model is used to support the generation of Pareto optimal solutions. DirHV-EGO is a recently proposed multiobjective Bayesian optimization method based on directional hypervolume improvement. ExTrEMO and invTrEMO are transfer multiobjective Bayesian optimization methods that exploit cross-task knowledge to improve optimization on a target task. These baselines allow us to compare mPFM-MOBO with both Pareto-set-learning-based multiobjective Bayesian optimization methods and transferrable multiobjective Bayesian optimization methods.

Synthetic Benchmarks. Following (Liu et al., 2025), we consider two groups of synthetic benchmark problems, mDTLZ1–4 and mDTLZ1^{-1}–mDTLZ4^{-1}, which are adapted from the widely used DTLZ (Deb et al., 2005) and DTLZ^{-1} (Ishibuchi et al., 2016) problem families, respectively. These benchmarks introduce two task parameters, δ_1 and δ_2, to control the similarity between source and target tasks. To construct the source-task distribution, we randomly sample δ_1 and δ_2 from $[0.7, 1]$ and $[-0.25, 0.25]$, respectively. During mPFM training, each epoch uses a batch of randomly sampled source tasks with different values of δ_1 and δ_2. For target tasks, we consider both in-distribution and out-of-distribution settings. In the in-distribution setting, we set $\delta_1 = 0.7$ and $\delta_2 = 0.25$, which lie on the boundary of the source-task distribution. This setting contains eight target problems, including mDTLZ1–mDTLZ4 and mDTLZ1^{-1}–mDTLZ4^{-1}. In the out-of-distribution setting, we set $\delta_1 = 0.6$ and $\delta_2 = 0.35$ for the same eight problem families, placing the target tasks outside the source-task distribution. This design allows us to examine whether the learned mPFM captures transferable information and whether such information remain effective under distribution shift.

Engineering Design Benchmarks. To further investigate the potential of practical applicability of mPFM-MOBO, we consider four engineering design benchmark problems from (Tanabe & Ishibuchi, 2020): the four-bar truss, hatch cover, coil compression spring, and welded beam design problems. Following the idea in (Liu et al., 2024b), we extend each problem into a family of related tasks by sampling selected design and physical parameters from predefined distributions. This construction allows mPFM to be trained across multiple parameter configurations of the same engineering problem and then adapted to a new configuration from that family. As in the synthetic benchmarks, the source tasks are generated from a prescribed parameter distribution, while the tar-

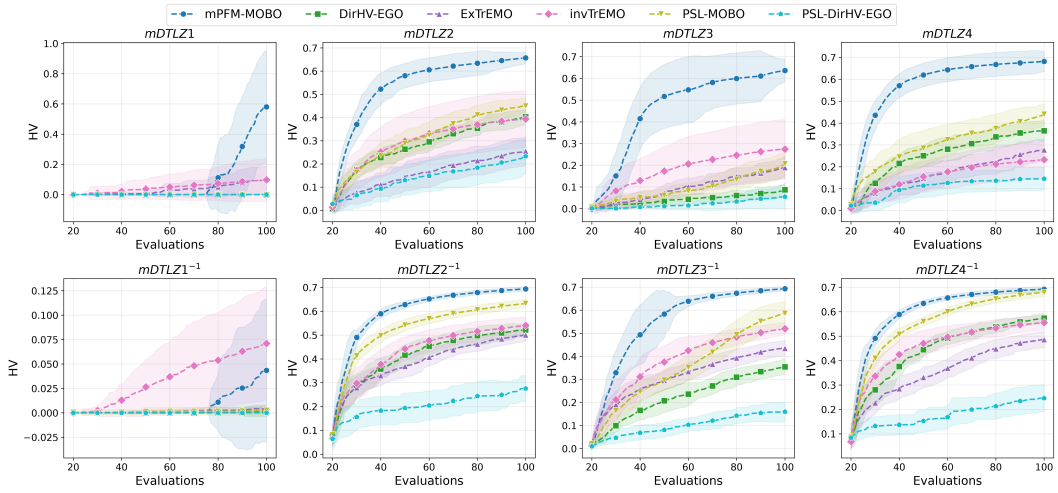


Figure 4: Comparison of HV convergence trends averaged over 20 independent runs of mPFM-MOBO, DirHV-EGO, ExTrEMO, invTrEMO, PSL-MOBO, and PSL-DirHV-EGO on out-of-distribution target synthetic tasks. Shaded areas represent one standard deviation on either side of the mean.

get tasks include both in-distribution and out-of-distribution settings. Detailed parameter ranges and task-generation procedures are provided in **Appendix C**.

Performance Metrics. For the comparison of results, we use the hypervolume (HV) as the performance metric to assess the quality of the solutions obtained by each baseline (Auger et al., 2012).

6.1 Performances of mPFM-MOBO on In-Distribution Synthetic Tasks

Figure 3 compares the HV convergence trajectories of mPFM-MOBO with the five baselines on the in-distribution synthetic target tasks. Overall, mPFM-MOBO achieves the best final performance on 7 out of the 8 tasks and exhibits consistently faster HV improvement during the early and middle stages of optimization. This behavior is particularly relevant in the Bayesian optimization setting, where performance under a limited evaluation budget is often more important than asymptotic convergence alone. The results therefore suggest that the learned meta Pareto flow provides useful structural guidance for locating promising Pareto optimal regions with fewer target-task evaluations, rather than merely improving the final fitted Pareto set. The only exception is mDTLZ4^{-1}, where PSL-MOBO attains a slightly higher final HV. However, mPFM-MOBO still converges faster in the initial stage and clearly outperforms the remaining baselines. This indicates that the learned flow prior remains beneficial for early search. In general, the results show that the benefit of meta Pareto flow learning is most evident in the low-data regime, supporting its role as an informative initialization and search prior for target-task multiobjective Bayesian optimization.

6.2 Performance of mPFM-MOBO on Out-of-Distribution Synthetic Tasks

Figure 4 compares the HV convergence trajectories of mPFM-MOBO and the five baselines on the out-of-distribution synthetic target tasks. Despite the distribution shift between the source and target tasks, mPFM-MOBO remains the best-performing or statistically competitive method on most problems. In particular, it significantly outperforms all baselines on mDTLZ1–mDTLZ4 and mDTLZ2^{-1}–mDTLZ3^{-1}, while also showing consistently strong convergence under limited target-task evaluations. This suggests that the meta Pareto gradient flow learned from the source tasks retains useful optimization structure beyond the training distribution and can provide effective guidance during target-task adaptation. On mDTLZ1^{-1}, invTrEMO achieves the highest final mean HV, although the relatively large variance makes several methods statistically comparable to mPFM-MOBO. This exception indicates that the benefit of the transferred flow prior can vary with the degree and nature of the task shift. Nevertheless, the overall performance across the out-of-

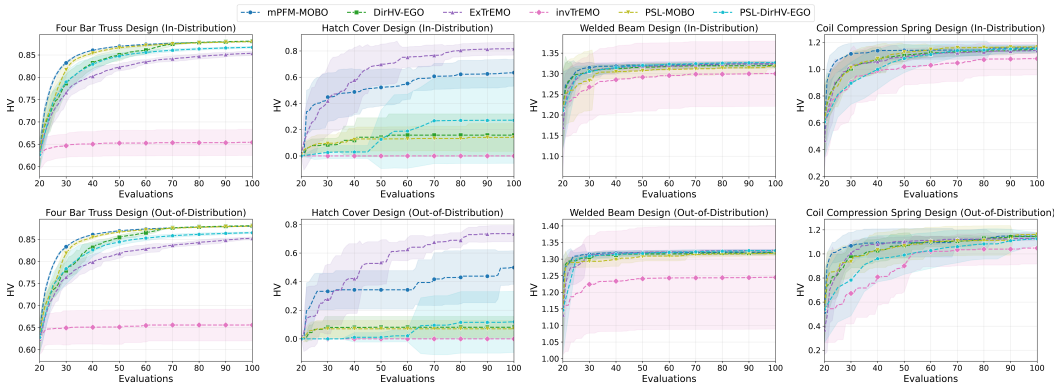


Figure 5: Comparison of HV convergence trends averaged over 20 independent runs of mPFM-MOBO, DirHV-EGO, ExTrEMO, invTrEMO, PSL-MOBO, and PSL-DirHV-EGO on in-distribution and out-of-distribution engineering design benchmark problems. Shaded areas represent one standard deviation on either side of the mean.

distribution tasks suggests that mPFM-MOBO does not rely solely on source-task Pareto-set similarity, but can retain useful transferable guidance when the target optimization problem departs from the source-task distribution.

6.3 Performance of mPFM-MOBO on Engineering Design Benchmarks

Figure 5 compares the HV convergence trajectories of mPFM-MOBO and the five baselines on both in-distribution and out-of-distribution engineering design tasks. On six of the eight tasks, mPFM-MOBO shows faster convergence during the early and middle stages of Bayesian optimization while maintaining competitive final HV, indicating that the transferred meta Pareto gradient flow can provide useful guidance under limited evaluation budgets. This advantage is particularly meaningful for engineering design problems, where objective evaluations are often expensive and early identification of promising regions is therefore important. The main exceptions are the in-distribution and out-of-distribution hatch cover design tasks, where ExTrEMO achieves better final performance. This suggests that the usefulness of the transferred flow prior depends on the structure of the underlying design problem and the degree to which source-task information remains relevant to the target task. Overall, the engineering results support the practical value of mPFM-MOBO as a transferable search prior across related design tasks, while also highlighting its problem-dependent behavior.

Remark: *In addition to the main results,* ***Appendix D*** *to* ***H*** *provide further analysis of mPFM-MOBO. These include visualizations of the learned optimization trajectories and Pareto fronts, ablation studies on ODE residual supervision and task-embedding adaptation, and a comparison of the batch optimization performance of mPFM-MOBO.*

7 Conclusion

In this paper, we propose PFL for learning preference-conditioned optimization trajectories toward Pareto optimal solutions. By formulating multiobjective optimization under Tchebycheff scalarization as a Pareto gradient flow, PFL parameterizes the resulting trajectories with a neural model and learns them through ODE residual supervision. We further extended PFL to the meta-learning setting, where shared structure across the Pareto flows of related source tasks is learned and adapted to unseen target tasks. The resulting meta Pareto flow model was integrated into multiobjective Bayesian optimization to provide transferable search guidance under limited evaluation budgets. Experiments on synthetic benchmarks and engineering design benchmark problems show that mPFM-MOBO generally improves convergence across related target tasks, including under distribution shift. Ablation results further indicate that incorporating process-level flow information can provide additional transferable guidance beyond learning terminal Pareto-set mappings alone.

Reproducibility Statement

We have provided detailed descriptions of the proposed meta Pareto flow learning method, algorithmic procedures, benchmark settings, implementation details, and hyperparameter configurations in the main paper and appendices. To further facilitate reproducibility, we will release the source code and experiment configurations upon acceptance of the paper.

AI Use Statement

Generative AI tools were used to assist with language editing and to improve the clarity and readability of the manuscript. They were also used to support the presentation of experimental results, including the plotting, styling, and visual refinement of figures, as well as the formatting and refinement of LaTeX tables. Generative AI was not used to generate, alter, or manipulate experimental data, conduct experiments, or determine the reported results. All AI-assisted text, figures, and tables were reviewed and verified by the authors, who take full responsibility for the final content of the paper.

References

Hossain M Amir and Takashi Hasegawa. Nonlinear mixed-discrete structural optimization. *Journal of Structural Engineering*, 115(3):626–646, 1989.

Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science*, 425:75–103, 2012.

Tianyi Bai, Yang Li, Yu Shen, Xinyi Zhang, Wentao Zhang, and Bin Cui. Transfer learning for bayesian optimization: A survey. *arXiv preprint arXiv:2302.05927*, 2023.

Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Mickael Binois and Nathan Wycoff. A survey on high-dimensional gaussian process modeling with application to bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):1–26, 2022.

Mickaël Binois, Victor Picheny, Patrick Taillandier, and Abderrahmane Habbal. The kalai-smorodinsky solution for many-objective bayesian optimization. *Journal of Machine Learning Research*, 21(150):1–42, 2020.

Jürgen Branke. *Multiobjective optimization: Interactive and evolutionary approaches*, volume 5252. Springer Science & Business Media, 2008.

Franklin Y Cheng and Xiang-Song Li. Generalized center method for multiobjective engineering optimization. *Engineering Optimization*, 31(5):641–661, 1999.

Ji Cheng, Bo Xue, and Qingfu Zhang. Parametric pareto set learning for expensive multi-objective optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 36829–36837, 2026.

LU Chengyu, Zhenhua Li, Xi Lin, Ji Cheng, and Qingfu Zhang. Neural evolution strategy for black-box pareto set learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. *Advances in neural information processing systems*, 33:9851–9864, 2020.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in neural information processing systems*, 34:2187–2200, 2021.

Kalyanmoy Deb. *Multi-objective optimisation using evolutionary algorithms: an introduction*. Springer, 2011.

Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, pp. 105–145. Springer, 2005.

Junhao Dong, Jiao Liu, Xinghua Qu, and Yew-Soon Ong. Confound from all sides, distill with resilience: Multi-objective adversarial paths to zero-shot robustness. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 624–634, 2025.

Michael TM Emmerich, Kyriakos C Giannakoglou, and Boris Naujoks. Single-and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

Paul Feliot, Julien Bect, and Emmanuel Vazquez. A bayesian approach to constrained single-and multi-objective optimization. *Journal of Global Optimization*, 67(1):97–133, 2017.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

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

Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *International conference on machine learning*, pp. 1492–1501. PMLR, 2016.

José Miguel Hernández-Lobato, Michael Gelbart, Matthew Hoffman, Ryan Adams, and Zoubin Ghahramani. Predictive entropy search for bayesian optimization with unknown constraints. In *International conference on machine learning*, pp. 1699–1707. PMLR, 2015.

Hisao Ishibuchi, Yu Setoguchi, Hiroyuki Masuda, and Yusuke Nojima. Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 21(2):169–190, 2016.

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

Joshua Knowles. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE transactions on evolutionary computation*, 10(1): 50–66, 2006.

Mina Konakovic Lukovic, Yunsheng Tian, and Wojciech Matusik. Diversity-guided multi-objective bayesian optimization with batch evaluations. *Advances in neural information processing systems*, 33:17708–17720, 2020.

Jouni Lampinen and Ivan Zelinka. Mixed integer-discrete-continuous optimization by differential evolution. In *Proceedings of the 5th international conference on soft computing*, volume 71, pp. 76. Citeseer, 1999.

Marco Laumanns and Jiri Ocenasek. Bayesian optimization algorithms for multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, pp. 298–307. Springer, 2002.

Bingdong Li, Zixiang Di, Yongfan Lu, Hong Qian, Feng Wang, Peng Yang, Ke Tang, and Aimin Zhou. Expensive multi-objective bayesian optimization based on diffusion models. *arXiv preprint arXiv:2405.08674*, 2024.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in neural information processing systems*, 32, 2019.

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. *arXiv preprint arXiv:2203.15386*, 2022a.

Xi Lin, Zhiyuan Yang, Xiaoyuan Zhang, and Qingfu Zhang. Pareto set learning for expensive multiobjective optimization. *Advances in Neural Information Processing Systems*, 35:19231–19247, 2022b.

Xi Lin, Xiaoyuan Zhang, Zhiyuan Yang, and Qingfu Zhang. Dealing with structure constraints in evolutionary pareto set learning. *IEEE Transactions on Evolutionary Computation*, 29(3):616–630, 2025.

Hai-Lin Liu, Fangqing Gu, and Qingfu Zhang. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE transactions on evolutionary computation*, 18(3):450–455, 2013.

Jiao Liu, Abhishek Gupta, Chinchun Ooi, and Yew-Soon Ong. Extremo: Transfer evolutionary multiobjective optimization with proof of faster convergence. *IEEE Transactions on Evolutionary Computation*, 29(1):102–116, 2024a.

Jiao Liu, Yew Soon Ong, and Melvin Wong. Finding sets of pareto sets in real-world scenarios–a multitask multiobjective perspective. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE, 2024b.

Jiao Liu, Yong Wang, Guangyong Sun, and Tong Pang. Constrained evolutionary bayesian optimization for expensive constrained optimization problems with inequality constraints. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 55(3):2009–2021, 2024c.

Jiao Liu, Abhishek Gupta, and Yew-Soon Ong. Bayesian inverse transfer in evolutionary multiobjective optimization. *ACM Transactions on Evolutionary Learning*, 4(4):1–27, 2025.

Andre KY Low, Flore Mekki-Berrada, Abhishek Gupta, Aleksandr Ostudin, Jiaxun Xie, Eleonore Vissol-Gaudin, Yee-Fun Lim, Qianxiao Li, Yew Soon Ong, Saif A Khan, et al. Evolution-guided bayesian optimization for constrained multi-objective optimization in self-driving labs. *npj Computational Materials*, 10(1):104, 2024.

Chengyu Lu, Zhenhua Li, Xi Lin, Ji Cheng, and Qingfu Zhang. Neural evolution strategy for blackbox pareto set learning. *Advances in Neural Information Processing Systems*, 38:171463–171495, 2026.

Benjamin P MacLeod, Fraser GL Parlane, Connor C Rupnow, Kevan E Dettelbach, Michael S Elliott, Thomas D Morrissey, Ted H Haley, Oleksii Proskurin, Michael B Rooney, Nina Taherimakhsousi, et al. A self-driving laboratory advances the pareto front for material properties. *Nature Communications*, 13(1):995, 2022.

Alan Tan Wei Min, Yew-Soon Ong, Abhishek Gupta, and Chi-Keong Goh. Multiproblem surrogates: Transfer evolutionary multiobjective optimization of computationally expensive problems. *IEEE Transactions on Evolutionary Computation*, 23(1):15–28, 2017.

Yew S Ong, Prasanth B Nair, and Andrew J Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41(4):687–696, 2003.

Yew-Soon Ong. Max-min surrogate-assisted evolutionary algorithm for robust design. *IEEE Transactions on Evolutionary Computation*, 10(4):392–404, 2006.

Valerio Perrone, Iaroslav Shcherbatyi, Rodolphe Jenatton, Cedric Archambeau, and Matthias Seeger. Constrained bayesian optimization with max-value entropy search. *arXiv preprint arXiv:1910.07003*, 2019.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Tapabrata Ray and KM Liew. A swarm metaphor for multiobjective design optimization. *Engineering optimization*, 34(2):141–153, 2002.

Yinuo Ren, Tesi Xiao, Tanmay Gangwani, Anshuka Rangi, Holakou Rahmanian, Lexing Ying, and Subhajit Sanyal. Multi-objective optimization via wasserstein-fisher-rao gradient flow. In *International Conference on Artificial Intelligence and Statistics*, pp. 3862–3870. PMLR, 2024.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Filippo Santambrogio. {Euclidean, metric, and Wasserstein} gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7(1):87–154, 2017.

Hans-Paul Paul Schwefel. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

Chin Sheng Tan, Abhishek Gupta, Yew-Soon Ong, Mahardhika Pratama, Puay Siew Tan, and Siew Kei Lam. Pareto optimization with small data by learning across common objective spaces. *Scientific Reports*, 13(1):7842, 2023.

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

Handing Wang, Yaochu Jin, Cuie Yang, and Licheng Jiao. Transfer stacking from low-to high-fidelity: A surrogate-assisted bi-fidelity evolutionary algorithm. *Applied Soft Computing*, 92: 106276, 2020.

Handing Wang, Liang Feng, Yaochu Jin, and John Doherty. Surrogate-assisted evolutionary multitasking for expensive minimax optimization in multiple scenarios. *IEEE Computational Intelligence Magazine*, 16(1):34–48, 2021.

Tingyang Wei, Jiao Liu, Abhishek Gupta, Puay Siew Tan, and Yew-Soon Ong. Bayesian forward-inverse transfer for multiobjective optimization. In *International Conference on Parallel Problem Solving from Nature*, pp. 135–152. Springer, 2024.

Tingyang Wei, Jiao Liu, Abhishek Gupta, Chin Chun Ooi, Puay Siew Tan, and Yew-Soon Ong. Parametric expensive multi-objective optimization via generative solution modeling. *arXiv preprint arXiv:2511.09598*, 2025a.

Tingyang Wei, Jiao Liu, Abhishek Gupta, Puay Siew Tan, and Yew-Soon Ong. (θ, θ)-parametric multi-task optimization: Joint search in solution and infinite task spaces. *IEEE Transactions on Evolutionary Computation*, 2025b.

Joel Wendin and Claudio Altafini. Gradient flow equations for deep linear neural networks: A survey from a network perspective. *SIAM Review*, 68(2):293–345, 2026.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Jian Cheng Wong, Abhishek Gupta, Chin Chun Ooi, Pao-Hsiung Chiu, Jiao Liu, and Yew-Soon Ong. Evolutionary optimization of physics-informed neural networks: Evo-pinn frontiers and opportunities. *IEEE Computational Intelligence Magazine*, 21(1):16–36, 2026a.

Jian Cheng Wong, Isaac Yin Chung Lai, Pao-Hsiung Chiu, Chin Chun Ooi, Abhishek Gupta, and Yew-Soon Ong. Transferable physics-informed representations via closed-form head adaptation. *arXiv preprint arXiv:2604.21761*, 2026b.

Jian Cheng Wong, Chin Chun Ooi, Abhishek Gupta, Pao-Hsiung Chiu, Joshua Shao Zheng Low, My Ha Dao, and Yew-Soon Ong. Evolutionary optimization of physics-informed neural networks: Advancing generalizability by the baldwin effect. *IEEE Transactions on Evolutionary Computation*, 2026c.

Meng Xu, Yi Mei, Fangfang Zhang, Yew Soon Ong, and Mengjie Zhang. Pareto set learning through genetic programming for multi-objective dynamic scheduling. *IEEE Transactions on Evolutionary Computation*, 2025.

Rongguang Ye, Longcan Chen, Jinyuan Zhang, and Hisao Ishibuchi. Evolutionary preference sampling for pareto set learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 630–638, 2024.

Dawei Zhan, Yuansheng Cheng, and Jun Liu. Expected improvement matrix-based infill criteria for expensive multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 21(6): 956–975, 2017. doi: 10.1109/TEVC.2017.2697503.

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

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

Xiaoyuan Zhang, Xi Lin, Bo Xue, Yifan Chen, and Qingfu Zhang. Hypervolume maximization: A geometric view of pareto set learning. *Advances in Neural Information Processing Systems*, 36: 38902–38929, 2023.

Xiaoyuan Zhang, Liang Zhao, Yingying Yu, Xi Lin, Yifan Chen, Han Zhao, and Qingfu Zhang. Libmoon: A gradient-based multiobjective optimization library in pytorch. *Advances in Neural Information Processing Systems*, 2024.

Zhiyao Zhang, Yong Wang, Guangyong Sun, and Tong Pang. A distribution information-based kriging-assisted evolutionary algorithm for expensive many-objective optimization problems. *IEEE Transactions on Evolutionary Computation*, 29(6):2656–2670, 2025a. doi: 10.1109/TEVC.2024.3519185.

Zhiyao Zhang, Yong Wang, Guangyong Sun, Tong Pang, and Ke Tang. Constrained probabilistic pareto dominance for expensive constrained multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 29(4):1138–1152, 2025b. doi: 10.1109/TEVC.2024.3394005.

Zhiyao Zhang, Yong Wang, Guangyong Sun, and Tong Pang. A novel evolutionary bayesian optimization algorithm based on decomposition for expensive constrained multiobjective optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 56(2):945–957, 2026. doi: 10.1109/TSMC.2025.3637210.

Liang Zhao and Qingfu Zhang. Hypervolume-guided decomposition for parallel expensive multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 28(2):432–444, 2023.

A Related Work

This section reviews the literature most relevant to our work. We first introduce existing studies on multiobjective Bayesian optimization, followed by recent advances in Pareto set learning. We then discuss the connections between these two research directions and clarify how the proposed method differs from existing approaches.

A.1 MULTIOBJECTIVE BAYESIAN OPTIMIZATION

Bayesian optimization (Garnett, 2023) is an iterative surrogate-assisted optimization framework that is particularly suitable for expensive optimization problems, where each objective evaluation may require substantial computational resources or time. Bayesian optimization typically constructs a probabilistic surrogate model, most commonly a Gaussian process (Williams & Rasmussen, 2006), from previously evaluated solutions to approximate the objective landscape and quantify predictive uncertainty. At each iteration, an acquisition function (Jones et al., 1998) uses the surrogate predictions to select a promising candidate by balancing the exploration of uncertain regions and the exploitation of regions with favorable predicted objective values. The selected candidate is then evaluated using the true objective function, and the resulting observation is incorporated into the dataset to update the surrogate model.

Multiobjective Bayesian optimization (Laumanns & Ocenasek, 2002) extends this framework to problems involving multiple conflicting objectives, with the aim of identifying a set of solutions that provides a useful approximation to the Pareto front. Since improvement can no longer be characterized by a single objective value, Multiobjective Bayesian optimization acquisition functions are commonly designed using decomposition (Knowles, 2006; Zhang et al., 2009), performance indicators (Emmerich et al., 2006), or entropy perspective (Belakaria et al., 2019) to quantify the potential contribution of a candidate to the current approximation set. Different acquisition designs can further emphasize specific algorithmic properties, such as promoting constrained optimization (Feliot et al., 2017; Zhang et al., 2025b; Hernández-Lobato et al., 2015; Perrone et al., 2019; Liu et al., 2024c), many-objective optimization (Binois et al., 2020; Zhan et al., 2017; Zhang et al., 2025a; 2026) or enabling the simultaneous evaluation of multiple candidates in batch settings (Daulton et al., 2020; 2021; Konakovic Lukovic et al., 2020; Hernández-Lobato et al., 2016).

Nevertheless, the effectiveness of Bayesian optimization relies heavily on the quality of its probabilistic surrogate. In particular, a Gaussian process requires sufficiently informative observations to provide reliable predictions of objective values and meaningful estimates of predictive uncertainty. Under a limited evaluation budget, however, the available observations may be too sparse to accurately characterize complex objective landscapes, especially in multiobjective optimization, where several objective functions must be modeled and a diverse approximation of the Pareto front must be identified. Consequently, multiobjective Bayesian optimization may exhibit limited sample efficiency during its early search stage, before sufficiently informative data have been collected. To alleviate this limitation, recent years have witnessed increasing interest in transfer Bayesian optimization approaches for expensive optimization problems (Bai et al., 2023). These methods complement the limited observations from the target problem with knowledge acquired from related tasks. Specifically, surrogate models (Ong et al., 2003; Ong, 2006) are employed to reduce the reliance on costly fitness evaluations, while transfer mechanisms reuse historical data, learned representations, or predictive models from multiple related tasks to construct more informative multitask or transfer-aware surrogates for evolutionary search (Min et al., 2017; Liu et al., 2024a; Wang et al., 2020; 2021). When useful correlations exist among the tasks, the transferred knowledge can improve surrogate modeling and facilitate search under stringent evaluation budgets. More recently, this idea has been extended to parametric optimization (Wei et al., 2025b; Cheng et al., 2026), where a potentially infinite family of related tasks is continuously defined over a parameter space and addressed within a unified optimization framework (Wei et al., 2025a).

A.2 PARETO SET LEARNING

The goal of multiobjective optimization is to identify a set of solutions that both converges toward and is well distributed along the Pareto front. Conventional multiobjective Bayesian optimization methods typically select a finite number of solutions associated with different trade-off preferences to approximate the PF. Under a limited evaluation budget, however, the resulting approximation may remain sparse and provide insufficient coverage of the PF, particularly for expensive optimization problems. Pareto set learning (Lin et al., 2022b; Zhang et al., 2023; Lin et al., 2022a; Lu et al., 2026; Lin et al., 2025; Ye et al., 2024; Li et al., 2024) offers a promising way to obtain a denser representation of the PF. It aims to learn a mapping from preference vectors, or equivalently points on the PF, to their corresponding solutions in the decision space. Recent online Pareto set learning methods have shown that the learned mapping can guide the search toward a denser PF approximation and, in some cases, improve optimization efficiency (Lin et al., 2022b).

Nevertheless, most existing Pareto set learning methods train the mapping using observations collected exclusively from the target task and therefore do not explicitly exploit knowledge available from related source tasks. (Tan et al., 2023) investigated inverse transfer for multiobjective optimization in a post-hoc setting, where source-task information was leveraged to improve the predictive accuracy of an inverse model after the optimization process. More recently, *inverse transfer* methods (Liu et al., 2025; Wei et al., 2024) have been developed for expensive multitask multiobjective optimization. These methods transfer information contained in the nondominated solution sets of related tasks to support Pareto set learning on a target task. When useful task correlations exist, such transfer can improve the data efficiency of target-task optimization and provide more informative training data for the final Pareto set model.

A.3 Distinctive Features of This Work.

Different from existing studies, this work proposes PFL, which learns preference-conditioned Pareto gradient flows from a collection of related source tasks and adapts the learned representation to an unseen target task. Compared with conventional multiobjective Bayesian optimization and Pareto set learning, the proposed framework has two distinctive features. First, unlike standard multiobjective Bayesian optimization, including transfer-based variants, PFL explicitly learns a continuous preference-conditioned model whose terminal states approximate the target Pareto set. Once adapted to the target task, the model can generate candidate Pareto optimal solutions for different preference vectors without requiring each candidate to be discovered independently through acquisition-function optimization. Second, unlike conventional Pareto set learning methods that are trained primarily using target-task information, the proposed framework leverages knowledge acquired from related source tasks through meta-learning. In particular, it transfers shared structure in the learned Pareto gradient flows to provide an informative initialization for target-task adaptation. When such structure is relevant to the target task, this can reduce the amount of target-specific information required to obtain a useful Pareto-set approximation and improve search efficiency under limited evaluation budgets.

B Implementation Details

Smoothed Estimation of the Pareto Gradient Field. Training mPFM with ODE residual supervision requires evaluating the gradient field

$$\nabla_{\mathbf{u}} f^{\mathrm{tch}}\left(\hat{\mathbf{u}}_\theta(\mathbf{w}, t); \mathbf{w}\right). \tag{16}$$

Directly using the exact objective gradient can lead to unstable supervision when the objective landscape is highly non-smooth or exhibits rapidly varying local curvature. Moreover, an overly localized gradient field may encourage the predicted trajectories to converge prematurely to nearby stationary points, which results in the sub-optimal of the terminal solutions generated for different preferences. To obtain a smoother training signal, we estimate the gradient of a Gaussian-smoothed scalarized objective using a zeroth-order evolution strategy estimator. Let

$$\mathbf{x}_t = \hat{\mathbf{u}}_\theta(\mathbf{w}, t) \tag{17}$$

denote the state predicted by mPFM. When constructing the target gradient field, we stop gradients through $\mathbf{x}_t$ and sample K independent perturbations

$$\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \qquad k = 1, \ldots, K. \tag{18}$$

The perturbed solutions are defined as

$$\mathbf{x}_{t,k} = [\mathbf{x}_t] + \sigma_0 \boldsymbol{\epsilon}_k, \tag{19}$$

where σ_0 is the smoothing scale. Each perturbed solution is evaluated using the augmented Tchebycheff scalarization:

$$s_k = f^{\mathrm{tch}}\left(\mathbf{x}_{t,k}; \mathbf{w}\right). \tag{20}$$

To reduce the variance of the estimator, we center the scalarized objective values within each set of perturbations:

$$\tilde{s}_k = s_k - \frac{1}{K} \sum_{j=1}^{K} s_j. \tag{21}$$

The gradient of the Gaussian-smoothed scalarized objective is then estimated by

$$\widehat{\nabla}_{\mathbf{u}} f^{\mathrm{tch}}(\mathbf{x}_t; \mathbf{w}) = \frac{1}{K\sigma_0} \sum_{k=1}^{K} \tilde{s}_k \boldsymbol{\epsilon}_k. \tag{22}$$

Finite Difference Approximation of the Temporal Derivative. The ODE residual also requires the temporal derivative $\partial \hat{\mathbf{u}}_\theta(\mathbf{w}, t)/\partial t$. Computing this derivative by automatic differentiation would require backpropagating through the time derivative during optimization, which introduces higher-order derivatives of the network output with respect to its parameters. To reduce the resulting computational and memory overhead, we approximate the temporal derivative using a forward finite difference (Chiu et al., 2022). For a small step size $\Delta t > 0$, the temporal derivative is approximated as

$$\frac{\partial \hat{\mathbf{u}}_\theta(\mathbf{w}, t)}{\partial t} \approx \frac{\hat{\mathbf{u}}_\theta(\mathbf{w}, t + \Delta t) - \hat{\mathbf{u}}_\theta(\mathbf{w}, t)}{\Delta t}. \tag{23}$$

Final Implementation of the Loss Function. Substituting this approximation into the Pareto gradient flow equation gives

$$\hat{\mathbf{u}}_\theta(\mathbf{w}, t + \Delta t) - \hat{\mathbf{u}}_\theta(\mathbf{w}, t) \approx -\Delta t\, \widehat{\nabla}_{\mathbf{u}} f^{\mathrm{tch}}_{\sigma(t)}\left(\hat{\mathbf{u}}_\theta(\mathbf{w}, t); \mathbf{w}\right), \tag{24}$$

where $\widehat{\nabla}_{\mathbf{u}} f^{\mathrm{tch}}_{\sigma(t)}$ denotes the smoothed gradient estimator introduced above. Accordingly, we calculate the estimated ODE residual as

$$\mathcal{L}_{\mathrm{DE}} = \mathbb{E}_{\mathbf{w},t}\left[\left\| \hat{\mathbf{u}}_\theta(\mathbf{w}, t + \Delta t) - \hat{\mathbf{u}}_\theta(\mathbf{w}, t) + \Delta t\, \widehat{\nabla}_{\mathbf{u}} f^{\mathrm{tch}}_{\sigma(t)}\left(\hat{\mathbf{u}}_\theta(\mathbf{w}, t); \mathbf{w}\right) \right\|_2^2\right]. \tag{25}$$

By substituting the estimated ODE residual into equation 4 and equation 10, we obtain the final training loss used in our implementation.

C Parameter Settings

C.1 Benchmark Problem Settings.

We evaluate mPFM-MOBO on both synthetic benchmark families and engineering design problems. The synthetic test suite consists of the mDTLZ and mDTLZ^{-1} families, with each problem containing three objectives and eight decision variables. Each task is parameterized by δ_1 and δ_2, where δ_1 controls the transformation of the decision variables and affects the geometry and coverage difficulty of the Pareto set, while δ_2 shifts the center of the distance function g. During meta-training, source tasks are independently generated by sampling $\delta_1 \sim \mathcal{U}(0.7, 1.0)$ and $\delta_2 \sim \mathcal{U}(-0.25, 0.25)$. At each training epoch, five parameter pairs are sampled to construct five source tasks. Bayesian optimization is then evaluated on both an in-distribution target task and an out-of-distribution target task. Specifically, the in-distribution target is defined by $(\delta_1, \delta_2) = (0.7, 0.25)$, which lies on the boundary of the source-task distribution, whereas the out-of-distribution target is defined by $(\delta_1, \delta_2) = (0.6, 0.35)$, for which both parameters fall outside their corresponding training ranges. The same parameter settings are used for the mDTLZ and mDTLZ^{-1} families.

We further consider four engineering design benchmark problems: four-bar truss design (Cheng & Li, 1999), hatch cover design (Amir & Hasegawa, 1989), coil compression spring design (Lampinen & Zelinka, 1999), and welded beam design (Ray & Liew, 2002). Each problem is extended into a family of related tasks by varying selected physical or operating parameters. At each meta-training epoch, five source tasks are independently sampled from the corresponding uniform parameter distributions. For evaluation, we consider both an in-distribution target task, whose parameters lie within the source-task ranges, and an out-of-distribution target task, whose parameters lie outside these ranges. The detailed settings are as follows:

- The four-bar truss problem contains two objectives and four decision variables. Its source tasks are generated by sampling the external load as $F \sim \mathcal{U}(8, 10)$ kN, the allowable stress as $\sigma \sim \mathcal{U}(8, 10)$ kN/cm^2, and Young's modulus as $E \sim \mathcal{U}(1.5 \times 10^5, 2.0 \times 10^5)$ kN/cm^2, while the member length is fixed at $L = 200$ cm. The in-distribution target task is defined by $F = 9$ kN, $\sigma = 9$ kN/cm^2, and $E = 1.75 \times 10^5$ kN/cm^2. The out-of-distribution target task uses $F = 11$ kN, $\sigma = 11$ kN/cm^2, and $E = 2.2 \times 10^5$ kN/cm^2. The member length remains fixed at $L = 200$ cm for both targets.

Table 1: Normalization boundary parameters used for HV computation on the engineering design problems.

Problem	Target	$\mathbf{y}_{\min}$	$\mathbf{y}_{\max}$
Four Bar Truss Design	In-Distribution	$(1237.8,\ 2.84 \times 10^{-3})$	$(2886.4,\ 4.11 \times 10^{-2})$
	Out-of-Distribution	$(1237.8,\ 2.76 \times 10^{-3})$	$(2886.4,\ 4.00 \times 10^{-2})$
Hatch Cover Design	In-Distribution	$(480.5,\ 0)$	$(481.88,\ 3.25)$
	Out-of-Distribution	$(480.5,\ 0)$	$(481.50,\ 2.20)$
Coil Compression Spring Design	In-Distribution	$(0,\ 0)$	$(2.0,\ 3.5 \times 10^{6})$
	Out-of-Distribution	$(0,\ 0)$	$(2.0,\ 3.5 \times 10^{6})$
Welded Beam Design	In-Distribution	$(0.0102,\ 2.76 \times 10^{-4},\ 0)$	$(34.9,\ 1.11 \times 10^{4},\ 3.04 \times 10^{8})$
	Out-of-Distribution	$(0.0102,\ 5.48 \times 10^{-4},\ 0)$	$(35.5,\ 2.19 \times 10^{4},\ 4.93 \times 10^{8})$

- The hatch cover problem contains two objectives and two decision variables. Its source tasks are generated by sampling Young's modulus as $E \sim \mathcal{U}(5.0 \times 10^5, 7.0 \times 10^5)$, the allowable bending stress as $\sigma_b^{\max} \sim \mathcal{U}(500, 700)$, and the maximum deflection as $\delta^{\max} \sim \mathcal{U}(1.5, 2.0)$. The in-distribution target task is specified by $E = 6.0 \times 10^5$, $\sigma_b^{\max} = 600$, and $\delta^{\max} = 1.75$, whereas the out-of-distribution target task uses $E = 7.5 \times 10^5$, $\sigma_b^{\max} = 750$, and $\delta^{\max} = 2.5$.
- The coil compression spring design problem contains two objectives and three decision variables. Its source tasks are generated by sampling the maximum working load as $F_{\max} \sim \mathcal{U}(900, 1100)$ N and the preload as $F_p \sim \mathcal{U}(250, 350)$ N. The in-distribution target task uses $F_{\max} = 1000$ N and $F_p = 300$ N, whereas the out-of-distribution target task is defined by $F_{\max} = 1200$ N and $F_p = 400$ N.
- Finally, the welded beam design problem contains three objectives and four decision variables. Its source tasks are generated by sampling the applied load as $P \sim \mathcal{U}(4000, 6000)$ lb, the beam length as $L \sim \mathcal{U}(10, 14)$ in, and Young's modulus as $E \sim \mathcal{U}(2.0 \times 10^7, 3.0 \times 10^7)$ psi. The in-distribution target task is defined by $P = 5000$ lb, $L = 12$ in, and $E = 2.5 \times 10^7$ psi. The corresponding out-of-distribution target task is specified by $P = 6500$ lb, $L = 15$ in, and $E = 3.2 \times 10^7$ psi.

C.2 Performance Metric Settings

We use the HV indicator as the primary metric for evaluating the convergence and coverage of the obtained Pareto-front approximation. All benchmark problems are formulated as minimization problems, and a larger HV indicates a better nondominated solution set under the specified normalization bounds and reference point. To account for differences in objective scales, HV is computed in a normalized objective space for all synthetic and engineering benchmarks.

For each target task, let $\mathbf{y}_{\min}$ and $\mathbf{y}_{\max}$ denote the component-wise lower and upper normalization bounds, respectively. Each objective vector $\mathbf{y}$ is normalized as $\widetilde{\mathbf{y}} = (\mathbf{y} - \mathbf{y}_{\min})/(\mathbf{y}_{\max} - \mathbf{y}_{\min} + 10^{-10})$. Under this transformation, objective values close to $\mathbf{y}_{\min}$ are mapped close to zero, whereas values close to $\mathbf{y}_{\max}$ are mapped close to one. HV is then computed in the normalized objective space using the common reference point $\widetilde{\mathbf{r}} = (1.1, \ldots, 1.1)$.

For mDTLZ1, we set the normalization boundary parameters to $\mathbf{y}_{\min} = 0$ and $\mathbf{y}_{\max} = 0.5$. For mDTLZ2–mDTLZ4, these parameters are set to $\mathbf{y}_{\min} = 0$ and $\mathbf{y}_{\max} = 1$. For the inverse variants, the corresponding boundary parameters are set to $\mathbf{y}_{\min} = -0.5$ and $\mathbf{y}_{\max} = 0$ for mDTLZ1^{-1}, and $\mathbf{y}_{\min} = -1$ and $\mathbf{y}_{\max} = 0$ for mDTLZ2^{-1}–mDTLZ4^{-1}. For both the in-distribution and out-of-distribution settings, Bayesian optimization is initialized with 20 evaluations, followed by 80 optimization iterations, resulting in a total evaluation budget of 100. For the engineering design problems, the settings of the boundary parameters are shown in Table 1.

C.3 mPFM Model Settings

The mPFM takes a preference vector and a time variable as inputs. For the synthetic benchmarks, the input dimension is four, consisting of a three-dimensional preference vector and one temporal coordinate. The shared backbone is a multilayer perceptron with hidden dimensions $[256, 128, 64]$ and ReLU activations. Both the data-dependent task embedding $\mathbf{e}_{\text{data}}$ and the optimizable task embedding $\mathbf{e}_{\text{opt}}$ have dimension 16. The output dimension is equal to the number of decision

variables of the corresponding problem. The terminal time is set to $t_{\max} = 2.0$. The finite-difference step is $\Delta t = 0.01$ for the synthetic benchmarks and $\Delta t = 0.02$ for the engineering problems. At each training step, 100 preference vectors are sampled. The smoothed gradient field is estimated using 32 Gaussian perturbations for the synthetic problems and 16 perturbations for the engineering problems. The initial perturbation standard deviation is set to $\sigma_0 = 0.2$ and remains constant during training. In terms of the loss function, the settings of the weights are as follows:$\lambda_{\mathrm{IC}} = 0.1, \lambda_{\mathrm{ODE}} = 20, \lambda_{\mathrm{Ter}} = 0.1, \lambda_{\mathrm{Reg}} = 0.01$. The augmentation coefficient in the Tchebycheff scalarization is set to $\rho = 10^{-4}$. mPFM is trained for 10,000 epochs using Adam with a learning rate of 10^{-4}. Five source tasks are sampled in each epoch. The task-specific embedding is adapted for 10 inner-loop steps with an inner learning rate of 10^{-4}. For the engineering problems, the gradient norm is clipped at 10 to improve training stability. Unless otherwise stated, standard normalized preference vectors are used in the augmented Tchebycheff scalarization.

C.4 mPFM-MOBO Settings

For each target task, Bayesian optimization starts from 20 solutions generated by Latin hypercube sampling and terminates after 100 objective evaluations. An independent Gaussian process is fitted to each objective by utilizing the `gpytorch` library (Gardner et al., 2018). Each GP uses a constant mean function and an isotropic radial basis function kernel. The GP inputs are normalized to $[0, 1]$, and the kernel length scale is constrained to $[10^{-4}, 10]$. A Gaussian likelihood is used with its noise variance constrained to $[10^{-5}, 1]$ and initialized at 10^{-4}. The GP hyperparameters are optimized by minimizing the negative marginal log likelihood using L-BFGS for at most 200 iterations, with a convergence tolerance of 10^{-6}. If numerical instability occurs, the likelihood noise is increased to 10^{-2} and the fitting procedure is repeated. At each Bayesian optimization iteration, the shared mPFM parameters are fixed, while the target-specific embedding $\mathbf{e}_{\mathrm{opt}}$ is adapted for 500 gradient steps using Adam with a learning rate of 10^{-4}. During the finetuning of $\mathbf{e}_{\mathrm{opt}}$, 20 preference vectors are sampled at each step, and the task embedding is optimized using the Tchebycheff scalarization of the GP lower confidence bounds with $\beta = 0.1$. After the finetuning, mPFM generates a pool of 1,000 preference-conditioned candidate solutions. Unless otherwise stated, one candidate is selected at each iteration using greedy hypervolume improvement based on the LCB values.

D Visualization of the Learned Pareto Flow Dynamics

In this subsection, we further visualize the Pareto flow dynamics learned by mPFM. Specifically, we consider three representative problems, namely mDTLZ2, mDTLZ2^{-1}, and the four-bar truss design problem. For each problem, we first solve the corresponding in-distribution target task using mPFM-MOBO to obtain the finetuned mPFM model. We then sample four representative time steps to illustrate the evolution of the learned flow. Figure 6(a) shows the objective values of the solutions queried by a set of preference vectors $\mathbf{w}$ at different time steps. Dashed lines connect the solutions generated by the same preference vector, illustrating their trajectories in the objective space. At $t = 0$, the queried solutions are close to the initialization (i.e., $(0, 0, \ldots, 0)$ in the decision space), resulting in objective values located in the lower-right region of the objective space. As the flow evolves to $t = 0.990$, the solutions progressively move toward the true Pareto front, indicating that the learned flow guides the solutions toward regions with improved objective values. At $t = 1.697$, the trajectories continue to spread along the Pareto front, leading to a wider coverage of different trade-offs. Finally, at $t = 2$, the generated solutions closely approximate the entire Pareto front. These observations provide an intuitive visualization of the optimization dynamics captured by the learned Pareto flow. Similar behaviors can also be observed for mDTLZ2^{-1} and the four-bar truss design problem, as shown in Figures 6(b) and 6(c). In both cases, solutions associated with different preference vectors gradually evolve toward the Pareto front while expanding their coverage over different trade-offs.

E Visualization of the Learned Pareto Sets

In this subsection, we further visualize the Pareto sets learned by mPFM. Specifically, we uniformly sample 5,000 preference vectors $\mathbf{w}$ from the simplex and use the learned mPFM to predict the terminal solutions at $t = 2$. These solutions are then evaluated by the objective functions and plotted in

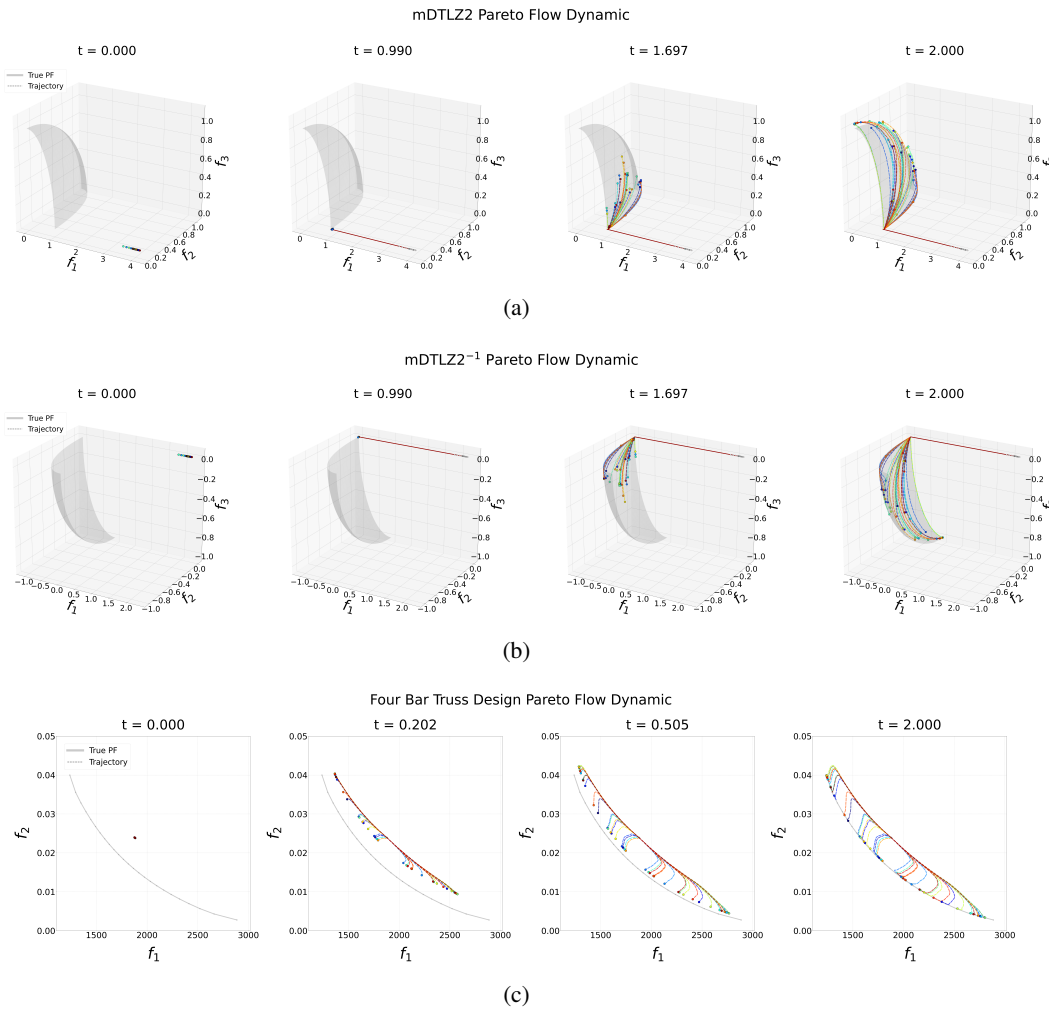


Figure 6: Pareto flow dynamics learned by mPFM, visualized in the objective space. (a) Pareto flow dynamics for the in-distribution mDTLZ2 task. (b) Pareto flow dynamics for the in-distribution mDTLZ2^{-1} task. (c) Pareto flow dynamics for the in-distribution four-bar truss design task.

the objective space to illustrate the distribution induced by the learned Pareto set. Figure 7 presents the resulting objective-space distributions. It can be observed that, across all benchmark problems, the generated solutions closely approximate the true Pareto fronts while maintaining a nearly uniform coverage over different trade-offs. These results suggest that mPFM is able to learn a Pareto set that generalizes well over the preference space. We also observe that the learned Pareto sets are not perfectly accurate for every problem. For example, on mDTLZ1, mDTLZ1^{-1}, mDTLZ4^{-1}, and the hatch cover design problem, the generated solutions do not completely cover the entire Pareto front, indicating a certain approximation error in the learned mapping. Nevertheless, considering that the target problem is optimized with a budget of only 100 evaluations, the learned Pareto sets still provide a good approximation of the underlying Pareto fronts. These results are consistent with the effectiveness of the proposed framework and further demonstrate that the learned prior can be efficiently adapted to new multiobjective optimization tasks with limited evaluations.

F Ablation Study on the ODE Residual Supervision

To isolate the contribution of learning Pareto gradient flow dynamics, we compare mPFM-MOBO with a Meta-PSL variant. Meta-PSL uses the same network architecture as mPFM, but removes

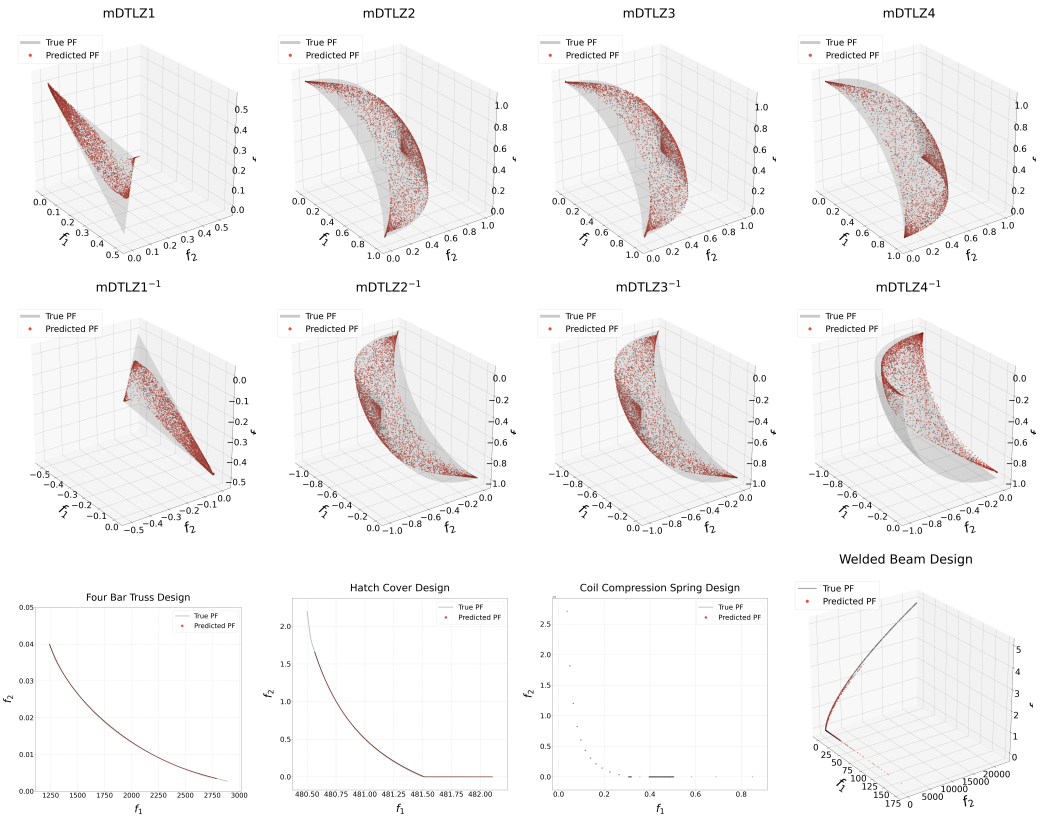
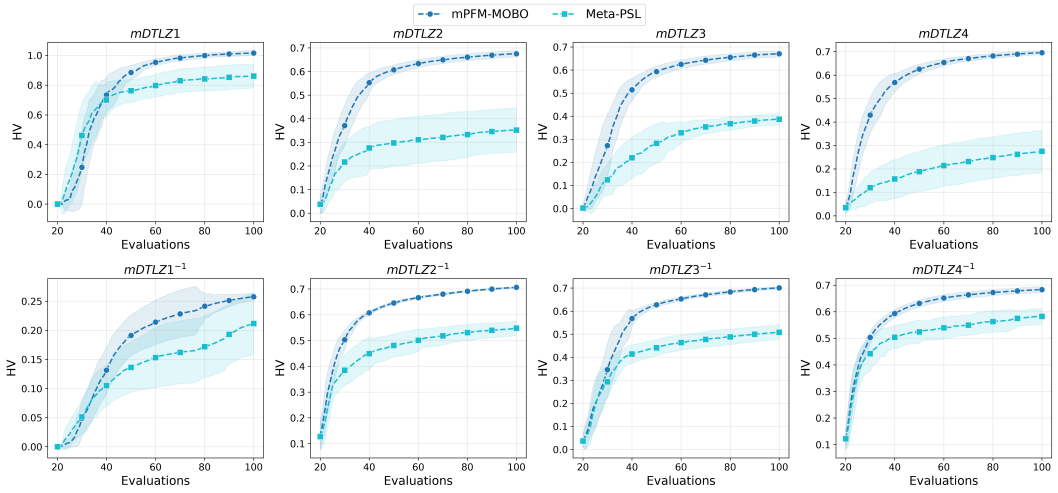


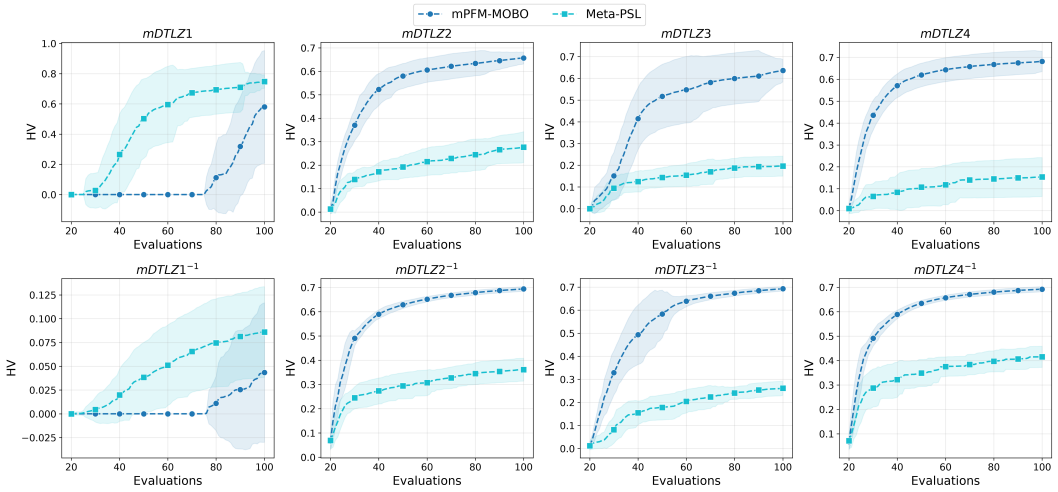
Figure 7: The Pareto fronts induced by the Pareto sets learned by mPFM. We uniformly sample 5,000 preference vectors $\mathbf{w}$ from the simplex and use the learned mPFM to generate the corresponding terminal solutions at $t = 2$. The generated solutions are evaluated using the objective functions and visualized in the objective space.

the Pareto gradient flow supervision from the training loss. It therefore only learns the terminal preference-to-solution mapping from source-task Pareto sets, without explicitly modeling preference-conditioned gradient-flow trajectories. As shown in Figure 8, removing the Pareto gradient flow supervision leads to substantial performance degradation in both in-distribution and out-of-distribution settings. Meta-PSL is significantly worse than mPFM-MOBO on all benchmark problems except the out-of-distribution mDTLZ1 and mDTLZ1^{-1} tasks. This suggests that, in the meta-learning setting, supervision from terminal Pareto sets alone may be insufficient for learning a transferable representation across tasks. Although related tasks may share similar optimization structures, their final Pareto sets can still be geometrically misaligned due to task-specific transformations, shifts, or variations in objective landscapes. As a result, directly fitting only the terminal Pareto optimal solutions provides a sparse and weak supervision signal. This makes it difficult for the meta-model to discover a common structure that can be effectively adapted to a new task through task conditioning. In contrast, Pareto flow trajectories provide process-level supervision for meta-learning. For each preference, the trajectory describes how a candidate solution moves toward the corresponding Pareto optimal solution under the objective-induced descent dynamics. These trajectories serve as preference-conditioned anchors that align source tasks not only by their final solutions, but also by the optimization paths leading to them. Consequently, mPFM can learn a shared dynamical representation that captures local descent directions and interactions among objectives, thereby improving transferability and reducing the risk of unstable or collapsed meta-learning.

To further examine when ODE residual supervision is beneficial, we investigate how the source-task distribution used for meta-training affects the performance of mPFM. Specifically, we construct three source-task groups with different levels of task diversity. Group 1 contains mDTLZ1–mDTLZ4, using the same distributions of δ_1 and δ_2 as in the main experiments. Group 2 contains



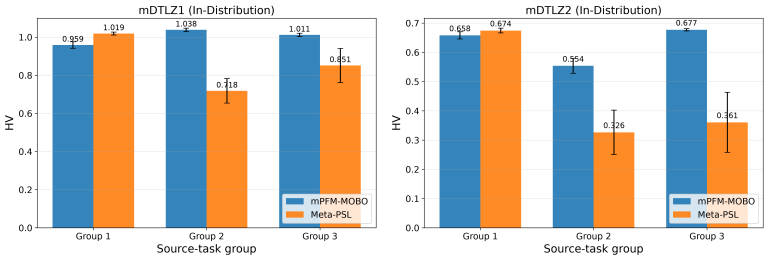
(a)



(b)

Figure 8: Comparison of the HV convergence trajectories of mPFM-MOBO and Meta-PSL on in-distribution and out-of-distribution synthetic benchmark problems. Shaded areas represent one standard deviation on either side of the mean. (a) Comparison on in-distribution synthetic benchmark problems. (b) Comparison on out-of-distribution synthetic benchmark problems.

mDTLZ1, mDTLZ2, mDTLZ1^{-1}, and mDTLZ2^{-1}, while Group 3 follows the main experimental setting and uses all eight benchmark families to generate source tasks. For each group, we train both mPFM and Meta-PSL and evaluate their downstream MOBO performance on in-distribution target tasks generated from mDTLZ1 and mDTLZ2. Figure 9 reports the resulting HV values. An interesting pattern emerges as the diversity of the source-task distribution increases. Under Group 1, Meta-PSL is competitive with, and in some cases outperforms, mPFM on the mDTLZ1 and mDTLZ2 target tasks. This is plausible because the source tasks in this group are relatively similar; in particular, mDTLZ2–mDTLZ4 share closely related Pareto-front geometries. In this setting, learning a shared terminal Pareto-set representation may already provide a sufficiently consistent meta-learning signal, leaving less room for additional flow supervision. In contrast, under Groups 2 and 3, where the inclusion of inverse mDTLZ families increases the diversity of the source-task distribution, mPFM provides substantially stronger guidance for downstream MOBO than Meta-PSL on both target tasks. Greater source-task diversity makes it more challenging to learn a common representation from terminal Pareto sets alone, whose locations and geometries can vary across



(a) (b)

Figure 9: Final HV of mPFM-MOBO and Meta-PSL on in-distribution (a) mDTLZ1 and (b) mDTLZ2 target tasks, under different groups of source task settings used for meta training.

Table 2: Ablation study on the task embedding vectors: mPFM-MOBO vs. mPFM-MOBO-w/o-FT vs. mPFM-MOBO-Fixed.

Problem	In-Distribution Tasks			Out-of-Distribution Tasks		
	mPFM-MOBO	mPFM-MOBO-w/o-FT	mPFM-MOBO-Fixed	mPFM-MOBO	mPFM-MOBO-w/o-FT	mPFM-MOBO-Fixed
	Avg. HV±Std.	Avg. HV±Std.	Avg. HV±Std.	Avg. HV±Std.	Avg. HV±Std.	Avg. HV±Std.
mDTLZ1	**1.0157 ± 0.0172**	0.0009 ± 0.0040	0.0000 ± 0.0000	**0.5804 ± 0.3871**	0.0000 ± 0.0000	0.0000 ± 0.0000
mDTLZ2	**0.6754 ± 0.0138**	0.3932 ± 0.0478	0.2443 ± 0.0655	**0.6570 ± 0.0259**	0.0876 ± 0.0327	0.0248 ± 0.0208
mDTLZ3	**0.6708 ± 0.0142**	0.0674 ± 0.0436	0.0055 ± 0.0064	**0.6360 ± 0.0529**	0.0000 ± 0.0000	0.0000 ± 0.0000
mDTLZ4	**0.6958 ± 0.0093**	0.3714 ± 0.0541	0.2423 ± 0.0696	**0.6817 ± 0.0478**	0.0774 ± 0.0366	0.0274 ± 0.0244

tasks. Pareto-flow supervision introduces additional constraints along the intermediate optimization trajectories, providing complementary information for learning shared structure across these more heterogeneous tasks. These results suggest that the benefit of ODE residual supervision becomes more evident as the source-task distribution becomes more diverse.

G Ablation Study on the Task Embedding Vectors

A key component of mPFM-MOBO is the target-task embedding, consisting of $\mathbf{e}^{\tau}_{\text{data}}$ and $\mathbf{e}^{\tau}_{\text{opt}}$, which provides task-specific conditioning for the pretrained mPFM during target-task optimization. To investigate the contribution of these two embeddings, we consider two ablated variants: 1) mPFM-MOBO-w/o-FT uses the same pretrained mPFM model but keeps $\mathbf{e}^{\tau}_{\text{opt}}$ fixed at its zero initialization, while retaining the adaptation of $\mathbf{e}^{\tau}_{\text{data}}$; and 2) mPFM-MOBO-Fixed keeps both $\mathbf{e}^{\tau}_{\text{data}}$ and $\mathbf{e}^{\tau}_{\text{opt}}$ fixed at zero throughout optimization. We compare these variants with the full mPFM-MOBO on both in-distribution and out-of-distribution mDTLZ target tasks, with the results reported in Table 2. Across all mDTLZ target problems, mPFM-MOBO-w/o-FT performs consistently better than or comparably to mPFM-MOBO-Fixed, indicating that $\mathbf{e}^{\tau}_{\text{data}}$ alone already provides useful target-specific information for conditioning the meta-learned model. Further adapting $\mathbf{e}^{\tau}_{\text{opt}}$ yields an additional and substantial improvement: the full mPFM-MOBO achieves significantly higher HV than mPFM-MOBO-w/o-FT on every in-distribution and out-of-distribution task. This suggests that the data embedding captures information available from the observed target-task data, while the optimization embedding further adapts the transferred Pareto-flow representation to the target optimization landscape.

H Batch Acquisition Capability of mPFM-MOBO

Batch acquisition is an important capability in Bayesian optimization, as it enables multiple candidate solutions to be evaluated in parallel and can substantially reduce wall-clock time for expensive optimization problems. Similar to standard PSL-based multiobjective Bayesian optimization methods, mPFM-MOBO also supports the batch candidate acquisition. To evaluate its performance in this setting, we compare mPFM-MOBO with three batch-capable baselines, namely PSL-MOBO, DirHV-EGO, and PSL-DirHV-EGO. For all methods, the batch size is set to five, and the total evaluation budget is fixed at 100. Experiments are conducted on both in-distribution and out-of-

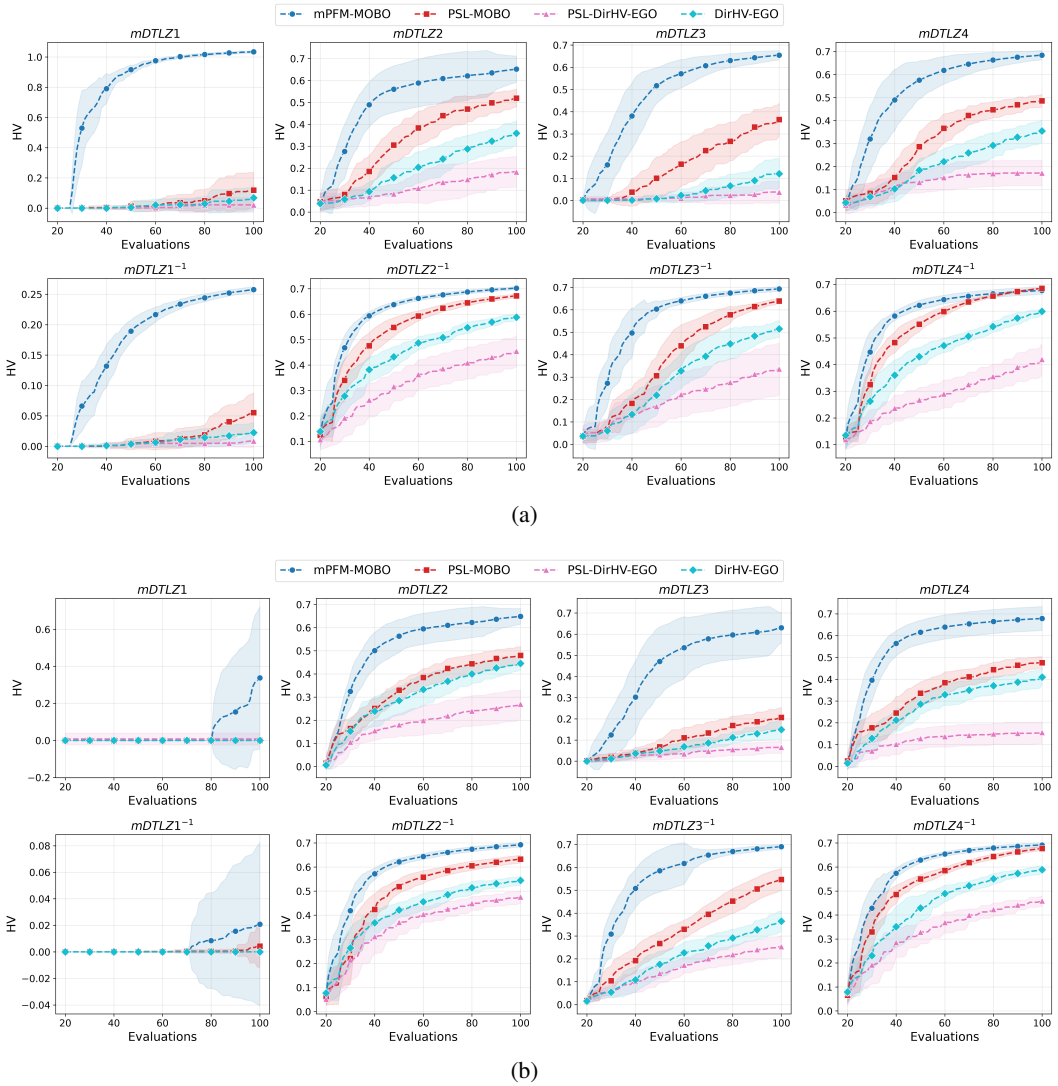


Figure 10: Comparison of the HV convergence trajectories of mPFM-MOBO, DirHV-EGO, PSL-MOBO, and PSL-DirHV-EGO on in-distribution and out-of-distribution synthetic benchmark problems under a batch size of five. Shaded areas represent one standard deviation on either side of the mean. (a) Comparison on in-distribution synthetic benchmark problems. (b) Comparison on out-of-distribution synthetic benchmark problems.

distribution synthetic benchmark problems. The resulting HV convergence trajectories are shown in Figure 10. From the result, mPFM-MOBO exhibits faster convergence than the compared baselines on synthetic benchmark problems and remains competitive in terms of final HV. The main exception is the in-distribution mDTLZ4^{-1} problem, where PSL-MOBO achieves a slightly higher final HV. Overall, these results indicate that mPFM-MOBO retains effective optimization performance under batch acquisition settings.