# IN-CONTEXT PREFERENCE-CONDITIONED PARETO SET LEARNING FROM LIMITED OFFLINE DATA

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Multi-objective optimization involves diverse preferences over conflicting objectives. Rather than returning a finite set of trade-off solutions, preference-conditioned Pareto set learning seeks to map a specified preference directly to a corresponding Pareto optimal solution. Yet in many real-world scenarios, optimization must proceed offline without further objective evaluations, leaving only a limited static dataset that sparsely covers the decision and objective spaces. Existing methods either optimize separately for each preference or train a problem-specific Pareto solution model, requiring repeated search across preferences or repeated training across problems. We introduce in-context preference-conditioned Pareto set learning (ICPSL), a framework that amortizes both preference-wise search and problem-specific model construction. ICPSL pretrains a foundation-style hypernetwork on a large and diverse set of synthetic prior data. Given the offline observations of an unseen problem, the pretrained hypernetwork learns to generate a problem-specific, preference-conditioned Pareto solution model in a single forward pass. The generated model can then be reused to predict solutions for arbitrary preferences without problem-specific training or preference-wise search. Experiments on synthetic and real-world problems show that ICPSL produces high-quality Pareto solutions while providing substantial speedups in problem-specific solution-model construction.

## 1 INTRODUCTION

Many real-world optimization problems involve multiple conflicting objectives, where improving one objective may necessarily compromise another (Ying et al., 2019; Tanabe & Ishibuchi, 2020; Dara et al., 2022). Consequently, there is generally no single optimal solution, the goal is to identify a Pareto set that characterizes the possible trade-offs among objectives. Conventional methods approximate this set with a finite collection of solutions, which may not include the solution that best matches a decision-maker's preferred trade-off (Deb et al., 2002; Beume et al., 2007). Preference-conditioned Pareto set learning (PSL) instead represents the trade-off structure as a model that maps an arbitrary preference $\boldsymbol{\lambda}$ to the corresponding optimal solution $\boldsymbol{x}^\star(\boldsymbol{\lambda})$ (Navon et al., 2021; Lin et al., 2022). By varying the specified preference, decision-makers can readily explore different trade-off regions of the approximate Pareto set, enabling flexible decision-making.

However, in many practical scenarios, this task becomes more challenging when optimization must proceed offline using only a limited, fixed dataset of previous observations without further queries to the true evaluations (Trabucco et al., 2021; Yu et al., 2021; Chen et al., 2023). As such datasets typically cover only a small portion of the decision and objective spaces, they provide limited information about promising designs for query preferences (Xue et al., 2024; Yuan et al., 2025; Zhu et al., 2026). The key challenge is therefore how to use the transferable and integrable information in the limited observations to infer high-quality Pareto optimal solutions for arbitrary unseen preferences.

Existing methods follow two main paradigms for producing preference-conditioned solutions. The distinction is whether the predicted solution $\hat{\boldsymbol{x}}(\boldsymbol{\lambda})$ is induced implicitly or represented explicitly as a solution model. Implicit method such as ParetoFlow learns a flow model of offline decisions and objective predictors, then uses the predictors to guide generation for all queried preferences (Yuan et al., 2025). This paradigm asks how well each preference-conditioned solution can be optimized given a reliable instance-specific model. However, this procedure still requires



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.

test-time search for each query preference, and makes the final solution sensitive to surrogate error when data are limited. By contrast, explicit methods exploit regularity across preferences to directly learn a solution model (Navon et al., 2021), typically a multilayer perceptron (MLP), that approximates the preference-to-solution mapping $\boldsymbol{\lambda} \mapsto \hat{\boldsymbol{x}}(\boldsymbol{\lambda}) \approx \boldsymbol{x}^\star(\boldsymbol{\lambda})$. Representative approaches include regression-based Pareto set learning (PSLR) developed from regression-based amortized optimization (Amos, 2023), and classic objective-based Pareto set learning (PSLO) (Lin et al., 2022) and recently proposed DOMOO (Zhu et al., 2026). Although these methods avoid repeated test-time search across preferences, they typically require training a solution model from scratch for each new problem instance. This motivates a general mechanism for explicitly learning a high-quality solution model from a limited offline static dataset, while avoiding both per-problem retraining and per-preference test-time search.

To this end, we propose *in-context preference-conditioned Pareto set learning (ICPSL)*. The core component is a Transformer-based, foundation-style hypernetwork pretrained on extensive synthetic prior data. Given the offline observations of a new problem instance, the hypernetwork performs in-context inference and directly generates an problem-specific solution model in a single forward pass. The generated MLP solution model explicitly represents the preference-to-solution mapping and returns a solution for any query preference through a single MLP forward pass. The framework therefore realizes two levels of amortization: the generated solution model amortizes solution search across query preferences for a given problem, while the pretrained hypernetwork amortizes solution-model construction across various problem instances. In this view, ICPSL moves beyond per-preference solution search and per-problem model training, shifting toward a foundation-style paradigm in which a pretrained model learns to use offline observations in context to construct solution models for arbitrary query preferences.

Our contributions are summarized as follows:

- We introduce ICPSL, a framework in which a foundation-style hypernetwork is pretrained on extensive synthetic prior data to generate a reusable problem-specific solution model from limited offline observations in one forward pass, thereby amortizing both solution search across preferences and model construction across problems.
- To accommodate potential discontinuities in the optimal preference-to-solution mapping, we model a preference-conditioned solution distribution through latent-conditioned components and develop three inference strategies to predict solutions.
- Compared with representative baselines, the proposed framework produces high-quality Pareto solutions on synthetic problems and real-world engineering applications, while providing substantial speedups in problem-specific solution-model construction.

## 2 Preliminaries and Related Work

### 2.1 Preliminaries

**Problem Definition.** In this paper, we consider the following multi-objective optimization problem (MOP):

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

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T$ is the feature vector in the decision space $\mathcal{X} \subset \mathbb{R}^d$, and $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^m$ is the objective vector containing $m$ continuous objective functions. Typically, there is no single solution that can simultaneously achieve the optimal for all objectives. Instead, the goal is to find a set of solutions with Pareto optimal objective trade-offs. We say an objective vector $\boldsymbol{f}(\boldsymbol{x})$ Pareto dominates another $\boldsymbol{f}(\boldsymbol{x}')$, denoted as $\boldsymbol{f}(\boldsymbol{x}) \prec \boldsymbol{f}(\boldsymbol{x}')$ if $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}'), \forall i \in \{1, \ldots, m\}$ and there exists $j \in \{1, \ldots, m\}$ such that $f_j(\boldsymbol{x}) < f_j(\boldsymbol{x}')$. A solution $\boldsymbol{x}^* \in \mathcal{X}$ is Pareto optimal if there is no $\boldsymbol{x}$ such that $\boldsymbol{f}(\boldsymbol{x}) \prec \boldsymbol{f}(\boldsymbol{x}^*)$. The set of all Pareto optimal solutions is called the Pareto set (PS), and the image of PS in the objective space is called the Pareto front (PF).

**Tchebycheff Scalarization.** Given the preference vector $\boldsymbol{\lambda} \in \Delta^{m-1}$, where $\Delta^{m-1} = \{\boldsymbol{\lambda} \in \mathbb{R}^m_+ : \sum_{j=1}^m \lambda_j = 1\}$, Tchebycheff scalarization measures the largest weighted deviation from the ideal

objective vector $\boldsymbol{f}^{\text{ideal}}$, with $f_j^{\text{ideal}} = \inf_{\boldsymbol{x}\in\mathcal{X}} f_j(\boldsymbol{x})$:

$$s_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \max_{j\in\{1,\dots,m\}} \lambda_j\big(f_j(\boldsymbol{x}) - f_j^{\text{ideal}}\big). \tag{2}$$

With a suitable ideal point, each Pareto optimal solution can be obtained by optimizing the Tchebycheff scalarization with a specific but unknown preference (Choo & Atkins, 1983):

$$\boldsymbol{x}^\star(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{x}\in\mathcal{X}} s_{\boldsymbol{\lambda}}(\boldsymbol{x}). \tag{3}$$

We study the problem when the objectives are unknown and the sole source of information is a fixed offline dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N_c}$, with $\boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i)$. No additional objective evaluation is allowed during training or inference. Our goal is to use $\mathcal{D}$ to infer all preference-conditioned Pareto optimal approximates $\hat{\boldsymbol{x}}(\boldsymbol{\lambda}) \approx \boldsymbol{x}^\star(\boldsymbol{\lambda})$ for arbitrary preferences.

### 2.2 Related Work

**Pareto Set Learning.** Pareto set learning (PSL) amortizes solution search across preferences by learning a preference-to-solution mapping for a given problem. Pareto HyperNetworks use a hypernetwork to generate preference-conditioned target-network weights (Navon et al., 2021), while PSL-MOBO applies preference-conditioned modeling to expensive black-box optimization (Lin et al., 2022). Subsequent work addresses different challenges in learning this mapping. EPS adapts preference sampling to the shape of the Pareto front (Ye et al., 2024), whereas CDM-PSL uses diffusion models to capture complex solution distributions in multi-objective Bayesian optimization (Li et al., 2025). These advances improve preference coverage or solution generation within a target problem, but still rely on problem-specific training rather than constructing a reusable model directly from a fixed offline context.

**Offline Multi-objective Optimization.** Offline model-based optimization seeks improved designs from fixed observations without further objective evaluations. Earlier work addresses challenges of this setting through conservative modeling, robust adaptation, and parallel mentoring (Trabucco et al., 2021; Yu et al., 2021; Chen et al., 2023). The benchmark of Xue et al. (2024) extends this setting to multiple objectives. Within offline MOO, ParetoFlow guides samples from a learned flow using objective predictors (Yuan et al., 2025), whereas DOMOO combines objective surrogates and out-of-distribution risk modeling with nested PSL (Zhu et al., 2026). These methods respectively improve preference-conditioned generation and mitigate out-of-distribution extrapolation, but still rely on problem-specific training or preference-wise search rather than constructing a reusable solution model in one forward pass.

**Amortized Multi-objective Optimization.** Amortized optimization learns to predict solutions across related problem instances instead of solving each instance from scratch (Amos, 2023). Recent work extends this principle to MOO through pretrained foundation-style models that transfer across problem instances. BOFormer uses Transformer and non-Markovian reinforcement learning to learn a multi-objective Bayesian-optimization policy (Hung et al., 2025), while TAMO pretrains a task-agnostic policy that proposes the next design from an optimization history in a single forward pass (Zhang et al., 2026). Both methods amortize candidate generation, but operate in an online setting in which proposed designs receive further objective evaluations. Moreover, they do not provide preference-conditioned solutions and thus cannot explore different trade-off regions by adjusting queried preferences.

## 3 Methodology

### 3.1 Framework Overview

As shown in Figure 1, ICPSL learns to directly construct a problem-specific Pareto solution model from the offline observations $\mathcal{D}$ of a new multi-objective optimization problem. To this end, we pretrain a foundation-style hypernetwork $H_{\boldsymbol{\theta}}$ on a large and diverse collection of synthetic multi-objective problems. At inference, the pretrained hypernetwork remains fixed and maps the offline observations $\mathcal{D}$ from an unseen target problem to the parameters $\boldsymbol{\phi}_{\mathcal{D}} = H_{\boldsymbol{\theta}}(\mathcal{D})$ of a problem-specific Pareto solution model. The resulting model $h_{\boldsymbol{\phi}_{\mathcal{D}}}$ is a lightweight MLP instantiated directly

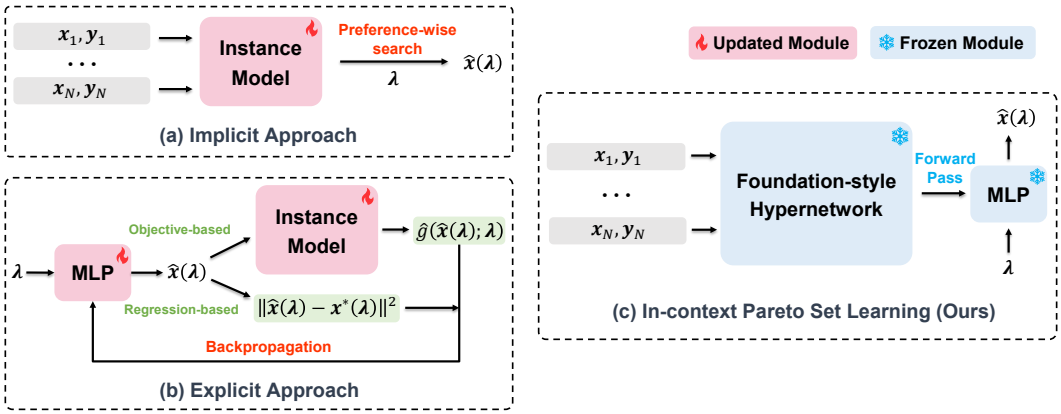



Figure 1: Comparison of offline preference-conditioned Pareto set learning paradigms. (a) The implicit approach performs preference-wise search over a problem-specific instance model. (b) The explicit approach trains a problem-specific MLP as the Pareto solution model through objective- or regression-based learning. (c) ICPSL uses a frozen foundation-style hypernetwork to generate an problem-specific MLP from offline observations in a single forward pass, which maps arbitrary preferences to solutions without problem-specific training or preference-wise search.

from the hypernetwork output. Without separate training or fine-tuning, it serves as a reusable Pareto solution model that maps an arbitrary preference $\boldsymbol{\lambda}$ to the corresponding predicted solution as $\hat{\boldsymbol{x}}_{\mathcal{D}}(\boldsymbol{\lambda}) = h_{\phi_{\mathcal{D}}}(\boldsymbol{\lambda})$.

Consequently, ICPSL requires neither problem-specific training nor test-time optimization over the queried preferences. This efficiency stems from two complementary levels of amortization. The pretrained hypernetwork first amortizes Pareto solution-model construction across problem instances, while the generated MLP subsequently amortizes solution search across preferences within each problem.

### 3.2 Synthetic Prior and Training Data

To generalize across multi-objective problems, the hypernetwork should be exposed to a large corpus of offline observations spanning diverse multi-objective problems during pretraining. However, obtaining observations at this scale from real-world systems is often impractical due to high evaluation costs, potential safety risks, privacy or proprietary restrictions. We therefore adopt prior-data pretraining (Nguyen et al., 2023; Hollmann et al., 2025; Zhang et al., 2026), synthesizing a large and diverse collection of training episodes from a designed generative prior that varies the decision and objective dimensionalities, context size, and objective-landscape geometry, thereby exposing the hypernetwork to a broad range of problem structures that may arise in practice.

#### 3.2.1 Synthetic Prior

We adopt a Gaussian process (GP) prior to synthesize objective functions because it provides a cost-effective and flexible mechanism for generating a large collection of diverse problem landscapes. Moreover, the kernel family and its hyperparameters provide direct control over the distribution of the sampled objective functions. For every objective, we independently draw a latent function from the GP prior,

$$\widetilde{f}_j \sim \mathcal{GP}(0, k_j), \qquad k_j \sim \mathrm{Uniform}(\mathcal{K}), \qquad j = 1, \ldots, m, \tag{4}$$

where $\mathcal{K}$ contains the RBF, Matérn-$3/2$, and Matérn-$5/2$ kernels. The RBF kernel is a universal approximator (Micchelli et al., 2006) to allow the prior to encompass a broad range of unknown objective functions, whereas the Matérn kernels complement it with rougher sample paths and different degrees of regularity. In addition, the automatic relevance determination (ARD) length scales are sampled independently to induce variation in characteristic scales and anisotropy. Together, these choices enable the synthetic prior to generate objectives with diverse landscape properties.

### 3.2.2 TRAINING DATA GENERATION

At each pretraining step, we construct a synthetic training episode in three steps. **(i) Prior configuration.** We sample the decision dimensionality $d$, the number of objectives $m$, the context size $N_c$, as well as the kernel choice and ARD length scales associated with each objective. **(ii) Function realization.** Given this configuration, we instantiate a multi-objective function $\boldsymbol{f} = (f_1, \ldots, f_m)$ from the GP prior. Each objective is represented as a reusable approximate GP path using random Fourier features (RFFs), allowing the same function realization to be evaluated consistently at the predicted solutions when computing the pretraining loss. **(iii) Context and query construction.** We sample decisions $\{\boldsymbol{x}_i\}_{i=1}^{N_c}$ uniformly from the decision space $\mathcal{X} = [0,1]^d$ and collect their observations to form the offline context $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N_c}$. We further sample a set of query preferences $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_q\}_{q=1}^{Q}$ from the simplex $\Delta^{m-1} = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m \mid \|\boldsymbol{\lambda}\|_1 = 1\}$ to evaluate the generated Pareto solution model during pretraining. Collectively, $\boldsymbol{f}$, $\mathcal{D}$, and $\boldsymbol{\Lambda}$ define a complete synthetic training episode.

## 3.3 MODEL ARCHITECTURE

The hypernetwork $H_{\boldsymbol{\theta}}$ transforms a variable-sized offline context $\mathcal{D}$ into a problem-specific Pareto solution model through three modules: a fixed-width context embedding, a permutation-equivariant Transformer encoder, and a low-rank parameter decoder. Together, these modules encode the offline observations into a problem-level representation and produce instance-specific parameters, which instantiate a lightweight MLP to generate a preference-conditioned solution distribution.

### 3.3.1 CONTEXT EMBEDDING

Given the offline context $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N_c}$, directly embedding each objective observation $\boldsymbol{y}_i$ would not explicitly characterize its quality under different trade-offs. We therefore construct a fixed-width representation using a bank of reference preferences $\{\overline{\boldsymbol{\lambda}}_j\}_{j=1}^{J} \subseteq \Delta^{m-1}$ that covers both the vertices and the interior of the preference simplex. For each observation $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, its Tchebycheff scalarization signature is $\boldsymbol{g}_i = (g_{i1}, \ldots, g_{iJ})$, where $g_{ij} = \max_{\ell \in \{1,\ldots,m\}} \overline{\lambda}_{j\ell} y_{i\ell}$. The resulting signature maps observations of different objective dimensionalities into a common space while exposing their quality across a shared collection of trade-offs. To accommodate different decision dimensionalities, each decision is zero-padded to $d_{\max}$ dimensions. Separate linear encoders then project the padded decision and its scalarization signature into a shared latent space, and their sum forms the corresponding context token:

$$\boldsymbol{e}_i = E_x(\mathrm{pad}(\boldsymbol{x}_i)) + E_g(\boldsymbol{g}_i)\,. \tag{5}$$

This construction produces one token for each offline observation rather than expanding the context over all observation–preference pairs. Consequently, the number of tokens remains $|\mathcal{D}|$, while each token summarizes the corresponding observation across the reference trade-offs.

### 3.3.2 TRANSFORMER ENCODER

After embedding, the context tokens are jointly processed by a Transformer encoder $T_{\boldsymbol{\tau}}$ composed of multi-head self-attention, position-wise feed-forward transformations and layer normalization. Self-attention captures dependencies among the offline observations, allowing each token to be interpreted in relation to the full context. Since $\mathcal{D}$ has no canonical ordering, no positional encoding is introduced. The encoded tokens are aggregated by average pooling to obtain a problem-level representation:

$$\{\boldsymbol{z}_i\}_{i=1}^{N_c} = T_{\boldsymbol{\tau}}\Big(\{\boldsymbol{e}_i\}_{i=1}^{N_c}\Big), \qquad \overline{\boldsymbol{z}}_{\mathcal{D}} = \frac{1}{N_c}\sum_{i=1}^{N_c} \boldsymbol{z}_i. \tag{6}$$

Together, position-free encoding and symmetric pooling make $\overline{\boldsymbol{z}}_{\mathcal{D}}$ invariant to the ordering of the offline observations.

#### 3.3.3 LOW-RANK PARAMETER DECODER

Given the pooled problem representation $\overline{\boldsymbol{z}}_{\mathcal{D}}$, a feed-forward parameter decoder $D_{\boldsymbol{\eta}}$ produces the instance-dependent parameters $\boldsymbol{\phi}_{\mathcal{D}}$ used to instantiate a problem-specific two-hidden-layer MLP:

$$\begin{aligned}\boldsymbol{\phi}_{\mathcal{D}} &= D_{\boldsymbol{\eta}}(\overline{\boldsymbol{z}}_{\mathcal{D}}) \\ &= \big\{A_1(\mathcal{D}), A_2(\mathcal{D}), W_o(\mathcal{D}), \boldsymbol{b}_1(\mathcal{D}), \boldsymbol{b}_2(\mathcal{D}), \boldsymbol{b}_o(\mathcal{D})\big\}.\end{aligned} \tag{7}$$

Generating the hidden-layer weight matrices densely would make the decoder output unnecessarily large. Instead, the decoder predicts the problem-dependent low-rank factors $A_1(\mathcal{D})$ and $A_2(\mathcal{D})$, which are combined with the globally learned projections $S_1$ and $S_2$ to construct the hidden-layer weight matrices $W_1(\mathcal{D})$ and $W_2(\mathcal{D})$:

$$W_1(\mathcal{D}) = A_1(\mathcal{D})S_1, \qquad W_2(\mathcal{D}) = A_2(\mathcal{D})S_2. \tag{8}$$

The factors $A_1(\mathcal{D})$ and $A_2(\mathcal{D})$, the full output matrix $W_o(\mathcal{D})$, and all biases are generated directly from the current context. In contrast, $S_1$ and $S_2$ are global parameters learned during pretraining and shared across problem instances.

#### 3.3.4 SOLUTION MODEL INSTANTIATION

To accommodate potential discontinuities in the optimal preference-to-solution mapping $\boldsymbol{\lambda} \mapsto \boldsymbol{x}^\star(\boldsymbol{\lambda})$, ICPSL models a preference-conditioned solution distribution $q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda})$. We represent this distribution using a family of latent-conditioned component $q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}, z)$ indexed by $z \sim U([0,1])$, allowing probability mass to be assigned to multiple high-quality solution regions associated with the same preference. To parameterize each component, the $k$-th realization $z_k$ of the latent variable $z$ is concatenated with a preference zero-padded to $m_{\max}$ dimensions, yielding $\boldsymbol{v}_k(\boldsymbol{\lambda}) = [\mathrm{pad}(\boldsymbol{\lambda}); z_k]$. The decoded parameters $\boldsymbol{\phi}_{\mathcal{D}}$ instantiate a problem-specific MLP $h_{\boldsymbol{\phi}_{\mathcal{D}}}$ that processes this input as

$$\boldsymbol{u}_{1k} = \rho(\boldsymbol{v}_k(\boldsymbol{\lambda})W_1(\mathcal{D}) + \boldsymbol{b}_1(\mathcal{D}))\,, \tag{9}$$

$$\boldsymbol{u}_{2k} = \rho(\boldsymbol{u}_{1k}W_2(\mathcal{D}) + \boldsymbol{b}_2(\mathcal{D}))\,, \tag{10}$$

$$[\boldsymbol{\mu}_k, \boldsymbol{\ell}_k] = \boldsymbol{u}_{2k}W_o(\mathcal{D}) + \boldsymbol{b}_o(\mathcal{D}), \tag{11}$$

where $\rho$ denotes ReLU. The outputs $\boldsymbol{\mu}_k$ and $\boldsymbol{\ell}_k$ parameterize the Gaussian mean and lower-triangular scale matrix, respectively. After converting $\boldsymbol{\ell}_k$ into a valid Cholesky factor $L_k$, these outputs define the latent-conditioned component

$$q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}, z_k) = \mathrm{LogitNormal}\big(\boldsymbol{x}; \boldsymbol{\mu}_k, L_k L_k^\top\big)\,. \tag{12}$$

Finally, the preference-conditioned solution distribution is obtained by marginalizing these components over $z$. In practice, we approximate this integral using $K$ randomized Sobol points $\{z_k\}_{k=1}^K$:

$$q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}) = \int q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}, z)\, p(z)\, \mathrm{d}z \approx \frac{1}{K}\sum_{k=1}^{K} q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}, z_k)\,. \tag{13}$$

The resulting distribution can allocate probability mass to multiple disconnected low-energy regions, thereby accommodating discontinuities in the optimal preference-to-solution mapping that may be difficult for a deterministic continuous solution model to represent. Appendix B further details the construction of the latent-conditioned components and their marginalization.

### 3.4 OBJECTIVE-BASED PRETRAINING

ICPSL is pretrained directly using evaluations of the sampled synthetic objective functions, without requiring precomputed Pareto-optimal solutions as supervision. For a query preference $\boldsymbol{\lambda}$, we define the Tchebycheff energy in the normalized objective space

$$s_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \max_{j \in \{1,\dots,m\}} \lambda_j f_j(\boldsymbol{x}). \tag{14}$$

Ideally, the preference-conditioned solution distribution should concentrate its probability mass on high-quality solutions under the queried preference. We therefore define the target preference-conditioned solution distribution as the Gibbs distribution induced by this energy,

$$p_\tau^\star(\boldsymbol{x} \mid \boldsymbol{\lambda}, \boldsymbol{f}) = \frac{\exp(-s_{\boldsymbol{\lambda}}(\boldsymbol{x})/\tau)}{Z(\boldsymbol{\lambda}, \boldsymbol{f})}, \qquad Z(\boldsymbol{\lambda}, \boldsymbol{f}) = \int_{\mathcal{X}} \exp(-s_{\boldsymbol{\lambda}}(\boldsymbol{u})/\tau)\, \mathrm{d}\boldsymbol{u}. \tag{15}$$

The temperature $\tau > 0$ controls its concentration, with smaller values placing more probability mass around low-energy solutions. Given the learned preference-conditioned solution distribution $q_{\phi_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda})$, ICPSL minimizes the free-energy loss

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{x}\sim q_{\phi_{\mathcal{D}}}(\boldsymbol{x}|\boldsymbol{\lambda})}[s_{\boldsymbol{\lambda}}(\boldsymbol{x})] + \tau\, \mathbb{E}_{\boldsymbol{x}\sim q_{\phi_{\mathcal{D}}}(\boldsymbol{x}|\boldsymbol{\lambda})}[\log q_{\phi_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda})]\,. \tag{16}$$

The first expectation shifts probability mass toward solutions with low Tchebycheff energy, whereas the second is the temperature-weighted negative differential entropy of the preference-conditioned solution distribution and discourages premature concentration. As derived in Appendix C.1, minimizing $\mathcal{L}$ is equivalent to minimizing the KL divergence between the learned preference-conditioned solution distribution and the Tchebycheff-energy-induced target distribution.

During pretraining, we estimate both expectations using reparameterized samples, evaluate their Tchebycheff energies through the synthetic objective paths, and compute the marginal log density by aggregating each sample's density across all latent-conditioned components. This objective requires neither Pareto-optimal solution labels nor a separate optimizer for the sampled training problems. Appendix C.2 provides the complete Monte Carlo estimator.

### 3.5 In-Context Inference

Once hypernetwork pretraining is complete, inference for a target problem operates directly on the generated problem-specific solution model $h_{\phi_{\mathcal{D}}}$. For each queried preference, the model defines a preference-conditioned solution distribution from which a single predicted solution is produced for decision making. We consider three inference strategies: maximum a posteriori (ICPSL-MAP), component-averaged mean (ICPSL-CAM), and mixture mean (ICPSL-MM).

**ICPSL-MAP.** This strategy draws one reparameterized candidate $\boldsymbol{x}_k$ from every latent-conditioned component and evaluates each candidate under the full preference-conditioned solution density. It then selects

$$\hat{\boldsymbol{x}}_{\mathrm{MAP}}(\boldsymbol{\lambda}) = \operatorname*{arg\,max}_{\boldsymbol{x}\in\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_K\}} q_{\phi_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda})\,. \tag{17}$$

This sample-based approximate MAP rule favors a locally high-density solution while accounting for overlap among all mixture components.

**ICPSL-CAM.** This strategy averages the sigmoid-transformed component means. Specifically, it applies the sigmoid directly to each pre-sigmoid mean and computes

$$\hat{\boldsymbol{x}}_{\mathrm{CAM}}(\boldsymbol{\lambda}) = \frac{1}{K}\sum_{k=1}^{K} \sigma(\boldsymbol{\mu}_k(\boldsymbol{\lambda}))\,. \tag{18}$$

This deterministic rule avoids Gaussian sampling and density evaluation, making it the least expensive way to aggregate all mixture components into one prediction.

**ICPSL-MM.** This strategy estimates the mean of the complete preference-conditioned solution distribution. It draws $R$ Gaussian samples from each latent-conditioned component and averages the corresponding bounded decisions:

$$\hat{\boldsymbol{x}}_{\mathrm{MM}}(\boldsymbol{\lambda}) = \frac{1}{KR}\sum_{k=1}^{K}\sum_{r=1}^{R} \sigma(\boldsymbol{\mu}_k(\boldsymbol{\lambda}) + L_k(\boldsymbol{\lambda})\boldsymbol{\epsilon}_{kr})\,, \qquad \boldsymbol{\epsilon}_{kr} \sim \mathcal{N}(\mathbf{0}, I). \tag{19}$$

Unlike ICPSL-CAM, this Monte Carlo estimate incorporates each component's covariance after the nonlinear sigmoid transformation. All three inference strategies operate solely on the generated solution model and require no parameter updates, preference-wise optimization, or additional evaluations of the true objectives.

## 4 Experiments

### 4.1 Experimental Setting

**Baselines.** We compare ICPSL with representative baselines, including the implicit method ParetoFlow (Yuan et al., 2025), the regression-based explicit method PSLR (Amos, 2023), and the

objective-based explicit methods PSLO (Lin et al., 2022) and DOMOO (Zhu et al., 2026). Detailed baseline configurations are provided in Appendix D.1.

**Benchmarks.** We evaluate all methods on a diverse suite of benchmarks covering synthetic functions (Blank & Deb, 2020) and real-world engineering design problems (Tanabe & Ishibuchi, 2020). For each problem, we construct an offline context of 10 observations using Latin hypercube sampling (LHS), scrambled Sobol sampling, or uniform random sampling. We conduct 10 independent runs for each sampling method, giving all algorithms the same context within each run. Further benchmark details are provided in Appendix D.3.

**Evaluation Protocol.** We evaluate the true objectives after inference and measure solution quality by hypervolume (HV) in the normalized objective space with reference point $\boldsymbol{r} = [1.1, ..., 1.1]^T \in \mathbb{R}^m$, where higher values are better. Over 10 independent runs, we report the mean HV and the half-width of its $95\%$ confidence interval, together with wall-clock times for problem-specific model construction and solution generation across all query preferences. Further details are provided in Appendix D.4.

**Implementation.** For ParetoFlow, PSLO, and DOMOO, we use the authors' official implementations. ICPSL and PSLR are implemented in PyTorch. Baseline code sources and further implementation details are provided in Appendix D.

## 4.2 Main Results

### 4.2.1 Optimization Performance

We evaluate all methods under offline contexts generated by LHS, Sobol, and uniform sampling, with lower average ranks indicating better optimization performance. As shown in Table 1, ICPSL-MM achieves the best overall average rank and the most wins, ranking first in 15 of the 30 cases.

Table 1: Average ranks, wins, and top-3 counts across all benchmark instances with LHS, Sobol, and uniform context distributions.

| Algorithm | LHS | Sobol | Uniform | Overall | Wins | Top-3 cases |
|---|---|---|---|---|---|---|
| ParetoFlow | 4.50 | 4.50 | 4.20 | 4.40 | 5/30 | 9/30 |
| PSLR | 3.70 | 3.40 | 3.80 | 3.63 | 2/30 | 18/30 |
| PSLO | 5.10 | 5.20 | 5.80 | 5.37 | 2/30 | 5/30 |
| DOMOO | 4.30 | **3.20** | 5.10 | 4.20 | 5/30 | 10/30 |
| ICPSL-MAP | 5.00 | 4.80 | 4.20 | 4.67 | 1/30 | 7/30 |
| ICPSL-CAM | **2.70** | 3.70 | 2.60 | 3.00 | 0/30 | **23/30** |
| ICPSL-MM | **2.70** | **3.20** | **2.30** | **2.73** | **15/30** | 18/30 |

For each sampling method, ICPSL-MM obtains the best average rank under the uniform context distribution and shares the best rank under both LHS and Sobol, demonstrating consistent performance across different offline observation patterns. ICPSL-CAM is also competitive, sharing the best LHS rank and achieving the highest top-3 count of 23/30. These results indicate that amortizing Pareto solution-model construction across problems does not compromise optimization quality. Complete performance results are provided in Appendix E.

### 4.2.2 Computational Efficiency

Beyond solution quality, we record the wall-clock times required for problem-specific Pareto solution-model construction and preference-conditioned solution generation. Figure 2 reports baseline-to-ICPSL runtime ratios, where values above one indicate that ICPSL is faster. For model construction, ICPSL provides orders-of-magnitude speedups over all four baselines across all 30 cases because a single hypernetwork forward pass replaces problem-specific training. At inference, ICPSL-CAM offers the best trade-off among the three variants, it achieves a averaged speedup of $1.3 \times 10^3$ over ParetoFlow in 30/30 cases and $1.8\times$ over PSLR in 26/30 cases. ParetoFlow has the longest inference time because it has to perform solution search for all queried preferences. Complete timing results are provided in Appendix E.

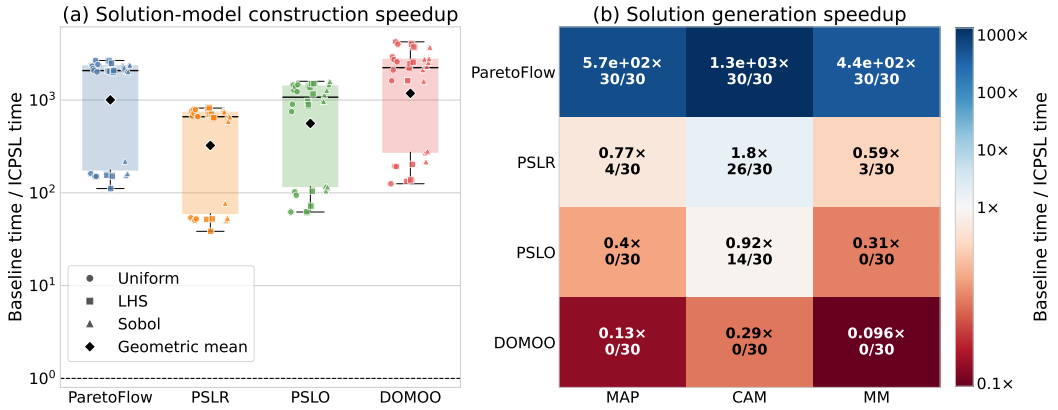



Figure 2: Runtime speedups of ICPSL over the baselines for problem-specific Pareto solution-model construction and preference-conditioned solution generation.

### 4.3 Additional Analysis

**Evolution of Solution-Model Quality.** We track solution-model quality across checkpoints saved every 30 epochs on three unseen problems during pretraining (Figure 3 in Appendix F.1). The results show that the solution quality generally increases with pretraining, although the trajectories are not monotonic. On VLMOP2 and ABC, ICPSL-CAM and ICPSL-MM consistently outperform ICPSL-MAP, and their improvement is particularly pronounced on VLMOP2. This pattern suggests that aggregating across latent-conditioned components helps convert the learned distribution into a more effective point prediction. The less stable RE23 trajectories under uniform contexts indicate that these gains are not uniform across offline observation patterns.

**Sensitivity to Context Size.** We examine how solution quality varies with different context sizes (Figure 4 in Appendix F.2). The results indicate that solution quality generally improves as the context grows, but the gains vary across benchmarks and inference strategies, with some trajectories rising gradually and others fluctuating or leveling off. This suggests that additional observations can provide useful problem-specific information, although their benefits are not uniformly concentrated at small context sizes. The smallest evaluation contexts also contain fewer observations than those used during pretraining, probing adaptation to greater data scarcity. Continued gains with larger contexts indicate that additional observations can still refine the generated model.

**Generalization to OOD Dimensions.** We evaluate generalization beyond the dimensionalities sampled during pretraining. As reported in Appendix F.3, ICPSL-MM achieves the highest hypervolume on all three five-dimensional problems under every context distribution, whereas ParetoFlow is strongest on DTLZ4. This contrast suggests that transferring learned trade-off structure to an unseen number of objectives is more challenging in these tests than extending to an additional decision variable, even though the architecture accommodates both dimensions. ICPSL nonetheless retains the fastest model construction across all OOD settings, achieving a substantial computational advantage while delivering leading or competitive solution quality.

## 5 Conclusion

We introduced ICPSL, which pretrains a foundation-style hypernetwork on a large and diverse set of synthetic prior data to generate a reusable and problem-specific solution model from limited offline context in one forward pass, amortizing both model construction across problems and solution search across preferences. Across synthetic and engineering benchmarks, ICPSL achieves high-quality Pareto solutions while providing substantial speedups in solution-model construction.

Our current evaluation primarily covers low-dimensional problems, leaving broader dimensional settings for future study. Marginal-density evaluation also becomes more costly as the number of latent components grows. Future work could diversify synthetic pretraining problems and develop more efficient marginal-density estimators to extend ICPSL to larger-scale settings.

### AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for helping implementing baseline methods and interpreting experimental results. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, designing or providing feedback on research methodology or experiments, and proposing or refine hypotheses, assisting with translation, cleaning and reformating dataset, supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures, summarizing or analyzing existing literature, drafting parts of the paper, improving its readability, and suggesting its structure. We manually reviewed all AI-assisted work, checked baseline implementations against experimental outputs, and verified figures, literature summaries, and result interpretations against their underlying materials. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

(This section is **recommended** and does not count toward the page limit.)

To the best of our knowledge, this work raises no concerns under the ICLR Code of Ethics.

### REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

Appendix D documents the implementation details used in our experiments. Table 2 identifies the baseline code sources, Appendix D.3 specifies the benchmark configurations and implementations, and Appendix D.4 describes the evaluation protocol.

## REFERENCES

Brandon Amos. Tutorial on amortized optimization. *Foundations and Trends in Machine Learning*, 16(5):592–732, 2023. doi: 10.1561/2200000102.

Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European journal of operational research*, 181(3):1653–1669, 2007.

Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *Ieee access*, 8: 89497–89509, 2020.

Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. SMASH: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rydeCEhs-.

Can Sam Chen, Christopher Beckham, Zixuan Liu, Xue Steve Liu, and Chris Pal. Parallel-mentoring for offline model-based optimization. *Advances in Neural Information Processing Systems*, 36: 76619–76636, 2023.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'Aurelio Ranzato, Sagi Perel, and Nando de Freitas. Towards learning universal hyperparameter optimizers with transformers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32053–32068, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/cf6501108fced72ee5c47e2151c4e153-Abstract-Conference.html.

Eng Ung Choo and Derek R Atkins. Proper efficiency in nonconvex multicriteria programming. *Mathematics of Operations Research*, 8(3):467–470, 1983. doi: 10.1287/moor.8.3.467.

Suresh Dara, Swetha Dhamercherla, Surender Singh Jadav, CH Madhu Babu, and Mohamed Jawed Ahsan. Machine learning in drug discovery: a review. *Artificial intelligence review*, 55(3):1947–1999, 2022.

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

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V. Naidu, and Colin White. ForecastPFN: Synthetically-trained zero-shot forecasting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 2403–2426, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/0731f0e65559059eb9cd9d6f44ce2dd8-Abstract-Conference.html.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1704–1713. PMLR, 2018. URL https://proceedings.mlr.press/v80/garnelo18a.html.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637:319–326, 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

Yu-Heng Hung, Kai-Jie Lin, Yu-Heng Lin, Chien-Yi Wang, Cheng Sun, and Ping-Chun Hsieh. BOFormer: Learning to solve multi-objective bayesian optimization via non-markovian reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=UnCKU8pZVe.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 565–576. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.47. URL https://aclanthology.org/2021.acl-long.47/.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 27063–27071, 2025. doi: 10.1609/aaai.v39i25.34913.

Xi Lin, Zhiyuan Yang, Xiaoyuan Zhang, and Qingfu Zhang. Pareto set learning for expensive multiobjective optimization. In *Advances in Neural Information Processing Systems*, volume 35, pp. 19231–19247, 2022. doi: 10.52202/068431-1398.

Alexandre Maraval, Matthieu Zimmer, Antoine Grosnit, and Haitham Bou Ammar. End-to-end meta-bayesian optimisation with transformer neural processes. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11246–11260, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/2561721d0ca69bab22b749cfc4f48f6c-Abstract-Conference.html.

Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(12), 2006.

Andreas C. Müller, Carlo Curino, and Raghu Ramakrishnan. MotherNet: Fast training and inference via hyper-network transformers. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/bed8e95c6a99df485a6fc8b973e2d6cb-Abstract-Conference.html.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25444–25470. PMLR, 2023. URL https://proceedings.mlr.press/v202/muller23a.html.

Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=NjF772F4ZZR.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16569–16594. PMLR, 2022. URL https://proceedings.mlr.press/v162/nguyen22b.html.

Tung Nguyen, Sudhanshu Agrawal, and Aditya Grover. ExPT: Synthetic pretraining for few-shot experimental design. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45856–45869, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8fab4407e1fe9006b39180525c0d323c-Abstract-Conference.html.

Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen. OmniPred: Language models as universal regressors. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=t9c3pfrR1X.

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

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pp. 10358–10368. PMLR, 2021.

Ke Xue, Rongxi Tan, Xiaobin Huang, and Chao Qian. Offline multi-objective optimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 55595–55624. PMLR, 2024. URL https://proceedings.mlr.press/v235/xue24b.html.

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. doi: 10.1145/3638529.3654024.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International conference on machine learning*, pp. 7105–7114. PMLR, 2019.

Sihyun Yu, Sungsoo Ahn, Le Song, and Jinwoo Shin. Roma: Robust model adaptation for offline model-based optimization. *Advances in Neural Information Processing Systems*, 34:4619–4631, 2021.

Ye Yuan, Can Chen, Christopher Pal, and Xue Liu. ParetoFlow: Guided flows in multi-objective optimization. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mLyyB4le5u.

Xinyu Zhang, Conor Hassan, Julien Martinelli, Daolang Huang, and Samuel Kaski. In-context multi-objective optimization. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=odmeUlWta8.

Andrey Zhmoginov, Mark Sandler, and Maksym Vladymyrov. HyperTransformer: Model generation for supervised and semi-supervised few-shot learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27075–27098. PMLR, 2022. URL https://proceedings.mlr.press/v162/zhmoginov22a.html.

Yiyi Zhu, Yaolin Wen, Xiang Xia, Xin An, Hanyi Si, Xiang Shu, Yangde Fu, Liang Dou, and Hong Qian. Diversity-driven offline multi-objective optimization via nested pareto set learning. *arXiv preprint arXiv:2606.15115*, 2026. URL https://arxiv.org/abs/2606.15115.

# APPENDIX

## A ADDITIONAL RELATED WORK

### A.1 UNIVERSAL REGRESSION MODELS

Universal regression models seek to transfer function inference across tasks rather than fit a new predictor from scratch for every function. Conditional Neural Processes (CNPs) encode an unordered set of input–output observations into a representation used to predict target values (Garnelo et al., 2018), while Transformer Neural Processes (TNPs) formulate uncertainty-aware meta-regression as sequence modeling (Nguyen & Grover, 2022). Prior-Data Fitted Networks (PFNs) instead pretrain a Transformer on datasets sampled from a prior to approximate Bayesian posterior prediction in context (Müller et al., 2022). TabPFN develops this paradigm into a foundation model for small tabular datasets (Hollmann et al., 2025). Subsequent work extends broadly pretrained regression to zero-shot forecasting with ForecastPFN (Dooley et al., 2023) and to heterogeneous numerical tasks represented as text with OmniPred (Song et al., 2024). These models predict responses or predictive distributions, so their use in optimization generally requires a subsequent search procedure. ICPSL instead constructs a preference-conditioned solution model directly from offline observations.

### A.2 HYPERNETWORKS

Hypernetworks generate the parameters of another neural network, providing a mechanism for conditioning an entire model on auxiliary information (Ha et al., 2017). SMASH conditions a hypernetwork on candidate architectures to generate their weights, thereby amortizing the cost of evaluating architectures during neural architecture search (Brock et al., 2018). HyperFormer instead uses shared hypernetworks conditioned on the task, Transformer layer, and adapter position to generate adapter parameters for parameter-efficient multi-task adaptation (Karimi Mahabadi et al., 2021). Beyond conditioning on architecture or task identifiers, hypernetworks can also generate model parameters directly from observed examples. HyperTransformer produces few-shot convolutional-network weights from support examples (Zhmoginov et al., 2022), while MotherNet combines prior-data pretraining with a Transformer hypernetwork to generate a compact classifier for a new tabular dataset in a single forward pass without dataset-specific gradient descent (Müller et al., 2025). Unlike prior hypernetwork approaches, ICPSL targets offline multi-objective optimization by generating a problem-specific solution model from offline observations to predict solutions for queried preferences. ICPSL further differs in its objective-based pretraining, which evaluates sampled solutions on synthetic objectives without requiring Pareto-optimal solution labels.

### A.3 FOUNDATION-STYLE OPTIMIZATION

Several methods transfer optimization experience across problem instances through pretrained sequence models. OptFormer learns function prediction and optimization-policy behavior from heterogeneous hyperparameter-optimization trajectories (Chen et al., 2022). PFNs4BO performs in-context Bayesian inference from a synthetic prior for Bayesian optimization (Müller et al., 2023). ExPT pretrains a few-shot experimental-design policy on synthetic functions (Nguyen et al., 2023). End-to-end meta-BO jointly learns a Transformer Neural Process surrogate and acquisition mechanism through reinforcement learning (Maraval et al., 2023). This direction has also been extended to multiple objectives. BOFormer uses non-Markovian deep Q-learning and Transformer sequence modeling to learn a non-myopic acquisition policy for multi-objective Bayesian optimization (Hung et al., 2025). TAMO pretrains a fully amortized policy that proposes the next design from the query history in a single forward pass across varying input and objective dimensions (Zhang et al., 2026). These approaches primarily target sequential candidate selection or experimental design. In contrast, ICPSL addresses offline multi-objective optimization without additional objective evaluations and directly constructs a reusable solution model for arbitrary queried preferences.

# B Details of Preference-Conditioned Solution Distribution Modeling

This section provides further details on preference-conditioned solution distribution modeling in ICPSL. We first describe how the problem-specific MLP parameterizes the latent-conditioned components. We then explain how component marginalization yields the preference-conditioned solution distribution and enables it to represent multiple solution modes. Finally, we discuss the main challenges associated with marginal-density evaluation and latent collapse.

## B.1 Latent-Conditioned Component Modeling

Given the offline context $\mathcal{D}$ from the target problem, the hypernetwork first produces the problem-dependent parameters $\boldsymbol{\phi}_{\mathcal{D}} = H_{\boldsymbol{\theta}}(\mathcal{D})$, which instantiates the problem-specific MLP $h_{\boldsymbol{\phi}_{\mathcal{D}}}$. For a preference $\boldsymbol{\lambda}$ and a scalar latent variable $z$, this MLP outputs

$$h_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{\lambda}, z) = \Big(\boldsymbol{\mu}(\boldsymbol{\lambda}, z), \widetilde{L}(\boldsymbol{\lambda}, z)\Big), \tag{20}$$

where $\boldsymbol{\mu}(\boldsymbol{\lambda}, z) \in \mathbb{R}^d$ is a Gaussian mean and $\widetilde{L}(\boldsymbol{\lambda}, z) \in \mathbb{R}^{d\times d}$ is a raw lower-triangular scale matrix. We obtain the numerically stable Cholesky factor $L(\boldsymbol{\lambda}, z)$ by retaining the strictly lower-triangular entries of $\widetilde{L}(\boldsymbol{\lambda}, z)$ and parameterizing its diagonal as

$$L_{rr}(\boldsymbol{\lambda}, z) = \mathrm{softplus}\Big(\widetilde{L}_{rr}(\boldsymbol{\lambda}, z)\Big) + \delta, \qquad r = 1, \ldots, d, \tag{21}$$

where $\delta > 0$ is a small constant to ensure that $\Sigma(\boldsymbol{\lambda}, z) = L(\boldsymbol{\lambda}, z)L(\boldsymbol{\lambda}, z)^\top$ is a valid and numerically stable covariance matrix. A reparameterized sample is generated as

$$\boldsymbol{a} = \boldsymbol{\mu}(\boldsymbol{\lambda}, z) + L(\boldsymbol{\lambda}, z)\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I), \qquad \boldsymbol{x} = \sigma(\boldsymbol{a}), \tag{22}$$

where the element-wise sigmoid maps the unconstrained Gaussian sample to the normalized decision domain $(0, 1)^d$, inducing the latent-conditioned logit-normal component

$$q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}, z) = \mathrm{LogitNormal}(\boldsymbol{x}; \boldsymbol{\mu}(\boldsymbol{\lambda}, z), \Sigma(\boldsymbol{\lambda}, z)). \tag{23}$$

This distribution forms a latent-conditioned component of the preference-conditioned solution distribution. Its location and local geometry depend jointly on the queried preference and latent variable.

## B.2 Component Marginalization and Its Role

ICPSL places a uniform prior over the latent variable $z \sim U([0, 1])$, and defines its preference-conditioned solution distribution by marginalization:

$$q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}) = \int_0^1 q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}, z)\, \mathrm{d}z. \tag{24}$$

This preference-conditioned solution distribution can be viewed as a continuous mixture of logistic-normal components. Different regions of the latent domain may correspond to distinct solution modes, and the probability mass assigned to each mode is determined by the proportion of latent values mapped to that mode. In practice, we approximate the integral using a randomized one-dimensional Sobol design $\{z_k\}_{k=1}^K$:

$$q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}) \approx \frac{1}{K}\sum_{k=1}^{K} q_{\boldsymbol{\phi}_{\mathcal{D}}}(\boldsymbol{x} \mid \boldsymbol{\lambda}, z_k). \tag{25}$$

For notational simplicity, we use $q$ to denote this finite-mixture approximation hereafter. The same latent design is shared across the query preferences of a problem, so each $z_k$ retains a consistent component identity as $\boldsymbol{\lambda}$ varies.

The latent variable is essential when nearby preferences correspond to well-separated low-energy solution regions. A deterministic continuous model that maps each preference directly to a single solution may interpolate between these regions, potentially yielding decisions with high Tchebycheff values. By contrast, distinct subsets of the latent domain can allocate probability mass to different regions and smoothly transfer that mass as the preference changes. ICPSL therefore models a preference-conditioned solution distribution over low-Tchebycheff solutions rather than forcing a continuous network to reproduce a single-valued optimal solution mapping $\boldsymbol{\lambda} \mapsto \boldsymbol{x}^\star(\boldsymbol{\lambda})$ that may be discontinuous.

### B.3 Marginalization Challenges

The main technical challenge lies in evaluating the density of the preference-conditioned solution distribution, as the continuous marginalization in Equation 24 generally has no closed-form solution. ICPSL addresses this challenge using the finite mixture in Equation 25, which approximates the continuous marginal with an explicit average over $K$ latent-conditioned components. The marginal density of each generated sample is therefore computed by evaluating it under all $K$ components and aggregating the resulting densities. With one sample drawn from each component, this procedure requires $K^2$ pairwise density evaluations for every problem–preference pair, which are computed in chunks to control memory usage. The complete Monte Carlo computation is detailed in Section C.2.

The second challenge is latent collapse when a preference admits multiple separated low-energy solution regions. If different values of $z$ produce nearly identical distribution parameters, the resulting components may concentrate on the same region rather than represent distinct modes. However, when the optimal solution mapping is unique and continuous, component overlap is natural because there are no distinct solution modes to represent. The marginal-entropy regularization described in Appendix C mitigates harmful collapse by discouraging premature concentration of the preference-conditioned solution distribution.

## C Details of the Tchebycheff Free-Energy Loss

This section derives the free-energy loss based on Tchebycheff aggregation for training the preference-conditioned solution distribution model and describes its Monte Carlo implementation.

### C.1 Free-Energy Loss and KL Interpretation

For a preference $\boldsymbol{\lambda} \in \Delta^{m-1}$ and a synthetic multi-objective function $\boldsymbol{f}$, ICPSL aggregates the objectives into the hard Tchebycheff energy

$$s_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \max_{j\in\{1,\dots,m\}} \lambda_j f_j(\boldsymbol{x}). \tag{26}$$

Ideally, a preference-conditioned solution distribution should concentrate its probability mass on high-quality solutions under the queried preference. We therefore define the target preference-conditioned solution distribution as the Gibbs distribution induced by this energy:

$$p^\star_\tau(\boldsymbol{x} \mid \boldsymbol{\lambda}, \boldsymbol{f}) = \frac{\exp(-s_{\boldsymbol{\lambda}}(\boldsymbol{x})/\tau)}{Z(\boldsymbol{\lambda}, \boldsymbol{f})}, \qquad Z(\boldsymbol{\lambda}, \boldsymbol{f}) = \int_{\mathcal{X}} \exp(-s_{\boldsymbol{\lambda}}(\boldsymbol{u})/\tau)\,\mathrm{d}\boldsymbol{u}. \tag{27}$$

The temperature $\tau > 0$ controls the concentration of this distribution. As $\tau$ decreases, probability mass concentrates around the global minimizers of the Tchebycheff energy, while larger values preserve broader coverage of near-optimal regions.

Given the learned preference-conditioned solution distribution $q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x} \mid \boldsymbol{\lambda})$, ICPSL minimizes the loss

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{x}\sim q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x}|\boldsymbol{\lambda})}[s_{\boldsymbol{\lambda}}(\boldsymbol{x})] + \tau\, \mathbb{E}_{\boldsymbol{x}\sim q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x}|\boldsymbol{\lambda})}[\log q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x} \mid \boldsymbol{\lambda})]\,. \tag{28}$$

To characterize how this loss aligns the learned preference-conditioned solution distribution with the target distribution, we express their discrepancy using the KL divergence. Starting from its definition and substituting the target density in Equation 27, we obtain

$$\begin{aligned}
&\mathrm{KL}(q_{\boldsymbol{\phi}_\mathcal{D}}(\cdot \mid \boldsymbol{\lambda}) \,\|\, p^\star_\tau(\cdot \mid \boldsymbol{\lambda}, \boldsymbol{f})) \\
&= \mathbb{E}_{\boldsymbol{x}\sim q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x}|\boldsymbol{\lambda})}[\log q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x} \mid \boldsymbol{\lambda}) - \log p^\star_\tau(\boldsymbol{x} \mid \boldsymbol{\lambda}, \boldsymbol{f})] \\
&= \mathbb{E}_{\boldsymbol{x}\sim q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x}|\boldsymbol{\lambda})}\left[\log q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x} \mid \boldsymbol{\lambda}) + \frac{s_{\boldsymbol{\lambda}}(\boldsymbol{x})}{\tau} + \log Z(\boldsymbol{\lambda}, \boldsymbol{f})\right] \\
&= \mathbb{E}_{\boldsymbol{x}\sim q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x}|\boldsymbol{\lambda})}[\log q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x} \mid \boldsymbol{\lambda})] \\
&\quad + \frac{1}{\tau}\mathbb{E}_{\boldsymbol{x}\sim q_{\boldsymbol{\phi}_\mathcal{D}}(\boldsymbol{x}|\boldsymbol{\lambda})}[s_{\boldsymbol{\lambda}}(\boldsymbol{x})] + \log Z(\boldsymbol{\lambda}, \boldsymbol{f}).
\end{aligned} \tag{29}$$

Multiplying Equation 29 by $\tau$ and using Equation 28, we have

$$\begin{aligned}
\tau\,\mathrm{KL}(q_{\boldsymbol{\phi}_\mathcal{D}}(\cdot \mid \boldsymbol{\lambda}) \,\|\, p^\star_\tau(\cdot \mid \boldsymbol{\lambda}, \boldsymbol{f})) &= \mathcal{L} + \tau \log Z(\boldsymbol{\lambda}, \boldsymbol{f}), \\
\mathcal{L} &= \tau\,\mathrm{KL}(q_{\boldsymbol{\phi}_\mathcal{D}}(\cdot \mid \boldsymbol{\lambda}) \,\|\, p^\star_\tau(\cdot \mid \boldsymbol{\lambda}, \boldsymbol{f})) - \tau \log Z(\boldsymbol{\lambda}, \boldsymbol{f}).
\end{aligned} \tag{30}$$

Because $\tau > 0$ and the partition function is independent of the model parameters, Equation 30 shows that minimizing $\mathcal{L}$ is exactly equivalent to minimizing the KL divergence between the learned preference-conditioned solution distribution and the Tchebycheff free-energy target distribution. The first expectation in Equation 28 shifts probability mass toward low-Tchebycheff solutions, while the second is the temperature-weighted negative differential entropy of the preference-conditioned solution distribution and discourages premature concentration. Section C.2 details the corresponding Monte Carlo implementation.

### C.2 Monte Carlo Estimation of the Free-Energy Loss

Consider a minibatch containing problem instances indexed by $b$, query preferences indexed by $q$, latent-conditioned components indexed by $k$, and Gaussian samples within each latent-conditioned component indexed by $r$. We use $j$ to index the latent-conditioned component under which a generated sample is evaluated. For each synthetic problem, ICPSL generates $K$ randomized Sobol latent points $\{z_{bk}\}_{k=1}^{K}$ and shares the same set across its query preferences. The generated MLP produces

$$(\boldsymbol{\mu}_{bqk}, L_{bqk}) = h_{\boldsymbol{\phi}_{\mathcal{D}_b}}(\boldsymbol{\lambda}_{bq}, z_{bk}), \qquad \boldsymbol{x}_{bqkr} = \sigma(\boldsymbol{\mu}_{bqk} + L_{bqk}\boldsymbol{\epsilon}_{bkr}), \tag{31}$$

where $\boldsymbol{\epsilon}_{bkr} \sim \mathcal{N}(\mathbf{0}, I)$ is also shared across the preferences of a problem. The synthetic GP paths are then evaluated for all generated samples to compute their Tchebycheff energies

$$s_{bqkr} = \max_{\ell \in \{1,\dots,m_b\}} \lambda_{bq\ell} f_{b\ell}(\boldsymbol{x}_{bqkr}). \tag{32}$$

The reparameterization in Equation 31 allows the energy gradient to propagate through the synthetic objectives and generated decisions to the hypernetwork parameters.

The critical computation is the marginal density of each sample. Using the finite set of Sobol latent points, the marginal density of each sample is approximated as

$$q_{\boldsymbol{\phi}_{\mathcal{D}_b}}(\boldsymbol{x}_{bqkr} \mid \boldsymbol{\lambda}_{bq}) \approx \frac{1}{K} \sum_{j=1}^{K} q_{\boldsymbol{\phi}_{\mathcal{D}_b}}(\boldsymbol{x}_{bqkr} \mid \boldsymbol{\lambda}_{bq}, z_{bj}), \tag{33}$$

so every $\boldsymbol{x}_{bqkr}$ must be evaluated under all $K$ latent-conditioned components. For numerical stability, the aggregation is performed in the log domain:

$$\log q_{\boldsymbol{\phi}_{\mathcal{D}_b}}(\boldsymbol{x}_{bqkr} \mid \boldsymbol{\lambda}_{bq}) = \operatorname{logsumexp}_j \left[\log q_{\boldsymbol{\phi}_{\mathcal{D}_b}}(\boldsymbol{x}_{bqkr} \mid \boldsymbol{\lambda}_{bq}, z_{bj})\right] - \log K. \tag{34}$$

This all-sample-to-all-component evaluation requires $O(BQK^2R)$ density computations.

Because the Gaussian is defined in the unconstrained variable $\boldsymbol{a}$ while the loss uses the bounded decision $\boldsymbol{x} = \sigma(\boldsymbol{a})$, each latent-conditioned component density includes the sigmoid change of variables:

$$\begin{aligned} \log q_{\boldsymbol{\phi}_{\mathcal{D}_b}}(\boldsymbol{x}_{bqkr} \mid \boldsymbol{\lambda}_{bq}, z_{bj}) = \log \mathcal{N}\big(\operatorname{logit}(\boldsymbol{x}_{bqkr}); \boldsymbol{\mu}_{bqj}, L_{bqj}L_{bqj}^{\top}\big) \\ - \sum_{u=1}^{d_b} \log[x_{bqkr,u}(1 - x_{bqkr,u})]. \end{aligned} \tag{35}$$

The inverse-Jacobian correction, corresponding to the second term in Equation 35, is evaluated from the pre-sigmoid sample as $\sum_{u=1}^{d_b} [\operatorname{softplus}(-a_{bqkr,u}) + \operatorname{softplus}(a_{bqkr,u})]$, which avoids unstable evaluations of $\log 0$ near the boundary.

Finally, combining the sampled Tchebycheff energy in Equation 32 with the marginal log-density approximation in Equation 34 yields the Monte Carlo objective

$$\widehat{\mathcal{L}} = \frac{1}{BQKR} \sum_{b=1}^{B} \sum_{q=1}^{Q} \sum_{k=1}^{K} \sum_{r=1}^{R} \left[s_{bqkr} + \tau \log q_{\boldsymbol{\phi}_{\mathcal{D}_b}}(\boldsymbol{x}_{bqkr} \mid \boldsymbol{\lambda}_{bq})\right]. \tag{36}$$

The final model uses minibatches of $B = 32$ problem instances, with $Q = 128$ query preferences per instance, $K = 128$ latent-conditioned components per preference, and $R = 1$ Gaussian sample per component, while the temperature is set to $\tau = 0.015$.

# D Implementation Details

## D.1 Baseline Details

We summarize the configurations and implementations of each baseline below. All methods receive identical offline contexts for a given benchmark, sampling scheme, and random seed, and use the same objective normalization, query preferences, and evaluation protocol. Method-specific hyperparameters follow the official implementations.

**ParetoFlow.** ParetoFlow fits a flow-matching model to the offline decisions and a separate objective predictor for each objective, and then guides the learned flow with the predictors to generate preference-conditioned solutions (Yuan et al., 2025). The flow-matching network has a hidden dimension of 512 and is trained for 500 epochs with batch size 64 and learning rate $10^{-3}$. Each objective predictor comprises two hidden layers of width 2,048 and is trained for 100 epochs with batch size 32, an initial learning rate of $10^{-3}$, and a multiplicative decay factor of 0.98. During solution generation, ParetoFlow integrates the guided dynamics over 500 steps, using 5 offspring per preference, cosine-based neighborhoods, and a guidance coefficient of 2.

**PSLR.** PSLR selects the offline design with the lowest augmented Tchebycheff value for each preference and uses it as the corresponding amortized regression target (Amos, 2023). It then trains a preference-to-solution MLP by minimizing the mean-squared error between its predictions and the regression targets. The MLP comprises two hidden layers of width 256 followed by a sigmoid output layer. Optimization is performed for 500 updates using Adam with learning rate $10^{-3}$, 10 randomly sampled preferences per update, and a Tchebycheff augmentation coefficient of 0.01.

**PSLO.** PSLO adopts the objective-based loss (Lin et al., 2022). It fits an independent Gaussian process to each normalized objective and trains a preference-to-solution MLP by directly minimizing the augmented Tchebycheff scalarization of the corresponding lower confidence bounds. The confidence coefficient is set to 0.1, and the solution MLP uses the same two-hidden-layer, width-256 architecture and sigmoid output as PSLR. The MLP is optimized for 500 updates using Adam with learning rate $10^{-3}$, 10 preferences per update, and a Tchebycheff augmentation coefficient of 0.01.

**DOMOO.** DOMOO integrates objective surrogates, energy-based modeling of out-of-distribution risk, and nested Pareto set learning technique (Zhu et al., 2026). Each objective surrogate contains two hidden layers of width 2,048 and is trained for 100 epochs with batch size 256, an initial learning rate of $10^{-3}$, and a multiplicative decay factor of 0.98. The energy model comprises two width-512 hidden layers and is trained for 50 epochs with learning rate $10^{-3}$. The two-hidden-layer, width-256 Pareto solution MLP is first pretrained for 200 epochs on the nondominated context designs and subsequently refined for 300 updates using batches of 256 preferences. The solution model is optimized with Adam at a learning rate of $10^{-3}$, with preferences sampled afresh from a Dirichlet distribution during the first 200 refinement updates.

Table 2: Code sources used to implement the baseline methods.

| Method | Implementation source | Code repository |
|---|---|---|
| ParetoFlow | Official ParetoFlow implementation | `https://github.com/mila-iqia/ParetoFlow` |
| PSLR | Implementation in PyTorch | `https://github.com/pytorch/pytorch` |
| PSLO | Official PSL-MOBO implementation | `https://github.com/Xi-L/PSL-MOBO` |
| DOMOO | Official DOMOO implementation | `https://github.com/YaolinWen/DOMOO` |

## D.2 ICPSL Implementation Details

We detail the implementation of ICPSL, covering synthetic data generation, the hypernetwork architecture, objective-based pretraining, and in-context inference. Table 3 summarizes the corresponding numerical settings.

Table 3: Key configurations of the reported ICPSL model.

| Component | Configuration | Setting |
|---|---|---|
| Synthetic data | Decision dimension | $d \sim \mathcal{U}\{1, \ldots, 4\}$ |
| | Objectives dimension | $m \sim \mathcal{U}\{2, 3\}$ |
| | Context size | $N_c \sim \mathcal{U}\{8, \ldots, 64\}$ |
| | Query preferences | $Q = 128$ |
| | Sobol latent points | $K = 128$ |
| | Kernel family | RBF, Matérn-$3/2$, or Matérn-$5/2$ |
| | ARD length scale | Log-uniform on $[0.1, 0.5]$ |
| | Random Fourier features | $2{,}048$ |
| | Objective sigmoid scale | $1.5$ |
| Architecture | Maximum decision dimension | $d_{\max} = 5$ |
| | Maximum objective dimension | $m_{\max} = 4$ |
| | Context reference preferences | $J = 128$ |
| | Transformer layers | $12$ |
| | Embedding dimension | $512$ |
| | Attention heads | $4$ |
| | Feed-forward dimension | $1{,}024$ |
| | Context pooling | Average |
| | Parameter decoder hidden layers | One layer of width $2{,}048$ |
| | Parameter decoder activation | GELU |
| | Low-rank dimension | $32$ |
| | Solution MLP hidden layers | Two layers of width $256$ |
| | Solution MLP activation | ReLU |
| | Cholesky diagonal floor | $\delta = 10^{-3}$ |
| Pretraining | Optimizer | AdamW |
| | Learning rate | $3 \times 10^{-5}$ |
| | Training epochs | $300$ |
| | Steps per epoch | $256$ |
| | Warmup epochs | $5$ |
| | Learning-rate schedule | Cosine decay |
| | Effective batch size | $128$ |
| | Temperature | $\tau = 0.015$ |
| | Samples per component | $R = 1$ |

#### D.2.1 Synthetic Prior and Data Generation

For every optimization step, the decision dimension, number of objectives, and GP configurations are resampled before independent objective functions are instantiated. Each GP path uses one of the three kernel families in Table 3, independently sampled coordinate-wise ARD length scales, and $2{,}048$ random Fourier features. Applying a sigmoid maps the sampled objectives to the unit interval while preserving repeatable evaluations of each realization.

Each synthetic problem is evaluated with the context size drawn uniformly between $8$ and $64$. Pretraining uses $128$ query preferences drawn from the simplex, and $128$ randomized one-dimensional Sobol latent values shared across those preferences. Entirely new objective functions, observations,

preferences, and latent values are generated online for every training step rather than retrieved from a fixed dataset.

#### D.2.2 HYPERNETWORK ARCHITECTURE

The context encoder uses $128$ fixed reference preferences, consisting of the simplex vertices and reproducible interior points, to construct a fixed-width Tchebycheff representation for each observation. Decisions are padded to $d_{\max} = 5$ and embedded together with these scalarization features in a $512$-dimensional token space. Twelve Transformer layers with four attention heads and width-$1{,}024$ feed-forward blocks process the tokens without positional encoding. Average pooling then produces the permutation-invariant problem representation.

The pooled representation is passed through a parameter decoder with a single hidden layer of width $2{,}048$ and GELU activation. This decoder generates the problem-dependent low-rank factors, output matrix, and biases of the solution MLP, while rank-$32$ projections are shared across problems. The instantiated MLP has two hidden layers of width $256$ with ReLU activations and maps each padded preference–latent pair to a Gaussian mean and a lower-triangular scale factor. A softplus transformation with a floor of $10^{-3}$ ensures positive Cholesky diagonals, and only coordinates associated with the active decision dimensions are retained.

#### D.2.3 OBJECTIVE-BASED PRETRAINING

ICPSL is pretrained using the Tchebycheff free-energy objective described in Section 3.4. At each optimization step, the model processes an effective batch of $128$ independently sampled synthetic problems. For every problem and query preference, one reparameterized sample is drawn from each of the $128$ latent-conditioned components, evaluated on the corresponding synthetic objectives, and scored using temperature $0.015$.

We pretrain the hypernetwork for $300$ epochs, with $256$ updates per epoch, using AdamW with zero weight decay and an initial learning rate of $3 \times 10^{-5}$. The learning rate is warmed up linearly over the first five epochs and follows a cosine schedule thereafter. This pretraining jointly updates the context embeddings, Transformer encoder, parameter decoder, and shared low-rank projections. The complete pretraining process took approximately $21$ hours on four NVIDIA L20 GPUs.

#### D.2.4 IN-CONTEXT INFERENCE

At inference stage, the frozen hypernetwork generates the parameters of a problem-specific solution MLP model from the normalized offline context in a single forward pass. We use $128$ randomized Sobol latent values shared across query preferences. ICPSL-MAP draws one sample from each latent-conditioned component and selects the candidate with the highest marginal density. ICPSL-CAM averages the sigmoid-transformed component centers, whereas ICPSL-MM estimates the mixture mean using $64$ samples per component. The same generated MLP is used for every query preference without parameter updates and preference-wise optimization.

### D.3 BENCHMARK DETAILS

Our main benchmark suite comprises seven continuous synthetic functions implemented in Pymoo (Blank & Deb, 2020), and three engineering design problems implemented in RE test suit (Tanabe & Ishibuchi, 2020). Their detailed configurations are summarized in Table 4.

Table 4: Benchmark configurations for main experiments.

| Benchmark | Category | Objective construction or design task | $d$ | $m$ |
|---|---|---|---|---|
| VLMOP1 | Classical synthetic | VLMOP test function | 1 | 2 |
| VLMOP2 | Classical synthetic | VLMOP test function | 3 | 2 |
| VLMOP3 | Classical synthetic | VLMOP test function | 2 | 3 |
| AR | Composite synthetic | Ackley and Rosenbrock | 2 | 2 |
| DR | Composite synthetic | Dixon–Price and Rastrigin | 2 | 2 |
| ARA | Composite synthetic | Ackley and Rastrigin | 2 | 2 |
| ABC | Composite synthetic | Ackley, Branin, and Currin | 2 | 3 |
| RE23 | Engineering design | Pressure-vessel design | 4 | 2 |
| RE24 | Engineering design | Hatch-cover design | 2 | 2 |
| RE32 | Engineering design | Welded-beam design | 4 | 3 |

### D.4 Evaluation Protocol Details

All algorithms operate in the normalized decision domain $[0, 1]^d$. Before evaluating the objectives, each benchmark maps the proposed decisions to its original domain. Each run uses 10 offline observations generated by LHS, scrambled Sobol, or uniform sampling, and all methods receive the same context for each of the 10 random seeds. For bi-objective problems, query preferences partition the unit interval into 200 equal segments and include both endpoints, yielding 201 preferences. For problems with $m > 2$ objectives, the fixed design also contains 201 preferences, including all $m$ simplex vertices and $201 - m$ interior preferences sampled uniformly from the preference simplex. All methods use the same preference design for a given problem.

To determine objective-wise normalization ranges, we evaluate the objectives at a large set of sampled designs, using 10,000 points from a fixed unscrambled Sobol sequence. After inference, the true objectives are evaluated at the predicted solutions to form the objective set $\mathcal{Y}$. We measure solution quality by hypervolume (HV) in the normalized objective space,

$$\mathrm{HV}(\mathcal{Y}; \boldsymbol{r}) = \mathrm{vol}\left( \bigcup_{\boldsymbol{y} \in \mathcal{Y}} [\boldsymbol{y}, \boldsymbol{r}] \right), \tag{37}$$

where $\boldsymbol{r} = [1.1, ..., 1.1]^T \in \mathbb{R}^m$ is the reference point and higher values indicate better Pareto-front approximation. Over 10 independent runs, we report mean HV with the half-width of its 95% confidence interval. We also record the wall-clock times for problem-specific model construction and solution generation across all query preferences. Timing is measured after device warm-up and excludes context sampling, objective evaluation, model loading, file I/O, and device-to-CPU transfers.

# E Detailed Results for Main Experiments

We report the complete per-benchmark results for solution quality and computational efficiency under LHS, Sobol, and uniform context distributions. Overall, ICPSL achieves competitive hypervolume across the benchmark suite while requiring substantially less problem-specific training time than the baselines. In the solution-quality tables, shaded bold entries indicate the highest displayed mean hypervolume in each column. In the efficiency tables, shading and boldface indicate the lowest displayed mean training time, with ties at the reported precision highlighted together.

## E.1 Solution Quality

Table 5: Mean hypervolume (↑) with 95% confidence intervals using LHS context over 10 independent runs.

| Algorithm | VLMOP1 | VLMOP2 | VLMOP3 | AR | DR | ARA | ABC | RE23 | RE24 | RE32 |
|---|---|---|---|---|---|---|---|---|---|---|
| ParetoFlow | 1.195e+00(1.2e-03) | 1.412e-01(2.6e-02) | 9.452e-01(1.9e-01) | 6.669e-01(1.2e-01) | 1.084e+00(5.7e-02) | 5.181e-01(1.2e-01) | 1.854e-01(2.8e-02) | **1.190e+00(1.0e-02)** | **1.179e+00(2.0e-02)** | 1.316e+00(1.1e-02) |
| PSLR | 1.194e+00(2.6e-03) | 2.015e-01(4.3e-02) | **1.134e+00(1.5e-01)** | 6.608e-01(1.3e-01) | 1.117e+00(5.5e-02) | 5.560e-01(1.7e-01) | 2.925e-01(2.9e-02) | 1.182e+00(7.7e-03) | 1.155e+00(1.9e-02) | 1.302e+00(1.4e-02) |
| PSLO | **1.199e+00(1.1e-06)** | 1.040e-01(4.8e-02) | 9.881e-01(2.1e-01) | 7.005e-01(1.3e-01) | 1.090e+00(5.3e-02) | 5.328e-01(1.9e-01) | 2.624e-01(5.7e-02) | 8.673e-01(3.1e-01) | 9.071e-01(2.5e-01) | 1.160e+00(2.4e-01) |
| DOMOO | **1.199e+00(9.8e-06)** | 1.571e-01(6.7e-02) | 9.928e-01(2.3e-01) | 6.813e-01(1.4e-01) | 9.841e-01(1.3e-01) | 4.631e-01(1.2e-01) | 2.803e-01(3.9e-02) | 1.072e+00(2.0e-01) | 1.161e+00(5.5e-02) | **1.317e+00(2.0e-02)** |
| ICPSL-MAP | 1.197e+00(1.5e-03) | 1.185e-01(3.4e-02) | 9.683e-01(2.0e-01) | 6.461e-01(1.3e-01) | 1.073e+00(6.4e-02) | 5.426e-01(1.3e-01) | 2.306e-01(4.6e-02) | 1.175e+00(1.2e-02) | 1.166e+00(2.2e-02) | 1.286e+00(2.0e-02) |
| ICPSL-CAM | 1.197e+00(1.2e-03) | 3.835e-01(5.0e-02) | 1.054e+00(1.7e-01) | 7.706e-01(1.2e-01) | 1.135e+00(5.3e-02) | 6.371e-01(1.3e-01) | 3.012e-01(2.3e-02) | 1.176e+00(1.1e-02) | 1.162e+00(1.5e-02) | 1.281e+00(1.9e-02) |
| ICPSL-MM | 1.197e+00(1.2e-03) | **4.087e-01(4.9e-02)** | 1.078e+00(1.4e-01) | **7.745e-01(1.2e-01)** | **1.137e+00(5.8e-02)** | **6.601e-01(1.4e-01)** | **3.115e-01(1.8e-02)** | 1.175e+00(1.1e-02) | 1.156e+00(1.5e-02) | 1.277e+00(2.1e-02) |

Table 6: Mean hypervolume (↑) with 95% confidence intervals using Sobol context over 10 independent runs.

| Algorithm | VLMOP1 | VLMOP2 | VLMOP3 | AR | DR | ARA | ABC | RE23 | RE24 | RE32 |
|---|---|---|---|---|---|---|---|---|---|---|
| ParetoFlow | 1.192e+00(3.2e-03) | 1.672e-01(4.7e-02) | 9.878e-01(1.9e-01) | 5.787e-01(1.3e-01) | 1.089e+00(4.0e-02) | 4.686e-01(1.0e-01) | 1.859e-01(2.2e-02) | 1.196e+00(5.5e-03) | **1.197e+00(7.4e-03)** | 1.318e+00(5.1e-03) |
| PSLR | 1.197e+00(1.6e-03) | 3.334e-01(4.8e-02) | **1.196e+00(1.3e-01)** | 5.584e-01(1.2e-01) | 1.117e+00(4.2e-02) | 5.889e-01(1.3e-01) | 3.116e-01(2.7e-02) | 1.187e+00(3.9e-03) | 1.115e+00(2.4e-02) | 1.310e+00(1.2e-02) |
| PSLO | **1.199e+00(8.3e-07)** | 1.771e-01(1.1e-01) | 1.014e+00(1.9e-01) | 5.500e-01(1.5e-01) | 1.038e+00(6.4e-02) | 5.425e-01(1.3e-01) | 2.608e-01(4.5e-02) | 1.005e+00(2.1e-01) | 7.207e-01(2.7e-01) | 1.054e+00(2.8e-01) |
| DOMOO | **1.199e+00(8.7e-06)** | 3.293e-01(8.3e-02) | 1.186e+00(8.6e-02) | 6.968e-01(8.5e-02) | 8.080e-01(2.3e-01) | 5.148e-01(1.0e-01) | 2.869e-01(2.8e-02) | **1.205e+00(4.2e-03)** | 1.132e+00(6.9e-02) | **1.327e+00(3.5e-03)** |
| ICPSL-MAP | 1.198e+00(2.4e-03) | 1.496e-01(4.7e-02) | 8.693e-01(2.2e-01) | 6.083e-01(1.5e-01) | **1.122e+00(3.0e-02)** | 5.377e-01(1.2e-01) | 2.222e-01(4.4e-02) | 1.178e+00(6.3e-03) | 1.130e+00(2.5e-02) | 1.304e+00(8.7e-03) |
| ICPSL-CAM | **1.199e+00(3.6e-04)** | 4.087e-01(3.0e-02) | 9.261e-01(2.0e-01) | 7.639e-01(1.0e-01) | 1.087e+00(6.1e-02) | 5.710e-01(9.4e-02) | 3.116e-01(2.4e-02) | 1.183e+00(5.8e-03) | 1.126e+00(1.8e-02) | 1.301e+00(9.2e-03) |
| ICPSL-MM | **1.199e+00(3.8e-04)** | **4.429e-01(2.3e-02)** | 9.818e-01(1.8e-01) | **7.793e-01(1.0e-01)** | 1.089e+00(5.7e-02) | **5.932e-01(9.1e-02)** | **3.168e-01(1.7e-02)** | 1.182e+00(5.8e-03) | 1.118e+00(1.8e-02) | 1.299e+00(9.3e-03) |

Table 7: Mean hypervolume (↑) with 95% confidence intervals using uniform context over 10 independent runs.

| Algorithm | VLMOP1 | VLMOP2 | VLMOP3 | AR | DR | ARA | ABC | RE23 | RE24 | RE32 |
|---|---|---|---|---|---|---|---|---|---|---|
| ParetoFlow | 1.185e+00(9.5e-03) | 1.679e-01(4.3e-02) | 9.592e-01(1.9e-01) | 6.662e-01(1.3e-01) | 1.003e+00(5.5e-02) | 6.411e-01(1.4e-01) | 2.439e-01(2.8e-02) | **1.190e+00(1.1e-02)** | **1.169e+00(4.3e-02)** | 1.312e+00(1.0e-02) |
| PSLR | 1.196e+00(2.9e-03) | 3.301e-01(3.7e-02) | 1.043e+00(1.7e-01) | 6.447e-01(1.4e-01) | 1.081e+00(5.3e-02) | 6.002e-01(1.5e-01) | 2.794e-01(3.1e-02) | 1.184e+00(8.6e-03) | 1.089e+00(5.5e-02) | 1.297e+00(1.6e-02) |
| PSLO | 1.196e+00(6.0e-03) | 6.835e-02(2.3e-02) | 1.037e+00(1.7e-01) | 7.567e-01(1.2e-01) | 9.210e-01(9.2e-02) | 5.642e-01(1.3e-01) | 2.773e-01(3.2e-02) | 8.950e-01(2.5e-01) | 7.684e-01(2.7e-01) | 1.045e+00(3.1e-01) |
| DOMOO | **1.199e+00(3.3e-05)** | 2.162e-01(9.0e-02) | 1.011e+00(1.9e-01) | 5.823e-01(1.3e-01) | 9.521e-01(1.7e-01) | 4.719e-01(1.3e-01) | 2.037e-01(4.6e-02) | 1.114e+00(1.7e-01) | 1.015e+00(2.0e-01) | **1.330e+00(8.1e-04)** |
| ICPSL-MAP | 1.175e+00(3.4e-02) | 1.107e-01(4.1e-02) | 1.060e+00(1.6e-01) | 7.417e-01(1.4e-01) | 1.085e+00(3.7e-02) | 6.496e-01(1.5e-01) | 2.504e-01(3.3e-02) | 1.161e+00(1.3e-02) | 1.104e+00(5.3e-02) | 1.289e+00(2.4e-02) |
| ICPSL-CAM | 1.198e+00(1.2e-03) | 3.182e-01(6.9e-02) | 1.084e+00(1.5e-01) | 7.952e-01(1.3e-01) | 1.087e+00(2.8e-02) | 6.646e-01(1.3e-01) | 2.940e-01(2.3e-02) | 1.172e+00(1.2e-02) | 1.099e+00(5.1e-02) | 1.288e+00(2.1e-02) |
| ICPSL-MM | 1.198e+00(1.3e-03) | **3.631e-01(5.0e-02)** | **1.101e+00(1.3e-01)** | **8.081e-01(1.2e-01)** | **1.090e+00(2.7e-02)** | **6.714e-01(1.3e-01)** | **3.060e-01(2.2e-02)** | 1.172e+00(1.2e-02) | 1.090e+00(5.0e-02) | 1.286e+00(2.1e-02) |

### E.2 Computational Efficiency

Table 8: Mean training / inference time in seconds (↓) using LHS context over 10 independent runs.

| Algorithm | VLMOP1 | VLMOP2 | VLMOP3 | AR | DR | ARA | ABC | RE23 | RE24 | RE32 |
|---|---|---|---|---|---|---|---|---|---|---|
| ParetoFlow | 7.60e+00/8.89e-01 | 7.25e+00/8.84e-01 | 7.24e+00/9.33e-01 | 8.47e+00/9.33e-01 | 7.19e+00/8.92e-01 | 7.14e+00/8.84e-01 | 8.63e+00/1.05e+00 | 7.19e+00/8.90e-01 | 7.15e+00/8.85e-01 | 7.25e+00/9.42e-01 |
| PSLR | 2.54e+00/8.53e-04 | 2.64e+00/5.57e-04 | 2.45e+00/6.17e-04 | 2.63e+00/8.67e-04 | 2.48e+00/1.23e-03 | 2.47e+00/1.36e-03 | 2.44e+00/1.28e-03 | 2.49e+00/8.12e-04 | 2.35e+00/1.25e-03 | 2.24e+00/1.84e-02 |
| PSLO | 3.07e+00/4.62e-04 | 3.36e+00/7.12e-04 | 3.12e+00/4.81e-04 | 4.58e+00/1.06e-03 | 4.83e+00/8.63e-04 | 4.62e+00/7.92e-04 | 5.20e+00/1.08e-03 | 4.95e+00/6.29e-04 | 4.19e+00/4.99e-04 | 5.23e+00/9.04e-04 |
| DOMOO | 6.53e+00/1.98e-04 | 5.63e+00/1.90e-04 | 7.89e+00/1.83e-04 | 7.13e+00/2.04e-04 | 8.69e+00/1.94e-04 | 8.83e+00/2.18e-04 | 1.39e+01/2.22e-04 | 9.63e+00/2.11e-04 | 9.22e+00/1.82e-04 | 1.30e+01/1.90e-04 |
| ICPSL-MAP | **4.91e-02**/1.17e-03 | **3.48e-03**/1.84e-03 | **3.51e-03**/1.42e-03 | **3.21e-03**/1.38e-03 | **3.53e-03**/1.41e-03 | **6.42e-02**/1.38e-03 | **3.47e-03**/1.42e-03 | **4.75e-02**/2.25e-03 | **3.60e-03**/2.81e-03 | **3.48e-03**/2.30e-03 |
| ICPSL-CAM | **4.91e-02**/6.74e-04 | **3.48e-03**/6.62e-04 | **3.51e-03**/6.66e-04 | **3.21e-03**/6.29e-04 | **3.53e-03**/6.55e-04 | **6.42e-02**/6.77e-04 | **3.47e-03**/6.55e-04 | **4.75e-02**/6.55e-04 | **3.60e-03**/6.53e-04 | **3.48e-03**/1.62e-03 |
| ICPSL-MM | **4.91e-02**/3.38e-03 | **3.48e-03**/1.91e-03 | **3.51e-03**/1.71e-03 | **3.21e-03**/1.73e-03 | **3.53e-03**/1.73e-03 | **6.42e-02**/3.12e-03 | **3.47e-03**/1.74e-03 | **4.75e-02**/2.03e-03 | **3.60e-03**/1.69e-03 | **3.48e-03**/2.10e-03 |

Table 9: Mean training / inference time in seconds (↓) using Sobol context over 10 independent runs.

| Algorithm | VLMOP1 | VLMOP2 | VLMOP3 | AR | DR | ARA | ABC | RE23 | RE24 | RE32 |
|---|---|---|---|---|---|---|---|---|---|---|
| ParetoFlow | 7.75e+00/9.20e-01 | 8.40e+00/9.00e-01 | 7.28e+00/9.39e-01 | 7.30e+00/8.97e-01 | 7.16e+00/8.86e-01 | 7.27e+00/8.98e-01 | 7.77e+00/1.02e+00 | 7.29e+00/9.17e-01 | 8.33e+00/9.01e-01 | 8.36e+00/9.68e-01 |
| PSLR | 2.49e+00/1.32e-03 | 2.73e+00/4.17e-04 | 2.43e+00/6.01e-04 | 2.57e+00/9.40e-04 | 2.38e+00/9.86e-04 | 2.31e+00/1.32e-03 | 2.57e+00/1.09e-03 | 2.12e+00/9.25e-04 | 2.41e+00/8.07e-04 | 2.30e+00/9.56e-03 |
| PSLO | 3.24e+00/9.60e-04 | 3.51e+00/2.68e-04 | 3.73e+00/8.59e-04 | 3.93e+00/5.54e-04 | 4.91e+00/8.45e-04 | 4.48e+00/5.25e-04 | 5.59e+00/5.07e-04 | 5.23e+00/4.74e-04 | 5.24e+00/5.66e-04 | 5.56e+00/6.58e-04 |
| DOMOO | 6.91e+00/2.04e-04 | 5.80e+00/2.01e-04 | 8.99e+00/2.00e-04 | 9.00e+00/2.29e-04 | 1.01e+01/2.05e-04 | 8.11e+00/2.21e-04 | 1.36e+01/1.83e-04 | 9.15e+00/3.54e-04 | 9.85e+00/1.99e-04 | 1.30e+01/1.85e-04 |
| ICPSL-MAP | **3.24e-03**/1.22e-03 | **3.62e-03**/1.85e-03 | **3.25e-03**/1.38e-03 | **3.32e-02**/1.43e-03 | **4.66e-02**/1.48e-03 | **3.51e-03**/1.43e-03 | **4.81e-02**/1.43e-03 | **3.59e-03**/2.25e-03 | **3.52e-03**/1.44e-03 | **3.49e-03**/2.22e-03 |
| ICPSL-CAM | **3.24e-03**/6.60e-04 | **3.62e-03**/2.79e-03 | **3.25e-03**/6.39e-04 | **3.32e-02**/6.71e-04 | **4.66e-02**/6.31e-04 | **3.51e-03**/6.81e-04 | **4.81e-02**/6.67e-04 | **3.59e-03**/6.58e-04 | **3.52e-03**/6.36e-04 | **3.49e-03**/6.77e-04 |
| ICPSL-MM | **3.24e-03**/2.25e-03 | **3.62e-03**/1.86e-03 | **3.25e-03**/1.74e-03 | **3.32e-02**/1.73e-03 | **4.66e-02**/4.48e-03 | **3.51e-03**/1.77e-03 | **4.81e-02**/1.73e-03 | **3.59e-03**/2.01e-03 | **3.52e-03**/1.90e-03 | **3.49e-03**/3.67e-03 |

Table 10: Mean training / inference time in seconds (↓) using uniform context over 10 independent runs.

| Algorithm | VLMOP1 | VLMOP2 | VLMOP3 | AR | DR | ARA | ABC | RE23 | RE24 | RE32 |
|---|---|---|---|---|---|---|---|---|---|---|
| ParetoFlow | 7.71e+00/9.27e-01 | 8.57e+00/9.36e-01 | 8.40e+00/1.01e+00 | 7.23e+00/9.14e-01 | 7.32e+00/9.24e-01 | 7.24e+00/9.10e-01 | 8.63e+00/1.01e+00 | 7.16e+00/8.82e-01 | 7.22e+00/8.81e-01 | 8.51e+00/9.53e-01 |
| PSLR | 2.57e+00/7.19e-04 | 2.75e+00/7.95e-04 | 2.44e+00/1.07e-02 | 2.59e+00/1.41e-03 | 2.53e+00/1.21e-03 | 2.43e+00/1.20e-03 | 2.55e+00/1.24e-03 | 2.50e+00/1.24e-03 | 2.37e+00/9.88e-04 | 2.33e+00/1.28e-03 |
| PSLO | 2.97e+00/7.03e-04 | 2.74e+00/4.93e-04 | 3.27e+00/8.91e-04 | 4.80e+00/7.41e-04 | 4.48e+00/8.14e-04 | 4.93e+00/4.09e-04 | 4.49e+00/4.96e-04 | 4.50e+00/7.83e-04 | 4.42e+00/7.95e-04 | 5.15e+00/5.95e-04 |
| DOMOO | 5.99e+00/2.56e-04 | 5.89e+00/2.02e-04 | 1.06e+01/1.96e-04 | 6.97e+00/1.85e-04 | 9.57e+00/1.93e-04 | 9.29e+00/1.79e-04 | 1.37e+01/1.83e-04 | 9.16e+00/2.02e-04 | 9.28e+00/1.80e-04 | 1.41e+01/2.16e-04 |
| ICPSL-MAP | **4.78e-02**/1.12e-03 | **3.65e-03**/1.78e-03 | **3.63e-03**/1.40e-03 | **3.31e-03**/1.44e-03 | **3.51e-03**/1.42e-03 | **4.81e-02**/1.47e-03 | **3.23e-03**/1.42e-03 | **4.77e-02**/3.52e-03 | **3.59e-03**/1.38e-03 | **3.52e-03**/2.26e-03 |
| ICPSL-CAM | **4.78e-02**/6.60e-04 | **3.65e-03**/6.85e-04 | **3.63e-03**/6.43e-04 | **3.31e-03**/6.71e-04 | **3.51e-03**/6.19e-04 | **4.81e-02**/6.37e-04 | **3.23e-03**/6.57e-04 | **4.77e-02**/6.17e-04 | **3.59e-03**/6.42e-04 | **3.52e-03**/6.62e-04 |
| ICPSL-MM | **4.78e-02**/2.22e-03 | **3.65e-03**/2.84e-03 | **3.63e-03**/1.73e-03 | **3.31e-03**/1.73e-03 | **3.51e-03**/1.75e-03 | **4.81e-02**/1.70e-03 | **3.23e-03**/1.73e-03 | **4.77e-02**/5.71e-03 | **3.59e-03**/1.73e-03 | **3.52e-03**/2.00e-03 |

## F Detailed Results for Additional Analyses

### F.1 Evolution of Solution-Model Quality




Figure 3: Evolution of hypervolume (higher is better) for ICPSL-MAP, ICPSL-CAM, and ICPSL-MM throughout pretraining under LHS, Sobol, and uniform context distributions.

### F.2 Sensitivity to Context Size

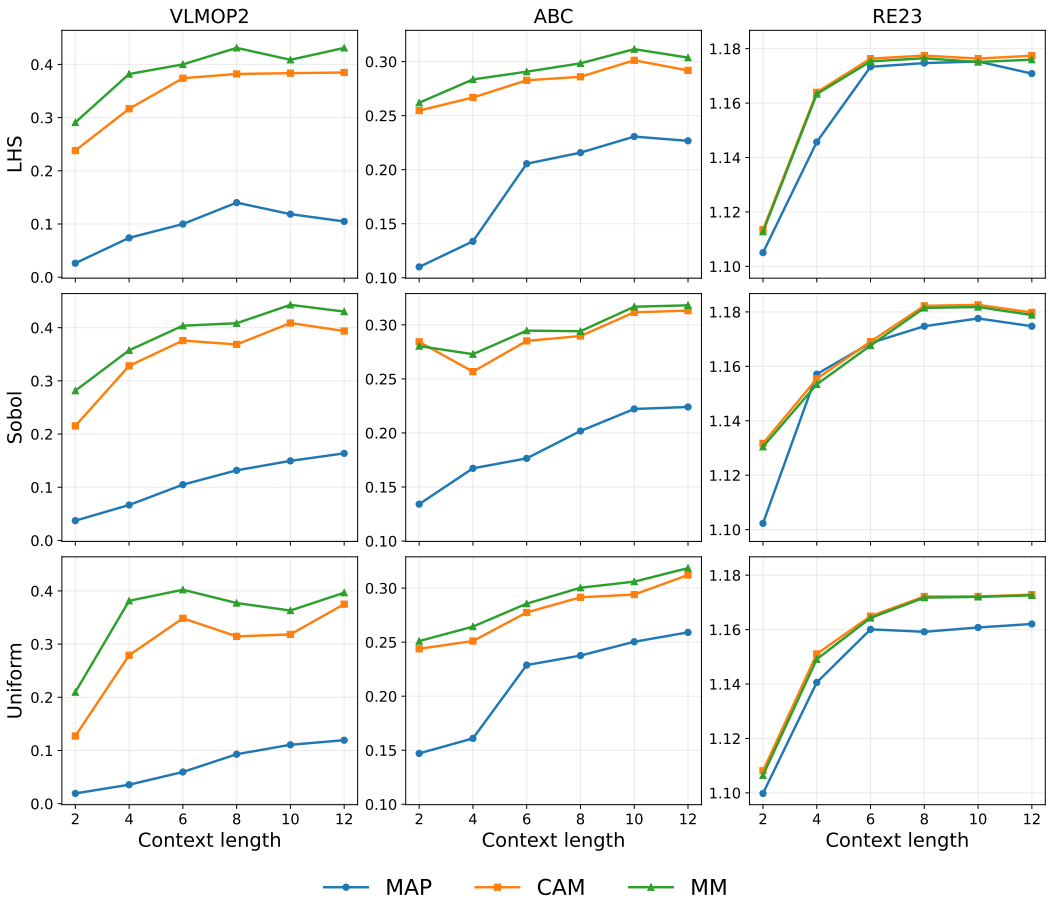



Figure 4: Hypervolume (higher is better) of ICPSL-MAP, ICPSL-CAM, and ICPSL-MM with varying context sizes under LHS, Sobol, and uniform context distributions.

### F.3 Generalization to OOD Dimensions

Table 11: Mean hypervolume (↑) with 95% confidence intervals on OOD-dimensional benchmarks using LHS context over 10 independent runs.

| Algorithm | AR (5d2m) | DR (5d2m) | ARA (5d2m) | DTLZ4 (4d4m) |
|---|---|---|---|---|
| ParetoFlow | 2.399e-01(2.5e-02) | 8.306e-01(4.3e-02) | 1.687e-01(1.3e-02) | **6.356e-01(1.1e-01)** |
| PSLR | 2.852e-01(4.9e-02) | 9.703e-01(6.8e-02) | 2.367e-01(4.0e-02) | 4.873e-01(8.3e-02) |
| PSLO | 2.397e-01(3.0e-02) | 6.403e-01(1.0e-01) | 1.441e-01(3.1e-02) | 3.958e-01(5.5e-03) |
| DOMOO | 2.193e-01(6.2e-02) | 7.125e-01(1.3e-01) | 1.312e-01(2.3e-02) | 3.922e-01(6.0e-02) |
| ICPSL-MAP | 3.065e-01(3.8e-02) | 9.214e-01(3.7e-02) | 1.736e-01(3.3e-02) | 4.538e-01(4.7e-02) |
| ICPSL-CAM | 4.945e-01(7.9e-02) | 9.875e-01(4.2e-02) | 4.593e-01(9.7e-02) | 4.032e-01(3.9e-04) |
| ICPSL-MM | **5.574e-01(7.5e-02)** | **1.025e+00(3.4e-02)** | **5.457e-01(8.1e-02)** | 4.030e-01(5.7e-04) |

Table 12: Mean hypervolume (↑) with 95% confidence intervals on OOD-dimensional benchmarks using Sobol context over 10 independent runs.

| Algorithm | AR (5d2m) | DR (5d2m) | ARA (5d2m) | DTLZ4 (4d4m) |
|---|---|---|---|---|
| ParetoFlow | 2.489e-01(2.2e-02) | 8.671e-01(3.9e-02) | 1.871e-01(2.3e-02) | **6.481e-01(8.8e-02)** |
| PSLR | 3.081e-01(3.2e-02) | 9.528e-01(5.0e-02) | 2.803e-01(5.1e-02) | 4.199e-01(3.1e-02) |
| PSLO | 2.404e-01(5.6e-02) | 7.160e-01(1.4e-01) | 1.798e-01(3.1e-02) | 3.535e-01(6.9e-02) |
| DOMOO | 2.360e-01(5.2e-02) | 5.169e-01(1.6e-01) | 1.257e-01(1.6e-02) | 3.537e-01(6.9e-02) |
| ICPSL-MAP | 3.455e-01(6.4e-02) | 9.219e-01(4.9e-02) | 2.484e-01(5.4e-02) | 4.349e-01(5.9e-02) |
| ICPSL-CAM | 5.483e-01(9.0e-02) | 9.883e-01(5.2e-02) | 4.659e-01(5.1e-02) | 4.031e-01(4.6e-04) |
| ICPSL-MM | **6.151e-01(9.8e-02)** | **9.897e-01(4.4e-02)** | **5.549e-01(5.7e-02)** | 4.029e-01(6.8e-04) |

Table 13: Mean hypervolume (↑) with 95% confidence intervals on OOD-dimensional benchmarks using uniform context over 10 independent runs.

| Algorithm | AR (5d2m) | DR (5d2m) | ARA (5d2m) | DTLZ4 (4d4m) |
|---|---|---|---|---|
| ParetoFlow | 2.969e-01(4.8e-02) | 8.684e-01(3.1e-02) | 2.104e-01(4.2e-02) | **6.231e-01(1.0e-01)** |
| PSLR | 3.537e-01(7.5e-02) | 9.302e-01(6.1e-02) | 2.809e-01(5.6e-02) | 4.888e-01(8.9e-02) |
| PSLO | 2.365e-01(4.0e-02) | 6.416e-01(1.1e-01) | 1.562e-01(3.5e-02) | 4.017e-01(1.6e-02) |
| DOMOO | 1.441e-01(3.5e-02) | 5.239e-01(1.7e-01) | 1.310e-01(4.5e-02) | 3.711e-01(7.6e-02) |
| ICPSL-MAP | 3.364e-01(5.9e-02) | 9.235e-01(5.3e-02) | 2.185e-01(6.0e-02) | 4.292e-01(2.8e-02) |
| ICPSL-CAM | 5.001e-01(5.1e-02) | 9.746e-01(4.9e-02) | 4.214e-01(5.7e-02) | 4.031e-01(5.3e-04) |
| ICPSL-MM | **5.625e-01(6.1e-02)** | **9.834e-01(4.6e-02)** | **5.084e-01(6.5e-02)** | 4.031e-01(5.2e-04) |

Table 14: Mean training / inference time in seconds (↓) on OOD-dimensional benchmarks using LHS context over 10 independent runs.

| Algorithm | AR (5d2m) | DR (5d2m) | ARA (5d2m) | DTLZ4 (4d4m) |
|---|---|---|---|---|
| ParetoFlow | 9.71e+00/5.56e+00 | 9.30e+00/5.27e+00 | 8.94e+00/4.64e+00 | 7.32e+00/9.88e-01 |
| PSLR | 4.77e-01/1.99e-04 | 5.65e-01/1.71e-04 | 5.44e-01/1.81e-04 | 6.26e-01/1.93e-04 |
| PSLO | 1.58e+00/1.76e-04 | 1.99e+00/1.78e-04 | 2.02e+00/1.87e-04 | 2.63e+00/1.74e-04 |
| DOMOO | 3.76e+00/1.63e-04 | 4.58e+00/1.61e-04 | 4.91e+00/1.73e-04 | 9.99e+00/1.64e-04 |
| ICPSL-MAP | **3.28e-03**/2.91e-03 | **3.22e-02**/5.67e-03 | **3.73e-03**/2.78e-03 | **4.67e-02**/2.18e-03 |
| ICPSL-CAM | **3.28e-03**/6.29e-04 | **3.22e-02**/6.27e-04 | **3.73e-03**/6.68e-04 | **4.67e-02**/6.63e-04 |
| ICPSL-MM | **3.28e-03**/2.33e-03 | **3.22e-02**/3.45e-03 | **3.73e-03**/2.37e-03 | **4.67e-02**/2.02e-03 |

Table 15: Mean training / inference time in seconds (↓) on OOD-dimensional benchmarks using Sobol context over 10 independent runs.

| Algorithm | AR (5d2m) | DR (5d2m) | ARA (5d2m) | DTLZ4 (4d4m) |
|---|---|---|---|---|
| ParetoFlow | 8.39e+00/3.00e+00 | 9.26e+00/4.92e+00 | 9.10e+00/5.37e+00 | 9.50e+00/5.40e+00 |
| PSLR | 5.40e-01/1.72e-04 | 5.32e-01/1.84e-04 | 5.51e-01/1.78e-04 | 5.09e-01/1.92e-04 |
| PSLO | 1.78e+00/2.24e-04 | 1.87e+00/1.76e-04 | 1.83e+00/1.85e-04 | 2.56e+00/2.67e-04 |
| DOMOO | 4.83e+00/1.96e-04 | 4.68e+00/1.65e-04 | 4.79e+00/1.63e-04 | 1.06e+01/1.71e-04 |
| ICPSL-MAP | **3.77e-03**/6.09e-03 | **3.29e-03**/2.84e-03 | **6.22e-03**/2.93e-03 | **3.61e-03**/2.22e-03 |
| ICPSL-CAM | **3.77e-03**/5.78e-03 | **3.29e-03**/6.56e-04 | **6.22e-03**/3.19e-03 | **3.61e-03**/6.45e-04 |
| ICPSL-MM | **3.77e-03**/4.20e-03 | **3.29e-03**/2.26e-03 | **6.22e-03**/2.25e-03 | **3.61e-03**/1.98e-03 |

Table 16: Mean training / inference time in seconds (↓) on OOD-dimensional benchmarks using uniform context over 10 independent runs.

| Algorithm | AR (5d2m) | DR (5d2m) | ARA (5d2m) | DTLZ4 (4d4m) |
|---|---|---|---|---|
| ParetoFlow | 9.29e+00/3.10e+00 | 9.16e+00/5.27e+00 | 9.46e+00/5.46e+00 | 9.70e+00/5.37e+00 |
| PSLR | 4.59e-01/1.75e-04 | 5.30e-01/1.82e-04 | 5.38e-01/1.84e-04 | 5.16e-01/1.84e-04 |
| PSLO | 2.01e+00/1.96e-04 | 1.97e+00/1.76e-04 | 1.88e+00/1.79e-04 | 2.54e+00/1.77e-04 |
| DOMOO | 4.63e+00/1.65e-04 | 4.68e+00/1.89e-04 | 4.95e+00/1.59e-04 | 9.55e+00/1.82e-04 |
| ICPSL-MAP | **3.17e-03**/2.95e-03 | **3.08e-03**/2.86e-03 | **3.33e-02**/2.84e-03 | **3.53e-03**/2.24e-03 |
| ICPSL-CAM | **3.17e-03**/6.40e-04 | **3.08e-03**/6.72e-04 | **3.33e-02**/6.38e-04 | **3.53e-03**/6.39e-04 |
| ICPSL-MM | **3.17e-03**/2.25e-03 | **3.08e-03**/2.25e-03 | **3.33e-02**/2.27e-03 | **3.53e-03**/1.99e-03 |