A Trajectory-Based Bayesian Approach to Multi-Objective Hyperparameter Optimization with Epoch-Aware Trade-Offs

Wenyu Wang[1] **Zheyi Fan**[2,3] **Szu Hui Ng**[1]

[1]Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore
[2]Academy of Mathematics and System Sciences, Chinese Academy of Sciences, China
[3]School of Mathematical Sciences, University of Chinese Academy of Sciences, China

Abstract

Training machine learning models inherently involves a resource-intensive and noisy iterative learning procedure that allows epoch-wise monitoring of the model performance. However, the insights gained from the iterative learning procedure typically remain underutilized in multi-objective hyperparameter optimization scenarios. Despite the limited research in this area, existing methods commonly identify the trade-offs only at the end of model training, overlooking the fact that trade-offs can emerge at earlier epochs in cases such as overfitting. To bridge this gap, we propose an enhanced multi-objective hyperparameter optimization problem that treats the number of training epochs as a decision variable, rather than merely an auxiliary parameter, to account for trade-offs at an earlier training stage. To solve this problem and accommodate its iterative learning, we then present a trajectory-based multi-objective Bayesian optimization algorithm characterized by two features: 1) a novel acquisition function that captures the improvement along the predictive trajectory of model performances over epochs for any hyperparameter setting and 2) a multi-objective early stopping mechanism that determines when to terminate the training to maximize epoch efficiency. Experiments on synthetic simulations and hyperparameter tuning benchmarks demonstrate that our algorithm can effectively identify the desirable trade-offs while improving tuning efficiency.

1 INTRODUCTION

With the expanding complexity of machine learning (ML) models, there is a significant surge in the demand for Hyperparameter Optimization (HPO). This surge is not only in pursuit of model prediction accuracy but also for ensuring the computational efficiency and robustness of models in real-world scenarios, which leads to the optimization task of finding the trade-off hyperparameter settings among multiple competing objectives $\boldsymbol{f} = \{f_1, \ldots, f_k\}$, known as Multi-Objective Hyperparameter Optimization (MOHPO) [Eggensperger et al., 2021, Karl et al., 2023, Morales-Hernández et al., 2023]. While MOHPO focuses solely on tuning hyperparameters $\boldsymbol{x}$, we extend this framework at its core by jointly tuning $\boldsymbol{x}$ and training epochs t, i.e., effectively optimizing objectives $\boldsymbol{f}(\boldsymbol{x}, t)$, to uncover superior trade-offs that emerge during iterative training (see Section 2 for more details).

Addressing HPO has long been challenging as it involves resource-intensive model training that prevents optimizers from exhaustively exploring the hyperparameter space. In this context, Bayesian Optimization (BO) has become increasingly popular [Srinivas et al., 2009, Bergstra et al., 2011, Lévesque et al., 2016, Foldager et al., 2023]. This approach builds a probabilistic surrogate model, e.g., Gaussian Process (GP) [Rasmussen and Williams, 2005], for the objective and samples a new solution by maximizing an acquisition function formulated by the prediction and uncertainty of the surrogate model. Nevertheless, traditional BO methods require observing the model performance that is fully trained after a maximum number of epochs, which could potentially lead to a waste of computational resources if early indications suggest sub-optimal performance. In general, the training behind many ML models is an iterative learning procedure where a gradient-based optimizer updates the model epoch by epoch. This procedure allows users to delineate a learning curve for any hyperparameter setting by epoch-wisely monitoring the intermediate model performance (see Figure 1(A) and (B)), benefiting from which prior research has introduced a set of epoch-efficient single-objective BO methods [Swersky et al., 2014, Dai et al., 2019, Nguyen et al., 2020, Belakaria et al., 2023] to avoid computational waste.

The concept of leveraging iterative learning to achieve more

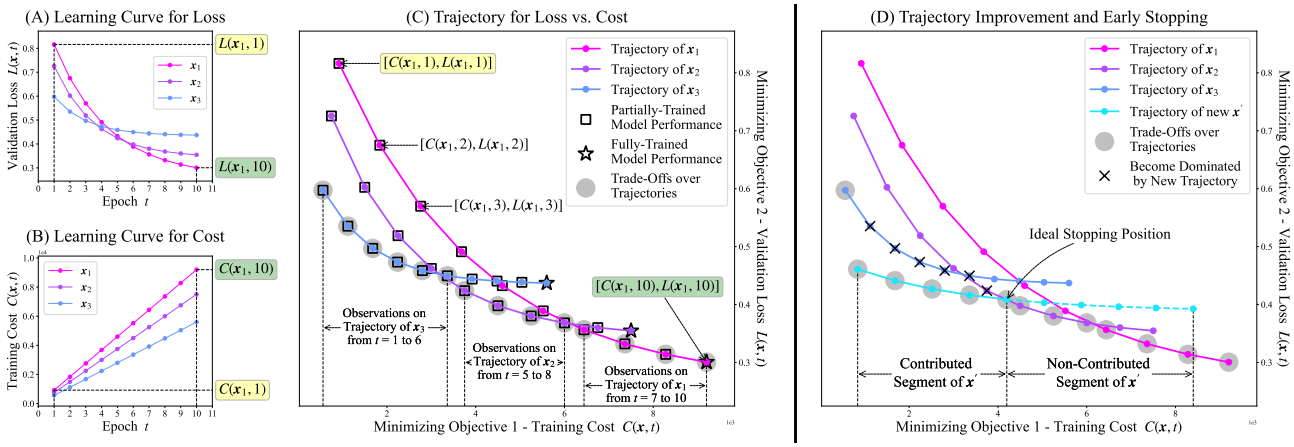

Figure 1: (A) and (B): Learning curves of three hyperparameter settings $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3 \in \mathbb{R}^d$; (C): Trajectories of $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ and trade-offs over their model performances; (D) Trajectory-based improvement and early stopping when a new $\boldsymbol{x}' \in \mathbb{R}^d$ is sampled. $L(\boldsymbol{x}, t)$ (or $C(\boldsymbol{x}, t)$) denotes the validation loss (or cost) of training with $\boldsymbol{x} \in \mathbb{R}^d$ for t epochs. The maximum number of epochs is 10.

granular control over training epochs can also be extended to MOHPO. Existing methods in multi-fidelity MOHPO [Belakaria et al., 2020, Schmucker et al., 2021] offer a relevant perspective by treating training epoch as an auxiliary parameter and aim to use the information obtained from lower fidelities (i.e., fewer training epochs) to facilitate the search for the trade-offs at the highest fidelity (i.e., the maximum training epoch). The key assumption underlying this stream of research is that higher fidelity levels generally lead to more desirable model performance. However, this assumption does not always hold in the context of ML training. For example, in the case of overfitting, a model trained for fewer epochs can outperform a fully-trained model, making it the desired outcome. This further leads to a critical question: *does a trade-off among multiple objectives emerge when the number of training epochs is fewer than the maximum allowed?*

Figure 1(C) depicts the objective space of an HPO with two objectives (training cost and validation loss), where each d-dimensional hyperparameter setting $\boldsymbol{x}$ is trained for up to 10 epochs, with its fully-trained model performance denoted by a star symbol. To better capture these dynamics, we extend the concept of the learning curve by introducing the notion of a "trajectory" to describe the evolution of model performance across epochs during the iterative training procedure. For instance, the trajectory of $\boldsymbol{x}_1 \in \mathbb{R}^d$ (denoted by the pink curve) consists of its model performances observed from epoch $t = 1$ to 10. The emergence of trajectories highlights the limitation of multi-fidelity MOHPO in that their optimality is limited to fully-trained model performances only, ignoring a large amount of partially-trained model performances ($t < 10$) that can also contribute to trade-offs.

To address the aforementioned limitation, in this study, we propose to jointly tune the hyperparameter setting and training epoch in the context of iterative learning, and formulate a novel Enhanced MOHPO (i.e., EMOHPO) whose optimization target is to uncover the trade-offs across all partially- and fully-trained model performances, or equivalently, the *trade-offs over trajectories* (depicted by the shaded points in Figure 1(C)). In fact, one important application of EMOHPO is to avoid the overfitting issue. The trade-offs of EMOHPO provide valuable insights for determining how many epochs should be allocated to achieve better generalization. Additionally, in scenarios where ML models are periodically retrained on similar datasets and need rapid deployment, such as recurrent data analyst jobs in the cloud [Casimiro et al., 2020, Mendes et al., 2020] and data drift detection in self-adaptive systems [Mahadevan and Mathioudakis, 2024, Casimiro et al., 2024], the decision maker can benefit from the trade-offs of EMOHPO to gain a better understanding of the optimal hyperparameter setting and training epoch to strike the desired balance between objectives while avoiding repetitive and costly tuning for each retraining cycle.

Meanwhile, it is important to note that while the training epoch in EMOHPO can be interpreted as a fidelity level, it is explicitly treated as a decision variable. This distinction makes the optimization methods designed for multi-fidelity MOHPO incompatible with EMOHPO, as they seek optimal hyperparameter settings at the highest fidelity level only and do not take training epoch as part of the decision-making process. To this end, we introduce a Trajectory-based MOBO (i.e., TMOBO) algorithm, which is designed to fully leverage the trajectory information for sequential sampling and granular control over training epochs. More specifically, TMOBO samples a hyperparameter setting in each iteration, as is common in BO methods; however, we introduce a specialized acquisition function that accounts for the contribution of the entire trajectory of model per-

formances, instead of any single one of them, associated with a hyperparameter setting. This extension is crucial as each trajectory may contribute to multiple trade-offs (illustrated in Figure 1(C)). Then, during the iterative learning of the sampled hyperparameter setting, TMOBO epoch-wisely updates predictions for the unobserved segment of the trajectory and decides on termination to improve algorithm efficiency. The proposed early stopping mechanism additionally ensures that the iterative learning continues until sufficient trade-offs along the trajectory have been collected, making it significantly different from the stopping criteria used in previous studies.

Contributions: (1) For the first time we formulate EMOHPO to account for the evolution of model performance across epochs and hence uncover trade-offs over trajectories which are often overlooked in multi-fidelity MOHPO studies. (2) We introduce a trajectory-based MOBO method to solve EMOHPO. The proposed method samples the next hyperparameter setting using a novel acquisition function that encapsulates trajectory information and determines when to terminate the iterative learning by a conservative early stopping mechanism. (3) Through comprehensive experiments on synthetic simulations and machine learning benchmarks, we demonstrate the effectiveness and efficiency of our method in identifying trade-offs while conserving computational resources.

2 ENHANCED MULTI-OBJECTIVE HYPERPARAMETER OPTIMIZATION

Throughout the paper, we consider the sequential minimization of an Enhanced Multi-Objective Hyperparameter Optimization problem (EMOHPO) formulated as follows,

$$\min_{(\boldsymbol{x},t)\in\mathbb{X}\times\mathbb{T}} \boldsymbol{f}(\boldsymbol{x},t) = [f_1(\boldsymbol{x},t),\ldots,f_k(\boldsymbol{x},t)]\,, \qquad (1)$$

where $\boldsymbol{f}$ comprises k objective functions, each of which represents a distinct performance measure of an ML model, e.g., validation loss and training cost. Each $\boldsymbol{x}$ denotes a d-dimensional hyperparameter setting within a compact set $\mathbb{X} \subset \mathbb{R}^d$, and $t \in \mathbb{T} = \{1,\ldots,t_{max}\}$ the number of epochs used for training. Due to the iterative learning procedure, querying any feasible pair $(\boldsymbol{x},t)$ in EMOHPO requires optimizers to sequentially observe the noisy performances $\boldsymbol{y}(\boldsymbol{x},t') = [y_1(\boldsymbol{x},t'),\ldots,y_k(\boldsymbol{x},t')]$ for each epoch $t'=1$ to t, where $y_i(\boldsymbol{x},t') = f_i(\boldsymbol{x},t') + \varepsilon_i$ and $\varepsilon_i \sim \mathcal{N}(0,\sigma_i^2)$ with variance σ_i^2 for any $i \in \{1,\ldots,k\}$. (For simplicity, we adopt the common assumption that the noise terms are i.i.d. across both hyperparameter settings and epochs [Dai et al., 2019, Klein et al., 2022].) In order words, querying $(\boldsymbol{x},t)$ results in a sequence of t queries from $(\boldsymbol{x},1)$ to $(\boldsymbol{x},t)$, which necessitates the design of a more sample-efficient strategy to avoid redundant queries of the same hyperparameter setting with different epoch numbers, especially in decreasing order.

In the context of iterative learning, each objective function $f_i(\boldsymbol{x},\cdot)$ with fixed hyperparameter setting $\boldsymbol{x}$ is generally viewed as a learning curve. However, a learning curve only allows for the analysis of one performance measure at a time. To comprehensively analyze multiple performance measures, we introduce the concept of trajectory for any hyperparameter setting $\boldsymbol{x}$ as the collection of all noise-free model performances during the training with $\boldsymbol{x}$, i.e.,

$$Trj(\boldsymbol{x}) := \{\boldsymbol{f}(\boldsymbol{x},t)\}_{t=1}^{t_{max}} = \{[f_1(\boldsymbol{x},t),\ldots f_k(\boldsymbol{x},t)]\}_{t=1}^{t_{max}}\,.$$

A trajectory inherently encapsulates the information provided by multiple learning curves. For notational convenience, let $\boldsymbol{z} = (\boldsymbol{x},t)$ and $\mathbb{Z} = \mathbb{X}\times\mathbb{T}$. We then adopt the standard definitions of Pareto optimality for multi-objective minimization problems.

Definition 1. *A solution $\boldsymbol{f}(\boldsymbol{z})$ dominates another solution $\boldsymbol{f}(\boldsymbol{z}')$, denoted by $\boldsymbol{f}(\boldsymbol{z}) \prec \boldsymbol{f}(\boldsymbol{z}')$, if and only if (1) $f_i(\boldsymbol{z}) \leq f_i(\boldsymbol{z}')$ for all $i \in \{1,\ldots,k\}$ and (2) $f_j(\boldsymbol{z}) < f_j(\boldsymbol{z}')$ for some $j \in \{1,\ldots,k\}$.*

Definition 2. *The Pareto-optimal set of $\mathbb{Z}$, denoted by Z^*, is composed of $\boldsymbol{z} \in \mathbb{Z}$ whose $\boldsymbol{f}(\boldsymbol{z})$ is not dominated by $\boldsymbol{f}(\boldsymbol{z}')$ of any other $\boldsymbol{z}' \in \mathbb{Z}$, i.e., $Z^* = \{\boldsymbol{z} \in \mathbb{Z} \mid \nexists \boldsymbol{z}' \in \mathbb{Z}, \boldsymbol{f}(\boldsymbol{z}') \prec \boldsymbol{f}(\boldsymbol{z})\}$. The corresponding set of solutions $F^* = \{\boldsymbol{f}(\boldsymbol{z}) \mid \boldsymbol{z} \in Z^*\}$ is referred to as Pareto-optimal front.*

As per Definitions 1 and 2, minimizing the EMOHPO in (1) is equivalent to locating the Pareto-optimal set over the entire search space, except that this space is composed of all feasible pairs of hyperparameter settings and training epochs. This aligns with the purpose of this study of finding the trade-offs over trajectories.

Finally, it is important to highlight that, beyond incorporating iterative learning, EMOHPO fundamentally differentiates itself from multi-fidelity MOHPO by including the training epoch as a decision variable. Compared to (1), multi-fidelity MOHPO confines its search to trade-offs among the fully-trained model performances (at the highest fidelity) only and can be expressed as,

$$\min_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{f}(\boldsymbol{x},t_{max}) = [f_1(\boldsymbol{x},t_{max}),\ldots,f_k(\boldsymbol{x},t_{max})]\,, \quad (2)$$

where the training epoch t_{max} is a fixed constant. In contrast, EMOHPO broadens the search domain from $\mathbb{X}\times\{t_{max}\}$ in multi-fidelity MOHPO to $\mathbb{X}\times\mathbb{T}$ and thus accounts for all observations on the trajectories. As a consequence, the Pareto-optimal front of EMOHPO is always superior to or at least equivalent to that of multi-fidelity MOHPO because the former additionally captures trade-offs that may emerge during iterative learning (see Figure 1(C)). This broader perspective enables more efficient decision-making in hyperparameter tuning, particularly for scenarios requiring model retraining.

3 RELATED WORK IN BAYESIAN OPTIMIZATION

Bayesian Optimization for Iterative Learning: By appropriately characterizing the learning curve, epoch-efficient BO aims to predict the fully-trained model performance based on a partially observed learning curve to avoid ineffective epochs of training. Freeze-Thaw BO [Swersky et al., 2014] introduces GP with an exponential decaying kernel to model the validation loss over time and allows the training to be paused and later resumed under hyperparameter settings that show promise. BOHB [Falkner et al., 2018], a BO extension of HyperBand [Li et al., 2018], allocates training epochs through random sampling and utilizes successive halving to eliminate suboptimal hyperparameter settings. Instead of dynamically allocating computational budget among hyperparameter settings, BO-BOS [Dai et al., 2019] combines BO with Bayesian optimal stopping to early stop the training with a hyperparameter setting predicted to yield poor model performance. Similarly, both BOIL [Nguyen et al., 2020] and BAPI [Belakaria et al., 2023] incorporate strategies for early stopping with a particular focus on considering the learning curve of the training cost. BOIL integrates the cost into the acquisition function to prioritize cost-effective ML training procedures, whereas BAPI imposes a fixed total budget over cost. Unfortunately, there has been no epoch-efficient multi-objective BO.

Multi-Objective Bayesian Optimization: Many MOBO methods have been developed for multi-objective HPO by extending the vanilla BO framework. These methods generally build a surrogate model for each objective to minimize the necessity of actual resource-intensive objective evaluations, and they differ in the implementation of acquisition function. A straightforward approach is to convert a multiple-objective problem into a single-objective problem through techniques like random scalarization, which allows a direct application of standard acquisition functions. For example, ParEGO [Knowles, 2006] and TS-TCH [Paria et al., 2020] respectively apply Expected Improvement (EI) [Jones et al., 1998] and Thompson Sampling (TS) [Thompson, 1933]. However, this approach often encounters limitations in adequately exploring the Pareto-optimal front. Therefore, acquisition functions biased towards the Pareto-optimal front, such as Expected Hypervolume Improvement (EHVI) [Emmerich et al., 2006], have been designed. Despite the effectiveness and popularity of EHVI, its computational intensity presents a significant challenge in the development of BO methods [Hupkens et al., 2015]. More recently, qEHVI [Daulton et al., 2020] and qNEHVI [Daulton et al., 2021] have extended EHVI for parallel multi-point selection and batch optimization through Monte Carlo (MC) integration [Emmerich et al., 2006], and they have demonstrated notable empirical performance. Alternatives to EHVI can be found in [Hernández-Lobato et al., 2016, Belakaria et al., 2019, Suzuki et al., 2020, Yang et al., 2022, Daulton et al., 2022].

Multi-Fidelity Bayesian Optimization: Multi-fidelity BO has a rich research history [Kandasamy et al., 2016, 2017, Sen et al., 2018, Wu et al., 2020, Fan et al., 2024] and it facilitates the optimization of fully-trained model performance by utilizing its low-fidelity approximations, which can be obtained either by using a partial training dataset or by limiting the number of training epochs. Although multi-fidelity BO shares similarities with epoch-efficient BO, it predetermines the fidelity level before initiating the iterative learning procedure and therefore ignores the observations during the procedure. For example, FABOLAS [Klein et al., 2017] considers the data subset size as the fidelity level and jointly selects the hyperparameter setting and the data subset for model training. BOCA [Kandasamy et al., 2017] expands the discrete fidelity space to continuous for a more general setting. Multi-fidelity MOBO has also been studied in the literature [Belakaria et al., 2020, Schmucker et al., 2021]; however, it is not applicable to solving EMOHPO as defined in (1). This is because multi-fidelity MOBO treats the training epoch merely as an additional degree of freedom, rather than part of the decision variables. Consequently, its objective remains focused on solving multi-fidelity MOHPO as defined in (2).

4 GAUSSIAN PROCESS FOR TRAJECTORY PREDICTION

Assume a black-box function f is sampled from a GP defined by a constant zero mean function and a kernel function $K(\boldsymbol{z}, \boldsymbol{z}')$. According to GP theory, the prior distribution over any finite set of n inputs $Z = \{\boldsymbol{z}_i\}_{i=1}^n$ is a multivariate Gaussian distribution,

$$f(Z) \sim \mathcal{N}(\mathbf{0}, K(Z, Z)),$$

where matrix $K(Z, Z) \in \mathbb{R}^{n \times n}$ with $[K(Z, Z)]_{i,j} = K(\boldsymbol{z}_i, \boldsymbol{z}_j)$. Conditioning on the corresponding observations $Y = \{y_i\}_{i=1}^n$ at Z, the posterior predictive distribution at any input $\boldsymbol{z} \in \mathbb{Z}$ is also a Gaussian distribution given by

$$f(\boldsymbol{z}) \mid Z, Y \sim \mathcal{N}\left(\mu(\boldsymbol{z}), \Sigma(\boldsymbol{z})\right), \tag{3}$$

with

$$\mu(\boldsymbol{z}) = K(\boldsymbol{z}, Z)\left[K(Z, Z) + \sigma^2 I\right]^{-1} Y,$$
$$\Sigma(\boldsymbol{z}) = K(\boldsymbol{z}, \boldsymbol{z}) - K(\boldsymbol{z}, Z)\left[K(Z, Z) + \sigma^2 I\right]^{-1} K(Z, \boldsymbol{z}),$$

where $K(\boldsymbol{z}, Z) = K(Z, \boldsymbol{z})^T \in \mathbb{R}^n$ with $[K(\boldsymbol{z}, Z)]_i = K(\boldsymbol{z}, \boldsymbol{z}_i)$. Refer to [Rasmussen and Williams, 2005] for a comprehensive review of GPs. In each iteration of the vanilla BO method, a new input $\boldsymbol{z}' \in \mathbb{Z}$ is selected by optimizing an acquisition function derived from the predictive mean μ and uncertainty Σ. Upon observing y' at $\boldsymbol{z}'$, BO advances to the next iteration with the updated input and observation sets $Z = Z \cup \{\boldsymbol{z}'\}$ and $Y = Y \cup \{y'\}$.

As each input $\boldsymbol{z} = (\boldsymbol{x}, t)$, we define the kernel function $K((\boldsymbol{x}, t), (\boldsymbol{x}', t'))$ as the product of a standard kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ over hyperparameter setting and a temporal kernel $K(t, t')$ over epochs, with the latter capturing relationships across different epochs for a fixed hyperparameter setting. For instance, Swersky et al. [2014] proposed an exponential decaying kernel to account for the validation loss that exponentially decreases over t. Belakaria et al. [2023] used a linear kernel when modeling learning curves related to training costs. Given that learning curves for different performance measures may exhibit different characteristics, in this study we employ specific temporal kernels if their behavior is known a prior. See Appendix B.1 for an illustrative example of GP prediction. By fitting a GP model for each objective function f_i, $i \in \{1, \ldots, k\}$, we can predict the trajectory $Trj(\boldsymbol{x})$ for any hyperparameter setting $\boldsymbol{x}$ and further improve the accuracy of its trajectory prediction by continuously monitoring the changes in model performance during iterative learning.

5 TRAJECTORY-BASED BAYESIAN OPTIMIZATION APPROACH

Now we introduce an epoch-efficient algorithm named Trajectory-based Multi-Objective Bayesian Optimization (i.e., TMOBO) for solving the EMOHPO as defined in (1). This algorithm is particularly designed to efficiently navigate the trade-off model performances across multiple epochs by leveraging the insights obtained from trajectories. At its core, TMOBO features a trajectory-based acquisition function to sample hyperparameter settings and a trajectory-based early stopping mechanism to determine the number of epochs to train with each hyperparameter setting. The pseudo-code of TMOBO is presented in Algorithm 1.

We initiate the algorithm by generating a set of uniformly distributed hyperparameter settings $X = \{\boldsymbol{x}_i\}_{i=1}^{n_0}$. The training datasets, including the set of query pairs Z and the set of noisy multi-objective observations Y, are then obtained by training the ML model with each $\boldsymbol{x} \in X$ for up to t_{max} epochs. Due to the unavailability of noise-free objective values, we consider the front F of the noisy observations in Y as an approximate representation of the Pareto-optimal front, which is continuously updated with each new observation.

Each iteration of TMOBO is centered around a two-level sampling strategy where a hyperparameter setting $\boldsymbol{x}'$ and its corresponding number of epochs t' are determined successively. The iteration starts with fitting $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_k]$ and $\boldsymbol{\Sigma} = [\Sigma_1, \ldots, \Sigma_k]$, with μ_i and Σ_i representing the predictive mean and uncertainty for the i-th objective function. An unvisited setting $\boldsymbol{x}'$ is then selected by maximizing an acquisition function that measures the potential improvement made by the trajectory of a hyperparameter setting as if it were to be fully trained. Thereafter, we train the ML model with $\boldsymbol{x}'$, monitor the model performance, and predict its future trajectory epoch by epoch. Once the criterion for early stopping is met, the training for $\boldsymbol{x}'$ is terminated to conserve the computational budget. Finally, TMOBO augments the primary datasets Z and Y by selecting the most informative observations associated with $\boldsymbol{x}'$.

Algorithm 1 Framework of TMOBO

Input: Initial sets of inputs Z and observations Y, and initial Pareto-optimal front F identified from Y.

```
while computational budget has not been exceeded do
    Fit k GPs with μ and Σ based on sets Z and Y.
    Sample a new x' by maximizing the TEHVI acquisition function.
    Initialize Z' ← ∅ and Y' ← ∅.
    for t' = 1 to t_max do
        Continue model training for the t'-th epoch to obtain observation y(x', t').
        Let Z' ← Z' ∪ {(x', t')} and Y' ← Y' ∪ {y(x', t')} and update front F.
        Fit k GPs with μ and Σ based on sets Z ∪ Z' and Y ∪ Y'.
        if EarlyStopping(x', t', μ, Σ, F) triggered then
            Break;
        end if
    end for
    Augment Z' and Y' into Z and Y respectively.
end while
```

5.1 TRAJECTORY-BASED ACQUISITION FUNCTION

As per Definition 3, Hypervolume (HV) evaluates the quality of a set of solutions in the objective space without any prior knowledge of actual Pareto-optimal front. The maximization over HV yields a set of solutions that are converged to and well-distributed along the Pareto-optimal front, which makes HV one of the most popular indicators used in multi-objective optimization. As per Definition 4, Hypervolume Improvement (HVI) is built upon HV and quantifies the increase in HV as the gain brought by a solution.

Definition 3. *The Hypervolume of a set of solutions $F \subset \mathbb{R}^k$ is the k-dimensional Lebesgue measure λ of the subspace dominated by F and bounded from above by a reference point $\boldsymbol{r} \in \mathbb{R}^k$, denoted by $HV(F \mid \boldsymbol{r}) = \lambda(\cup_{\boldsymbol{y} \in F} [\boldsymbol{y}, \boldsymbol{r}])$, where $[\boldsymbol{y}, \boldsymbol{r}]$ denotes the hyper-rectangle bounded by $\boldsymbol{y}$ and $\boldsymbol{r}$.*

Definition 4. *The Hypervolume Improvement of a solution $\boldsymbol{y}'$ with respect to a set of solutions F and a reference point $\boldsymbol{r} \in \mathbb{R}^k$ is the increase in hypervolume caused by including $\boldsymbol{y}'$ in set F, denoted by $HVI(\boldsymbol{y}' \mid F, \boldsymbol{r}) = HV(F \cup \{\boldsymbol{y}'\} \mid \boldsymbol{r}) - HV(F \mid \boldsymbol{r})$.*

Given that the objective values of any out-of-sample $\boldsymbol{z}$ are unknown ahead of time, the direct computation of HVI for $\boldsymbol{z}$ is infeasible. Therefore, Expected Hypervolume Improvement (EHVI) [Zitzler et al., 2007, Daulton et al., 2020] has been used in the BO framework to estimate the gain of $\boldsymbol{z}$ by taking the expectation of HVI over the predictive distribution of its objective values, i.e.,

$$\begin{aligned} EHVI(\boldsymbol{z} \mid F, \boldsymbol{r}) &= \mathbb{E}\left[HVI(\boldsymbol{f}(\boldsymbol{z}) \mid F, \boldsymbol{r})\right] \\ &= \int HVI(\boldsymbol{f}(\boldsymbol{z}) \mid F, \boldsymbol{r}) \mathbb{P}(\boldsymbol{f}(\boldsymbol{z}) \mid Z, Y) d\boldsymbol{f}. \end{aligned}$$

Recall that in our study each $\boldsymbol{z}$ is composed of a hyperparameter setting $\boldsymbol{x}$ and a specific epoch number t. Obviously, EHVI determines a hyperparameter setting $\boldsymbol{x}$ purely based on its model performance after t epochs without considering any valuable insights from the past observed or future potential model performance on the trajectory $Trj(\boldsymbol{x})$. Meanwhile, due to its joint sampling of hyperparameter setting and training epochs, EHVI ignores the fact that model performances at $\boldsymbol{z}_1 = (\boldsymbol{x}, t_1)$ and $\boldsymbol{z}_2 = (\boldsymbol{x}, t_2)$ can be observed in a single query in EMOHPO. In order words, this means that redundant model training with the same hyperparameter setting is allowed, which can cause inefficiencies in the optimization of EMOHPO.

Lemma 1. *Let X^*_{Trj} denote the set of hyperparameter settings that belong to the Pareto-optimal set of EMOHPO, i.e., $X^*_{Trj} := \{\boldsymbol{x} \in \mathbb{X} \mid \exists t \in \mathbb{T}, \nexists (\boldsymbol{x}', t') \in \mathbb{X} \times \mathbb{T}, \boldsymbol{f}(\boldsymbol{x}', t') \prec \boldsymbol{f}(\boldsymbol{x}, t)\}$. Then, the Pareto-optimal set of EMOHPO is equivalent to the Pareto-optimal set of $X^*_{Trj} \times \mathbb{T}$.*

The above lemma (see proof in Appendix A) inspires the idea of solving EMOHPO by identifying the set X^*_{Trj}, which consists of hyperparameter settings whose trajectories are "best", or more precisely, contribute to Pareto optimality, rather than directly searching for Z^*. By focusing on X^*_{Trj} and subsequently observing the trajectory of each $\boldsymbol{x}^* \in X^*_{Trj}$, we effectively reduce the search space to a lower-dimensional domain, allowing us to determine $\boldsymbol{x}$ independently of t. Within the BO framework, it can be achieved by iteratively finding the trajectory that makes the most significant improvement. To this end, we introduce the Trajectory-based EHVI (TEHVI) that operates over $\boldsymbol{x}$ only and wraps t into the trajectory $Trj(\boldsymbol{x})$,

$$\begin{aligned} TEHVI(\boldsymbol{x} \mid F, \boldsymbol{r}) &:= \mathbb{E}\left[HVI\left(Trj(\boldsymbol{x}) \mid F, \boldsymbol{r}\right)\right] \\ &= \mathbb{E}\left[HVI\left(\left\{\boldsymbol{f}(\boldsymbol{x}, t)\right\}_{t=1}^{t_{max}} \mid F, \boldsymbol{r}\right)\right]. \end{aligned}$$

By maximizing TEHVI across the hyperparameter space, we attempt to locate the hyperparameter setting that has the best trajectory regarding the current front F. However, it is worth noting that TEHVI is equivalent to the joint EHVI of multiple positions along a trajectory, which has no known analytical form and becomes particularly complicated when t_{max} is large. Therefore, following the previous studies on the fast computation of EHVI [Emmerich et al., 2006, Daulton et al., 2020], we resort to the Monte Carlo (MC) integration for approximating TEHVI, i.e.,

$$TEHVI(\boldsymbol{x} \mid F, \boldsymbol{r}) \approx \frac{1}{M} \sum_{m=1}^{M} HVI\left(\widehat{Trj}_m(\boldsymbol{x}) \mid F, \boldsymbol{r}\right),$$

where $\widehat{Tr}_m(\boldsymbol{x}) := \{\widehat{\boldsymbol{f}}_m(\boldsymbol{x}, t)\}_{t=1}^{t_{max}}$ denotes a predictive trajectory of $\boldsymbol{x}$ sampled from the joint posterior of GPs and M denotes the total number of samples. To further alleviate the computational burden, we adopt the candidate search strategy that maximizes the approximated TEHVI over a fixed-size set of candidate hyperparameter settings. Each candidate is generated by adding a Gaussian perturbation to the evaluated hyperparameter setting whose trajectory has contributed to the current front the most. It has been shown that such a candidate search guarantees asymptotic convergence to the global optimum [Regis and Shoemaker, 2007, Wang et al., 2023]. More importantly, it enables the simultaneous computation of TEHVI for multiple candidates in a batch to significantly improve efficiency. Please refer to Appendix B for more details.

5.2 EARLY STOPPING AND AUGMENTATION

The Pareto-optimal front of EMOHPO is generally composed of trajectory segments of different hyperparameter settings because trajectories can intertwine within the objective space. As illustrated by Figure 1(D), the trajectory of a newly sampled hyperparameter setting $\boldsymbol{x}'$ is split into contributed and non-contributed segments where the former pushes the front forward while the latter falls into the area already dominated by the front. Intuitively, during the iterative learning procedure, as we move from the contributed towards the non-contributed segment, the procedure should be immediately terminated at the end of the contributed segment, i.e., the ideal stopping position.

However, the complete trajectory $Trj(\boldsymbol{x}')$ cannot be observed until the ML model has been fully trained with the hyperparameter setting $\boldsymbol{x}'$. To this end, we estimate a conservative stopping epoch by considering both predictive mean and uncertainty associated with the positions along the trajectory,

$$t^* = \sup\left\{t \in \mathbb{T} \mid \boldsymbol{\mu}(\boldsymbol{x}', t) - \beta^{\frac{1}{2}} \boldsymbol{\Sigma}(\boldsymbol{x}', t) \prec \boldsymbol{y}, \exists \boldsymbol{y} \in F\right\},$$

where β is a predetermined constant controlling the confidence level. Inspired by the Lower Confidence Bound (LCB) [Srinivas et al., 2009], the conservative stopping epoch is the maximum number of epochs after which any future model performance is unlikely to improve the current front with a high probability. As the iterative learning procedure proceeds by epoch, the trajectory prediction is progressively updated based on the new observations on $Trj(\boldsymbol{x}')$ and so is

the conservative stopping epoch. The iterative learning procedure terminates once the current training epoch t' exceeds the conservative stopping epoch t^*.

Consequently, at the end of each iteration of TMOBO, we obtain the temporary datasets $Z' = \{(\boldsymbol{x}', t)\}_{t=1}^{t'}$ and $Y' = \{\boldsymbol{y}(\boldsymbol{x}', t)\}_{t=1}^{t'}$, where the actual stopping epoch t' can take any value from $\{1, \ldots, t_{max}\}$. However, retaining all new observations in Z' and Y' for training GP models in subsequent iterations is inefficient, especially when t_{max} is large. Following the active data augmentation used in [Nguyen et al., 2020, Belakaria et al., 2023], we opt to augment only a subset of Z' to the primary dataset Z by sequentially selecting an input $(\boldsymbol{x}', t) \in Z'$ at which the GP predictive uncertainty is highest. As more than one GP models are used to approximate the objectives, similar to the scenario considered in [Belakaria et al., 2023], the model uncertainty of $(\boldsymbol{x}', t)$ is computed as the sum of normalized variances predicted by each GP model at $(\boldsymbol{x}', t)$. Through this method, we can effectively control the increase in the size of the training set while ensuring the accuracy and training efficiency of GP models.

6 NUMERICAL EXPERIMENTS

In this section, we conduct a comprehensive empirical analysis of the performance of TMOBO on several synthetic and real-world benchmark problems that are formulated as EMOHPO. Recall that EMOHPO is essentially a multi-objective optimization problem defined over the combined space of hyperparameter settings and training epochs. Therefore, we compare TMOBO with several state-of-the-art MOBO methods, namely ParEGO [Knowles, 2006], qEHVI [Daulton et al., 2020], and qNEHVI [Daulton et al., 2021]. Notably, as discussed in Sections 1 and 3, we exclude multi-fidelity optimization algorithms from the comparison because they do not consider the possibility that earlier epochs (or lower fidelity levels) could contribute to Pareto-optimal set or front. As a result, these methods are not applicable to solving EMOHPO. To ensure a fair comparison, we further enhance selected MOBO methods by collecting all the observations $\{\boldsymbol{y}(\boldsymbol{x}, 1), \ldots, \boldsymbol{y}(\boldsymbol{x}, t)\}$ into their results when sampling at the pair $(\boldsymbol{x}, t)$. For clarity, in the following discussion, we use ParEGO-T to represent the enhanced version of ParEGO, and similarly for qEHVI-T and qNEHVI-T.

Considering that all algorithms are stochastic, we perform 20 independent trial runs of each algorithm on each test problem. The logarithm of the HV difference between the front found by an algorithm and the true Pareto-optimal front is computed to measure the performance [Daulton et al., 2020, 2021]. Since the true Pareto-optimal front is generally unknown in practice, we approximate it by aggregating all observed solutions across all algorithms and trials and extracting the non-dominated set to form an empirical Pareto front. Although the empirical front may not fully capture the true Pareto front, this ensures an unbiased comparison, as no algorithm is favored a priori, and each method's performance is evaluated relative to the best-known solutions discovered collectively. Similarly, the reference point is set as the least favorable solution with each dimension corresponding to the worst observed value of an objective so that it upper bounds all observations in the objective space for HV computation. Further details on algorithm configurations and corresponding sensitivity analyses are provided in Appendices C.1 and D, respectively. In the rest of this section, we showcase and analyze the main numerical results of this study but refer interested readers to Appendix E for detailed results and additional experiments.

6.1 SYNTHETIC SIMULATIONS

The numerical experiments start with running TMOBO and alternative algorithms on a set of synthetic problems, through which we assess TMOBO's capability to leverage the trajectory information under different trajectory characteristics. Each synthetic problem (with objectives $[f_1(\boldsymbol{x}, t), \ldots, f_k(\boldsymbol{x}, t)]$) is formulated as an epoch-dependent counterpart of a standard multi-objective benchmark problem (with objectives $[\bar{f}_1(\boldsymbol{x}), \ldots, \bar{f}_k(\boldsymbol{x})]$), where $f_i(\boldsymbol{x}, t) = \bar{f}_i(\boldsymbol{x}) \cdot g_i(t)$ and curve function $g_i(t)$ emulates the iterative learning procedure. This construction approach enables us to diversify the trajectory characteristics of a synthetic problem by specifying the shape of $g_i(t)$ associated with each objective. In this study, we define the curve function $g_i(t)$ as either monotonic (M), quadratic (Q), or periodic (P) and constrain $g_i(t)$ to be positive to preserve the challenges posed by the original multi-objective benchmark problem (see Appendix C.2).

We select five widely used multi-objective benchmark problems from ZDT [Zitzler et al., 2000] and DTLZ [Deb et al., 2002] test suites with $d = 5$ and add Gaussian noise with a standard deviation of 1% of the range of each objective. Each subplot of Figure 2 depicts the box plots over four synthetic problems with different trajectory characteristics derived from the same multi-objective benchmark. For instance, "ZDT1(M-P)" refers to the synthetic test problem derived from ZDT1 with the first objective multiplied by the monotonic curve function and the second objective by the periodic curve function. It can be observed that with a maximum budget of 150 iterations (or the number of times the iterative learning procedure is executed), TMOBO consistently achieves the lowest HV difference among all synthetic problems, and the solutions obtained by TMOBO generally dominate a large proportion of those obtained by alternative algorithms that do not exploit trajectory information. After being enhanced by trajectory observations, the performance of two HV-based methods, qNEHVI-T and qEHVI-T, is comparable to TMOBO on ZDT2(M-M) and ZDT2(M-Q) while ParEGO-T has the worst performance. These findings

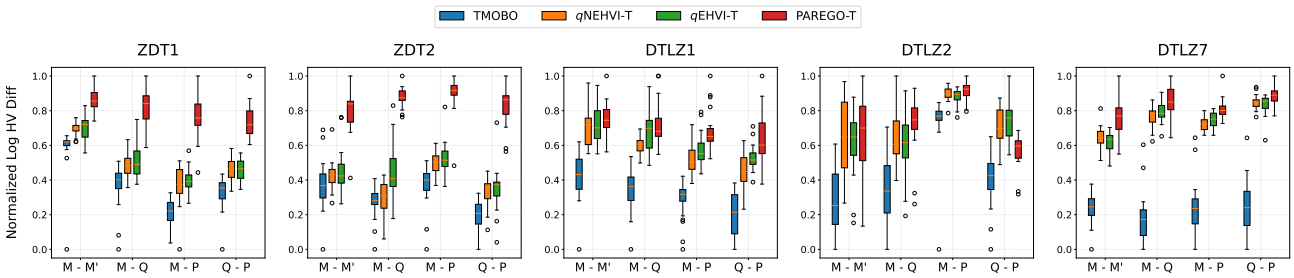

Figure 2: Box plots for 20 problems derived from ZDT1, ZDT2, DTLZ1, DTLZ2, and DTLZ7. Each algorithm runs for 20 independent trials. The logarithm of Hypervolume difference is computed at the end of each trial and is normalized in $[0, 1]$.

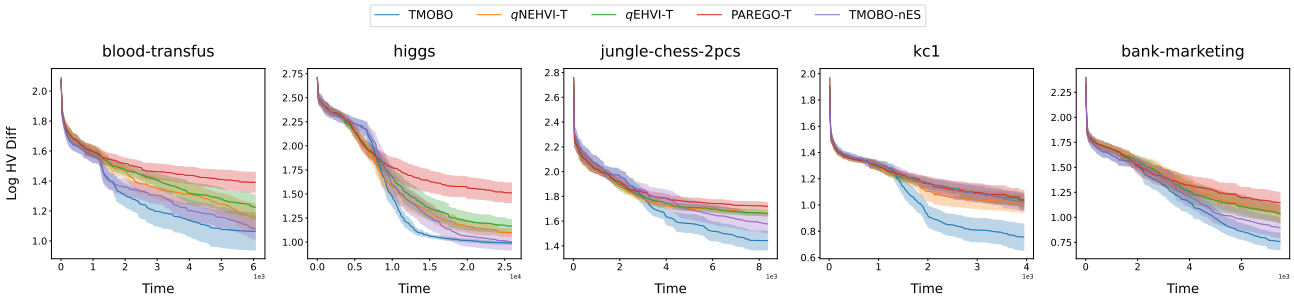

Figure 3: Average log Hypervolume difference against time for each algorithm on five different hyperparameter tuning tasks. Each algorithm runs for 20 independent trials. The shaded region indicates two standard errors of the mean.

indicate that the trajectory information is beneficial to the optimization of EMOHPO and that TMOBO is able to maintain its advantages when processing trajectories with even complicated characteristics.

6.2 HYPERPARAMETER TUNING BENCHMARKS

Five different Multi-Layer Perceptron (MLP) hyperparameter tuning tasks obtained from LCBench [Zimmer et al., 2021] are first utilized to examine the performance of TMOBO, where each task aims to optimize five hyperparameters (i.e., learning rate, momentum, weight decay, max dropout rate, and max number of units) by minimizing validation loss and training cost simultaneously on a specific training dataset. Following previous work [Falkner et al., 2018, Martinez-Cantin, 2018, Daxberger et al., 2019, Perrone et al., 2018], we utilize the surrogate version of LCBench implemented in YAHPO Gym [Pfisterer et al., 2022] and HPOBench [Eggensperger et al., 2021], where the performance metrics (i.e., objectives) are approximated by a high-quality surrogate. This approach avoids biased implementation errors, enables extensive testing, and ensures reproducibility across any environments (see Appendix C.2). On each dataset, we observe epoch-wise model performance for any feasible hyperparameter settings up to 50 epochs.

Figure 3 compares algorithm performance over 20 independent replications, measured against cumulative model training time (excluding algorithm overhead) since model training is generally more computationally expensive and dominates hyperparameter optimization costs. This ensures a clearer understanding of how optimization efforts scale with the model complexity, thereby facilitating a more insightful comparison across different models and optimization methods. Besides the enhanced MOBO methods, we also implement a variant of TMOBO, named TMOBO-nES, which does not have an early stopping mechanism and trains each sampled setting for the maximum number of epochs. We note that TMOBO and TMOBO-nES spend more time in initialization as they train the initial settings thoroughly to capture the trajectory characteristics. Despite the initialization, given the same time budget, TMOBO achieves significantly lower HV difference than the other enhanced MOBO methods across all tasks. Meanwhile, the average performance of TMOBO surpasses TMOBO-nES, which shows the advantage of using an early stopping mechanism. While TMOBO-nES aims to learn better about trajectory characteristics through full model training, the early stopping mechanism enables TMOBO to save effort by terminating non-contributed training to explore more regions of interest.

We further apply TMOBO, TMOBO-nES, and qNEHVI-T to a more challenging CNN-based task: tuning MobileNetV2 [Sandler et al., 2018] on CIFAR-10 image dataset by optimizing learning rate, momentum, weight decay, batch size, and training epochs. However, unlike LCBench experiments, here we measure total wall-clock time, including CNN training and algorithm overhead. Figure 4 shows that

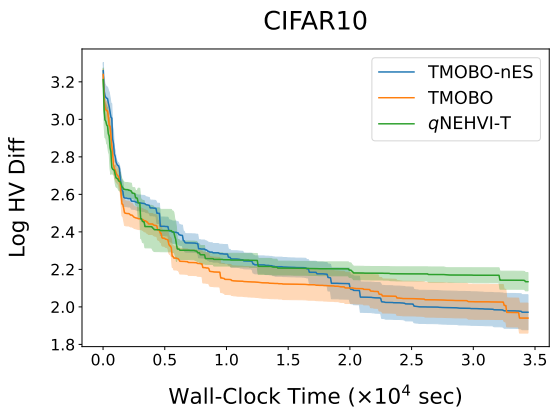

Figure 4: Average log HV difference against wall-clock time for each algorithm on the hyperparameter tuning task of MobileNetV2 on the CIFAR-10 dataset. The shaded region indicates two standard errors of the mean.

after 35,000 seconds, both TMOBO and TMOBO-nES surpass qNEHVI-T in HV difference, with TMOBO converging faster early on due to early stopping. Though TMOBO-nES eventually matches TMOBO's performance, the latter's efficiency in the initial phase highlights its practical advantage for computationally expensive tasks. Given the inherent complexity of this CNN-based task, we believe these results provide strong evidence for the scalability and practical applicability of TMOBO.

7 CONCLUSIONS

In this study, we consider multi-objective hyperparameter optimization with iterative learning procedures. Our interest centers on how trajectory information affects the distribution of trade-offs and on how to leverage this information to perform an effective and efficient search for trade-offs. To this end, we extend the conventional MOHPO problem to EMOHPO by including the training epoch as an explicit decision variable so as to reveal the trade-offs that may occur along trajectories. These frequently overlooked trade-offs play a beneficial role in decision-making for addressing overfitting issue and optimizing ML model deployment with retraining schemes. As there are no algorithms specially designed to handle EMOHPO, we then propose the TMOBO algorithm to first sample the hyperparameter setting with the largest trajectory-based contribution and then determine when to early stop the training with it based on the trajectory predictions of GP models.

Through experiments on synthetic simulations and hyperparameter tuning benchmarks, TMOBO has demonstrated its advantage in locating better solutions by exploiting the trajectory information compared to traditional multi-objective optimization methods. Considering that the iterative processing procedure inherent in many real-world simulations or experiments, such as drug design and material engineering, shares similar characteristics with the training of ML models, it is meaningful to explore the formulation of EMOHPO problems across a variety of practical scenarios and to extend the success of TMOBO algorithm in this study. However, optimizing the TEHVI acquisition function remains challenging. Therefore, further exploitation is needed to derive an analytical form of TEHVI or to develop more efficient approximations.

Acknowledgements

We gratefully acknowledge the anonymous reviewers for their valuable feedback and constructive suggestions. We would also like to thank Songhao Wang and Haowei Wang for their helpful discussions regarding this study. Szu Hui Ng's work is supported in part by the Ministry of Education, Singapore (Grant: R-266-000-149-114).

References

Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Multi-fidelity multi-objective bayesian optimization: An output space entropy search approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10035–10043, 2020.

Syrine Belakaria, Janardhan Rao Doppa, Nicolo Fusi, and Rishit Sheth. Bayesian optimization over iterative learners with structured responses: A budget-aware planning approach. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 9076–9093, 2023.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

Maria Casimiro, Diego Didona, Paolo Romano, Luis Rodrigues, Willy Zwaenepoel, and David Garlan. Lynceus: Cost-efficient tuning and provisioning of data analytic jobs. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 56–66, 2020.

Maria Casimiro, Diogo Soares, David Garlan, Luís Rodrigues, and Paolo Romano. Self-adapting machine learning-based systems via a probabilistic model checking framework. *ACM Transactions on Autonomous and Adaptive Systems*, 2024.

Zhongxiang Dai, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. Bayesian optimization meets bayesian optimal stopping. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 1496–1506, 2019.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 9851–9864, 2020.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. In *Advances in Neural Information Processing Systems*, volume 34, pages 2187–2200, 2021.

Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-objective bayesian optimization over high-dimensional search spaces. In *Proceedings of the 38th Uncertainty in Artificial Intelligence Conference*, pages 507–517, 2022.

Erik Daxberger, Anastasia Makarova, Matteo Turchetta, and Andreas Krause. Mixed-variable bayesian optimization. *arXiv preprint arXiv:1907.01329*, 2019.

Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 1, pages 825–830, 2002.

Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, René Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. Hpobench: A collection of reproducible multi-fidelity benchmark problems for hpo. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

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

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1437–1446, 2018.

Mingzhou Fan, Byung-Jun Yoon, Edward Dougherty, Francis Alexander, Nathan Urban, Raymundo Arroyave, and Xiaoning Qian. Multi-fidelity bayesian optimization with multiple information sources of input-dependent fidelity. In *Proceedings of the 40th Uncertainty in Artificial Intelligence Conference*, 2024.

Jonathan Foldager, Mikkel Jordahn, Lars K Hansen, and Michael R Andersen. On the role of model uncertainties in bayesian optimisation. In *Proceedings of the 39th Uncertainty in Artificial Intelligence Conference*, pages 592–601, 2023.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, and Joaquin Vanschoren. An open source automl benchmark. *arXiv preprint arXiv:1907.00909*, 2019.

Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pages 1492–1501, 2016.

Iris Hupkens, André Deutz, Kaifeng Yang, and Michael Emmerich. Faster exact algorithms for computing expected hypervolume improvement. In *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization*, pages 65–79, 2015.

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

Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity bayesian optimisation with continuous approximations. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1799–1808, 2017.

Florian Karl, Tobias Pielok, Julia Moosbauer, Florian Pfisterer, Stefan Coors, Martin Binder, Lennart Schneider, Janek Thomas, Jakob Richter, Michel Lang, et al. Multi-objective hyperparameter optimization in machine learning — an overview. *ACM Transactions on Evolutionary Learning and Optimization*, 3(4):1–50, 2023.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 528–536, 2017.

Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. In *International Conference on Learning Representations*, 2022.

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

Julien-Charles Lévesque, Christian Gagné, and Robert Sabourin. Bayesian hyperparameter optimization for ensemble learning. In *Proceedings of the 32nd Uncertainty in Artificial Intelligence Conference*, 2016.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Ananth Mahadevan and Michael Mathioudakis. Cost-aware retraining for machine learning. *Knowledge-Based Systems*, 293:111610, 2024.

Ruben Martinez-Cantin. Funneled bayesian optimization for design, tuning and control of autonomous systems. *IEEE Transactions on Cybernetics*, 49(4):1489–1500, 2018.

Pedro Mendes, Maria Casimiro, Paolo Romano, and David Garlan. Trimtuner: Efficient optimization of machine learning jobs in the cloud via sub-sampling. In *28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–8, 2020.

Alejandro Morales-Hernández, Inneke Van Nieuwenhuyse, and Sebastian Rojas Gonzalez. A survey on multi-objective hyperparameter optimization algorithms for machine learning. *Artificial Intelligence Review*, 56(8): 8043–8093, 2023.

Vu Nguyen, Sebastian Schulze, and Michael Osborne. Bayesian optimization for iterative learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 9361–9371, 2020.

Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A flexible framework for multi-objective bayesian optimization using random scalarizations. In *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*, volume 115, pages 766–776, 2020.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. Yahpo gym-an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *Proceedings of the First International Conference on Automated Machine Learning*, volume 188, pages 1–39, 2022.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 11 2005.

Rommel G Regis and Christine A Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4):497–509, 2007.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

Robin Schmucker, Michele Donini, Muhammad Bilal Zafar, David Salinas, and Cédric Archambeau. Multi-objective asynchronous successive halving. *arXiv preprint arXiv:2106.12639*, 2021.

Rajat Sen, Kirthevasan Kandasamy, and Sanjay Shakkottai. Multi-fidelity black-box optimization with hierarchical partitions. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4538–4547, 2018.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, page 1015–1022, 2009.

Shinya Suzuki, Shion Takeno, Tomoyuki Tamura, Kazuki Shitara, and Masayuki Karasuyama. Multi-objective bayesian optimization using pareto-frontier entropy. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 9279–9288, 2020.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15 (2):49–60, 2014.

Wenyu Wang and Christine A Shoemaker. Reference vector assisted candidate search with aggregated surrogate for computationally expensive many objective optimization problems. *INFORMS Journal on Computing*, 35(2):318–334, 2023.

Wenyu Wang, Taimoor Akhtar, and Christine A Shoemaker. Efficient multi-objective optimization through parallel surrogate-assisted local search with tabu mechanism and asynchronous option. *Engineering Optimization*, pages 1–17, 2023.

Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical multi-fidelity bayesian optimization for hyperparameter tuning. In *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*, pages 788–798, 2020.

Kaifeng Yang, Michael Affenzeller, and Guozhi Dong. A parallel technique for multi-objective bayesian global optimization: Using a batch selection of probability of improvement. *Swarm and Evolutionary Computation*, 75: 101183, 2022.

Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021.

Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, pages 862–876, 2007.

A PROOF FOR LEMMA 1

Recall that Lemma 1 claims that the Pareto-optimal set of $X^*_{Trj} \times \mathbb{T}$ is equal to the Pareto-optimal set of EMOHPO, where $X^*_{Trj} = \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{X}, \exists t \in \mathbb{T}, \nexists(\boldsymbol{x}', t') \in \mathbb{X} \times \mathbb{T}, \boldsymbol{f}(\boldsymbol{x}', t') \prec \boldsymbol{f}(\boldsymbol{x}, t)\}$. We start by establishing the following lemma.

Lemma 2. *Let $Z \subset \mathbb{R}^d$ and Z^* denote its Pareto-optimal set. If Z_U is a subset of Z such that $Z^* \subseteq Z_U \subseteq Z$, the Pareto-optimal set of Z_U, denoted by Z^*_U, is equal to Z^*, i.e., $Z^*_U = Z^*$.*

Proof. As per Definition 2, we have,

$$Z^* = \{\boldsymbol{z} \mid \boldsymbol{z} \in Z, \nexists \boldsymbol{z}' \in Z, \boldsymbol{f}(\boldsymbol{z}') \prec \boldsymbol{f}(\boldsymbol{z})\} \tag{4}$$

and

$$Z^*_U = \{\boldsymbol{z} \mid \boldsymbol{z} \in Z_U, \nexists \boldsymbol{z}' \in Z_U, \boldsymbol{f}(\boldsymbol{z}') \prec \boldsymbol{f}(\boldsymbol{z})\}. \tag{5}$$

(a) Let $\boldsymbol{z} \in Z^* \subseteq Z_U$. There does not exist $\boldsymbol{z}' \in Z$ such that $\boldsymbol{f}(\boldsymbol{z}') \prec \boldsymbol{f}(\boldsymbol{z})$. Since $Z_U \subseteq Z$, there does not exist $\boldsymbol{z}' \in Z_U$ such that $\boldsymbol{f}(\boldsymbol{z}') \prec \boldsymbol{f}(\boldsymbol{z})$. By (5), $\boldsymbol{z} \in Z^*_U$ and hence $Z^* \subseteq Z^*_U$.

(b) Assume that $Z^*_U \nsubseteq Z^*$. Then, there exists $\boldsymbol{z}$ satisfying

1. $\boldsymbol{z} \in Z^*_U \Rightarrow \nexists \boldsymbol{z}' \in Z_U$ such that $\boldsymbol{f}(\boldsymbol{z}') \prec \boldsymbol{f}(\boldsymbol{z})$;

2. $\boldsymbol{z} \notin Z^* \Rightarrow \exists \boldsymbol{z}'' \in Z$ such that $\boldsymbol{f}(\boldsymbol{z}'') \prec \boldsymbol{f}(\boldsymbol{z})$.

Therefore, $\exists \boldsymbol{z}'' \in Z \setminus Z_U$ such that $\boldsymbol{f}(\boldsymbol{z}') \prec \boldsymbol{f}(\boldsymbol{z})$. Since $Z^* \subseteq Z_U$, $Z^* \cap (Z \setminus Z_U) = \emptyset$ and $\boldsymbol{z}'' \notin Z^*$. Then, $\exists \boldsymbol{z}''' \in Z^* \subseteq Z_U$ such that $\boldsymbol{f}(\boldsymbol{z}''') \prec \boldsymbol{f}(\boldsymbol{z}'') \prec \boldsymbol{f}(\boldsymbol{z})$, which contradicts to the first condition $\boldsymbol{z} \in Z^*_U$. Therefore, $Z^*_U \subseteq Z^*$.

Combining these two inclusions together, we conclude that $Z^*_U = Z^*$. □

As each query pair $\boldsymbol{z} = \{\boldsymbol{x}, t\}$, the Pareto-optimal set Z^* of EMOHPO, i.e., the trade-offs over trajectories, can be equivalently expressed as,

$$Z^* = \{(\boldsymbol{x}, t) \mid (\boldsymbol{x}, t) \in \mathbb{X} \times \mathbb{T}, \nexists(\boldsymbol{x}', t') \in \mathbb{X} \times \mathbb{T}, \boldsymbol{f}(\boldsymbol{x}', t') \prec \boldsymbol{f}(\boldsymbol{x}, t)\}. \tag{6}$$

Then, we have,

$$\begin{aligned} Z^* &\subseteq \{(\boldsymbol{x}, t'') \mid \boldsymbol{x} \in \mathbb{X}, \exists t \in \mathbb{T}, (\boldsymbol{x}, t) \in Z^*, t'' \in \mathbb{T}\} \\ &= \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{X}, \exists t \in \mathbb{T}, (\boldsymbol{x}, t) \in Z^*\} \times \{t'' \mid t'' \in \mathbb{T}\} = X^*_{Trj} \times \mathbb{T}. \end{aligned} \tag{7}$$

Since $Z^* \subseteq X^*_{Trj} \times \mathbb{T} \subset \mathbb{X} \times \mathbb{T}$, we complete the proof by applying Lemma 2.

B ALGORITHM DETAILS

B.1 GAUSSIAN PROCESS PREDICTION IN TMOBO

We begin by visualizing GP's capability of predicting learning curve at a single iteration while running TMOBO on a real hyperparameter tuning benchmark kc1. The prediction results are illustrated in Figure B.1. Initially, even though the learning curve (depicted by orange stars) follows an uncommon shape and remains entirely unobserved, the GP model, which has been trained on data from the first 35 iterations, provides a high-quality approximation of the learning curve (depicted by the green curve) of the selected hyperparameter setting $\boldsymbol{x}$. This allows TMOBO to establish a solid foundation for trajectory prediction by leveraging multiple GP models simultaneously and computing a reliable TEHVI acquisition value before the algorithm decides to start the iterative learning process at $\boldsymbol{x}$.

As the iterative learning process progresses for the selected setting $\boldsymbol{x}$, true but noisy observations are revealed to the GP model epoch by epoch. These observations continuously enhance the predictive quality of the GP model. As shown in Figure B.1, after incorporating the revealed observations (depicted by red stars) from the learning curve, the updated GP

predictions align more closely with the true data. This refinement leads to a substantial reduction in prediction uncertainty, enabling TMOBO to make more informed decisions for early stopping.

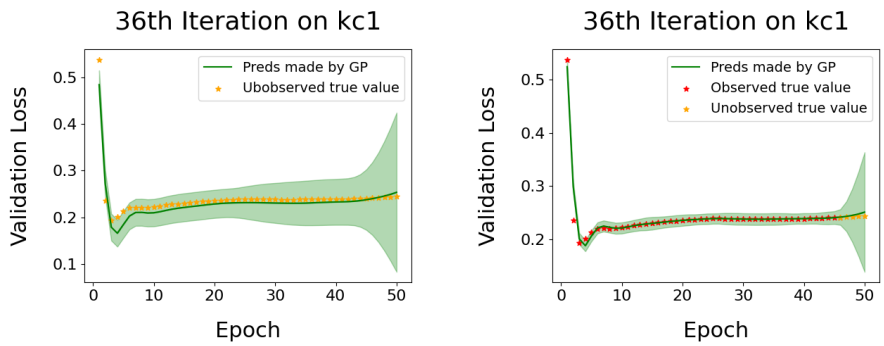

Figure B.1: An illustrative example of GP prediction obtained after running TMOBO on the kc1 hyperparameter tuning task for 35 iterations. [Left] GP prediction (green curve) of validation loss for the selected hyperparameter setting over epochs 1 to 50 before any true observations (orange stars) from its learning curve are known. [Right] Updated GP predictions after some observations (red stars) on the learning curve are revealed and incorporated into the GP model.

B.2 CANDIDATE SEARCH STRATEGY

Acquisition function plays a critical part in the BO framework but maximizing it presents inherent challenges. As discussed in Section 5.1, the TEHVI acquisition function extended from EHVI has no known analytical form and is multi-modal. Furthermore, approximating TEHVI by the MC method incurs significant computational overhead, making it hard to optimize efficiently. To this end, our algorithm TMOBO adopts a candidate search to determine new hyperparameter settings by strategic sampling instead of iterative optimization of the acquisition function. The candidate search first proposed by Regis and Shoemaker [2007] aims to generate a set of random settings (termed as "candidates") around a previously visited high-quality hyperparameter setting (termed as "center"). The new hyperparameter setting is then selected from these candidates by comparing their respective acquisition function values. This approach has demonstrated its effectiveness in global optimization [Regis and Shoemaker, 2007] and has been extended to multi-objective cases in [Wang et al., 2023, Wang and Shoemaker, 2023]. The specific implementation details of the candidate search in TMOBO are provided below:

Center Selection. The effectiveness of a candidate search largely depends on the quality of the center chosen from previously visited hyperparameter settings. Intuitively, the trajectories of candidates near a high-quality center have a high chance of improving the current front F. However, it is inappropriate to directly use TEHVI to distinguish the best hyperparameter setting that has been visited, as trajectories of the visited hyperparameter settings have already contributed to constructing the front. Instead, similar to Def 4, we evaluate the contribution of a visited hyperparameter setting $\boldsymbol{x}$ by the difference between the HV of the front F and it excluding the observed trajectory of $\boldsymbol{x}$. Therefore, the center is selected as,

$$\boldsymbol{x}^c = \underset{\boldsymbol{x} \in X}{\operatorname{argmax}} \ \big[HV\big(F \mid \boldsymbol{r}\big) - HV\big(F \setminus \{\boldsymbol{y}(\boldsymbol{x}, t)\}_{t=0}^{t'_{max}} \mid \boldsymbol{r}\big)\big], \tag{8}$$

where $\boldsymbol{r}$ denotes the reference point and $\{\boldsymbol{y}(\boldsymbol{x}, t)\}_{t=1}^{t'_{max}}$ denotes the observed trajectory when training the model with $\boldsymbol{x}$ up to t'_{max} epochs. Note that $t'_{max} \leq t_{max}$ due to the early stopping mechanism.

Candidate Generation. After selecting the center $\boldsymbol{x}^c$, we generate a candidate $\bar{\boldsymbol{x}}$ by adding Gaussian perturbation with zero mean and covariance matrix $\gamma^2(\boldsymbol{x}^c)I^d$ to the center, i.e., $\bar{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{x}^c, \gamma^2(\boldsymbol{x}^c)I^d)$, where $\gamma(\boldsymbol{x}^c)$ denote the search radius specified for $\boldsymbol{x}^c$ with γ_{max} being the initial value. Let $\bar{X}$ denote the set of q independently generated candidates. Moreover, to dynamically balance exploration and exploitation, if the candidate search led by center $\boldsymbol{x}^c$ finally fails to yield a new setting to improve the front F, we halve its search radius so as to prioritize the points closer to it, and if the search fails for multiple times, we exclude $\boldsymbol{x}^c$ as a center in the subsequent iterations. Finally, we calculate the TEHVI value for each candidate in $\bar{X}$ and select the one with the highest TEHVI value to train the ML model in the current iteration.

C EXPERIMENT SETUP

C.1 ALGORITHM CONFIGURATIONS

As discussed in Section 4, the kernel of GP models in TMOBO is implemented by the product of a Matérn kernel (with smoothness parameter $\nu = 2.5$) over the hyperparameter space and a temporal kernel over epochs. In real-world benchmarks, where some prior knowledge of the objective's characteristics is available, we use informed temporal kernels: an exponential decay kernel when the objective is validation loss and a linear kernel when the objective is training cost. However, in synthetic simulations, which are designed to reflect more general settings, we default to a Matérn temporal kernel with $\nu = 2.5$, providing a general-purpose modeling choice. The GP models are implemented using the GPyTorch library [Gardner et al., 2018] and fitted via standard maximum likelihood estimation with gradient-based optimization of kernel hyperparameters.

Furthermore, within the candidate search strategy of TMOBO, the initial search radius γ_{max} associated with any visited hyperparameter setting is set to 0.2 as recommended by [Regis and Shoemaker, 2007]. Moreover, to ensure a sufficiently dense neighborhood around the center, the number of candidates q generated around a center is set to $100d$. Then, in the early stopping mechanism, we determine the conservative stopping epoch in a way similar to computing the lower confidence bound and maintain a fixed value of β, which controls the confidence level, at 2.0. Finally, at the end of each iteration, TMOBO takes an active data augmentation method to minimize the prediction uncertainty of the GP model by using only a subset of trajectory observations. To maintain the GP training efficiency, we limit the size of this subset to a maximum of 10.

We use the open-source Python implementations for ParEGO, qEHVI, and qNEHVI from the BoTorch library (accessible at `https://github.com/pytorch/botorch` under MIT License) and adhere to the default algorithm configurations. The enhanced versions of the alternative algorithms, namely ParEGO-T, qEHVI-T, and qNEHVI-T, are implemented by collecting all the intermediate observations $\{\boldsymbol{y}(\boldsymbol{x}, 1), \ldots, \boldsymbol{y}(\boldsymbol{x}, t)\}$ to refine the current front whenever a query pair $(\boldsymbol{x}, t)$ is sampled.

In each experimental trial, we initialize each algorithm with $2(d + 1)$ samples drawn from a Sobol sequence. For the approximation of the acquisition function by Monte Carlo integration, we consistently employ 128 MC samples across all iterations. HV-based acquisition functions in qEHVI, qNEHVI, and TMOBO are inherently sensitive to the choice of the reference point $\boldsymbol{r}$. However, determining an appropriate reference point generally requires prior knowledge of the problem, which is unrealistic for real-world applications. Thus, we adopt an adaptive strategy for all algorithms where each element of the reference point is continuously updated to the corresponding worst values encountered thus far. All experiments are run on a GeForce RTX 2080 Ti GPU with 11GB RAM.

C.2 PROBLEMS AND BENCHMARKS

The classical multi-objective optimization benchmarks used to formulate the synthetic test problems in numerical experiments are provided below:

ZDT1 Benchmark [Zitzler et al., 2000]:

$$\begin{aligned} \min_{\boldsymbol{x}} \ & [f_1(\boldsymbol{x}), f_2(\boldsymbol{x})], \\ & f_1(\boldsymbol{x}) = x_1, \\ & f_2(\boldsymbol{x}) = u(\boldsymbol{x}) \left[1 - \sqrt{x_1/u(\boldsymbol{x})}\right], \end{aligned} \tag{9}$$

where

$$u(\boldsymbol{x}) = 1 + \frac{9}{d-1} \sum_{i=2}^{d} x_d,$$

and $\boldsymbol{x} = [x_1, \ldots, x_d] \in [0, 1]^d$. This benchmark has a convex Pareto-optimal front with Pareto-optimal solutions being $0 \leq x_1^* \leq 1$ and $x_i^* = 0$ for $i = 2, \ldots, d$.

ZDT2 Benchmark [Zitzler et al., 2000]:

$$\begin{aligned}\min_{\boldsymbol{x}} \ & [f_1(\boldsymbol{x}), f_2(\boldsymbol{x})], \\ & f_1(\boldsymbol{x}) = x_1, \\ & f_2(\boldsymbol{x}) = u(\boldsymbol{x}) \left[1 - (x_1/u(\boldsymbol{x}))^2\right],\end{aligned} \tag{10}$$

where

$$u(\boldsymbol{x}) = 1 + \frac{9}{d-1} \sum_{i=2}^{d} x_d,$$

and $\boldsymbol{x} = [x_1, \ldots, x_d] \in [0,1]^d$. This benchmark has a concave Pareto-optimal front with Pareto-optimal solutions being $0 \leq x_1^* \leq 1$ and $x_i^* = 0$ for $i = 2, \ldots, d$.

DTLZ1 Benchmark [Deb et al., 2002]:

$$\begin{aligned}\min_{\boldsymbol{x}} \ & [f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})], \\ & f_1(\boldsymbol{x}) = \frac{1}{2} x_1 x_2 \cdots x_{k-1}(1 + u(\boldsymbol{x}_k)), \\ & f_2(\boldsymbol{x}) = \frac{1}{2} x_1 x_2 \cdots (1 - x_{k-1})(1 + u(\boldsymbol{x}_k)), \\ & \qquad \vdots \\ & f_{k-1}(\boldsymbol{x}) = \frac{1}{2} x_1 (1 - x_2)(1 + u(\boldsymbol{x}_k)), \\ & f_k(\boldsymbol{x}) = \frac{1}{2}(1 - x_1)(1 + u(\boldsymbol{x}_k)),\end{aligned} \tag{11}$$

where

$$u(\boldsymbol{x}_k) = 100 \left[|\boldsymbol{x}_k| + \sum_{x_i \in \boldsymbol{x}_k} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)) \right],$$

$\boldsymbol{x} = [x_1, \ldots, x_d] \in [0,1]^d$ and $\boldsymbol{x}_k$ denotes the last $(d - k + 1)$ variables of $\boldsymbol{x}$. The search space of this benchmark contains multiple local Pareto-optimal fronts. The global Pareto-optimal front is a linear hyperplane with Pareto-optimal solutions being $x_i^* = 0.5$ for $x_i \in \boldsymbol{x}_k$.

DTLZ2 Benchmark [Deb et al., 2002]:

$$\begin{aligned}\min_{\boldsymbol{x}} \ & [f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})], \\ & f_1(\boldsymbol{x}) = \cos(\pi x_1/2)\cos(\pi x_2/2) \cdots \cos(\pi x_{k-1}/2)(1 + u(\boldsymbol{x}_k)), \\ & f_2(\boldsymbol{x}) = \cos(\pi x_1/2)\sin(\pi x_2/2) \cdots \cos(\pi x_{k-1}/2)(1 + u(\boldsymbol{x}_k)), \\ & \qquad \vdots \\ & f_{k-1}(\boldsymbol{x}) = \cos(\pi x_1/2)\sin(\pi x_2/2)(1 + u(\boldsymbol{x}_k)), \\ & f_k(\boldsymbol{x}) = \sin(\pi x_1/2)(1 + u(\boldsymbol{x}_k)),\end{aligned} \tag{12}$$

where

$$u(\boldsymbol{x}_k) = \sum_{x_i \in \boldsymbol{x}_k} (x_i - 0.5)^2,$$

$\boldsymbol{x} = [x_1, \ldots, x_d] \in [0,1]^d$ and $\boldsymbol{x}_k$ denotes the last $(d - k + 1)$ variables of $\boldsymbol{x}$. This benchmark has a concave Pareto-optimal front with Pareto-optimal solutions being $x_i^* = 0.5$ for $x_i \in \boldsymbol{x}_k$.

DTLZ7 Benchmark [Deb et al., 2002]:

$$
\begin{aligned}
\min_{\boldsymbol{x}} \ & [f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})], \\
& f_1(\boldsymbol{x}) = x_1, \\
& \quad \vdots \\
& f_{k-1}(\boldsymbol{x}) = x_{k-1}, \\
& f_k(\boldsymbol{x}) = h(f_1, f_2, \ldots, f_{k-1}, u)(1 + u(\boldsymbol{x}_k)),
\end{aligned} \tag{13}
$$

where

$$
\begin{aligned}
u(\boldsymbol{x}_k) &= 1 + \frac{9}{|\boldsymbol{x}_k|} \sum_{x_i \in \boldsymbol{x}_k} x_i, \\
h(f_1, f_2, \ldots, f_{k-1}, u) &= k - \sum_{i=1}^{k-1} \left[\frac{f_i}{1+u} (1 + \sin(3\pi f_i)) \right],
\end{aligned}
$$

$\boldsymbol{x} = [x_1, \ldots, x_d] \in [0,1]^d$ and $\boldsymbol{x}_k$ denotes the last $(d-k+1)$ variables of $\boldsymbol{x}$. The Pareto-optimal front of this benchmark is composed of 2^{k-1} disconnected regions with Pareto-optimal solutions being $x_i^* = 0$ for $x_i \in \boldsymbol{x}_k$.

Given the i-th objective function $f_i(\boldsymbol{x})$ of any benchmark above, we construct its epoch-dependent counterpart as $f_i(\boldsymbol{x}, t) = f_i(\boldsymbol{x}) \cdot g_i(t)$ to simulate the training procedure by the curve function $g_i(t)$. In our experiments, we utilize different curve functions to diversify the learning curve characteristics, and these functions are,

- Monotonically Increasing Curve: $g^M(t \mid t_{max}) = 0.5 + 1/\left(1 + e^{(-0.2(t - t_{max}/2))}\right)$,
- Monotonically Decreasing Curve: $g^{M'}(t \mid t_{max}) = 0.3 + 1/\left(1 + e^{(0.1(t - t_{max}/3))}\right)$,
- Quadratic Curve: $g^Q(t \mid t_{max}) = 0.5 + 2\left(t/t_{max} - 2/3\right)^2$,
- Periodic Curve: $g^P(t \mid t_{max}) = 1 + 0.5\sin(4\pi t/t_{max})$,

where $t \in \mathbb{T} = [1, 2, \ldots, t_{max}]$ and $t_{max} = 50$.

LCBench [Zimmer et al., 2021]: This benchmark is designed to give insights on multi-fidelity optimization with learning curves for Auto Deep Learning. LCBench was originally developed upon tabular data. Benefitting from the surrogate implementation by HPOBench (under Apache License 2.0) [Eggensperger et al., 2021] and YAHPO Gym (under Apache License 2.0) [Pfisterer et al., 2022], we are allowed to observe the intermediate model performance for any feasible hyperparameter setting after each epoch. The maximum number of epochs for training MLP is set to 50. We choose to minimize validation loss and training time, which are commonly considered in many MOHPO studies. For a demonstration, we focus on tuning the five hyperparameters (i.e., $d = 5$) of MLP including learning rate in $[1 \times 10^{-4}, 1 \times 10^{-1}]$, momentum in $[0.10, 0.99]$, weight decay in $[1 \times 10^{-5}, 1 \times 10^{-1}]$, maximum dropout rate in $[0.0, 1.0]$, and maximum number of neurons in $[64, 1024]$. LCBench utilizes diverse datasets from AutoML Benchmark [Gijsbers et al., 2019] hosted on OpenML [Vanschoren et al., 2014]. For experimentation, we select "blood-transfus", "higgs", "jungle-chess-2pcs", "kc1", and "banck-marketing", each of which has at least two attributes and between 500 and 1,000,000 data points.

CNN: This benchmark provides insights into hyperparameter optimization for modern computer vision applications. The convolutional neural network model MobileNetV2 is implemented in PyTorch and optimized for both validation accuracy and training efficiency. Following standard computer vision practices, we optimize four key hyperparameters (i.e., $d = 4$), including learning rate in $[1 \times 10^{-4}, 1 \times 10^{-1}]$, momentum in $[0.0, 0.99]$, weight decay in $[1 \times 10^{-5}, 1 \times 10^{-2}]$, and batch size in $[128, 512]$. The model is trained for up to 50 epochs on CIFAR-10 dataset, which consists of 60,000 32×32 color images across 10 classes. This configuration represents a more complex and computationally intensive optimization scenario compared to LCBench tasks, allowing us to assess our method's scalability to challenging real-world deep learning applications.

D SENSITIVITY ANALYSIS ON ALGORITHM CONFIGURATIONS

In this section, we conduct a series of sensitivity analyses to demonstrate how TMOBO's performance responds to its key algorithmic configurations. These configurations include the parameters for candidate search, Monte Carlo approximation, data augmentation strategy, and early stopping.

D.1 PARAMETERS FOR CANDIDATE SEARCH

Regarding the candidate search, we examine both the number of candidates and the search radius. Intuitively, a larger candidate set increases the likelihood of getting promising hyperparameter settings; however, it can impose unnecessary computational costs at the same time. As shown in the third row of Figure D.2, using $100d$ to $200d$ candidates generally leads to a lower (better) overall distribution in the boxplots (where d is the dimension of the domain of hyperparameter settings), especially when the objectives become more complex. Therefore, we choose a default setting of $100d$ candidates to balance effectiveness and efficiency. This choice is further supported by the statistical analysis in Table D.1, which indicates that our default setting is statistically comparable to or better than other settings, except in scenario Q-P where $200d$ outperforms.

For the search radius r, we can observe from the last row of Figure D.2 that a larger value of radius generally leads to a lower distribution in the boxplots; however, the marginal gains diminish gradually, and performance can even deteriorate beyond $r = 0.2$ in some cases. This behaviour is expected because the radius controls the balance between exploration and exploitation, and a larger radius biases the algorithm toward exploration by increasing the chance of selecting hyperparameter settings further from the center, which can slow convergence. Statistical results in Table D.1 indicate that $r = 0.2$ is significantly better than settings below 0.2 in half of the cases and comparable to higher settings across all cases. Consequently, we adopt it as our default.

D.2 PARAMETER FOR MONTE CARLO APPROXIMATION

Regarding the Monte Carlo approximation, we showcase the algorithm's performance with the number of MC samples M being 32, 64, 128, 256, and 512. It can be observed from the second row of Figure D.2 that the algorithm's performance is relatively insensitive to changes in M within this range. While increasing the number of MC samples can improve the integration of predictive uncertainty, this improvement does not necessarily translate into better overall algorithm performance, but it raises the computational burden due to the expensive TEHVI acquisition function. Considering the added computational cost associated with significantly larger values of M, we believe that 128 samples provide a good balance, capturing sufficient information for accurate TEHVI approximation without incurring excessive computational expense. Moreover, the statistical tests in Table D.1 indicate that the algorithm performance using default setting of 128 is comparable to both lower and higher sample sizes across all tested cases.

D.3 PARAMETER FOR DATA AUGMENTATION STRATEGY

We first compare data augmentation strategies by evaluating the impact of retaining 5, 10, or 15 observations per trajectory in the GP model. This analysis helps us assess whether a careful selection of augmented observations can manage GP complexity effectively while making reasonable prediction for optimization. Our findings in Figure D.2 indicate that the algorithm's performance is relatively insensitive to changes in this parameter across the four problems under study; in some cases, augmenting fewer data points even leads to a lower distribution in the boxplots. Considering the computational complexity of training a GP model, we maintain 10 observations per trajectory as our default setting, with the results in Table D.1 showing that this choice yields performance statistically comparable to other settings.

In parallel, we explore scalable GP models as an alternative to manual data augmentation. To this end, we implement a sparse GP with inducing points using the GPyTorch library [Gardner et al., 2018] and compare its performance against the data augmentation strategy. The Sparse GP (SGP) approach avoids the need for manual data curation and, as shown in Table D.1, typically achieves performance comparable to the 10-observation strategy in most cases. This indicates that both a careful selection of augmented observations and the use of sparse approximations are effective means of managing GP complexity in TMOBO. Moreover, as more algorithm iterations are executed, scalable GP models can serve as a suitable alternative.

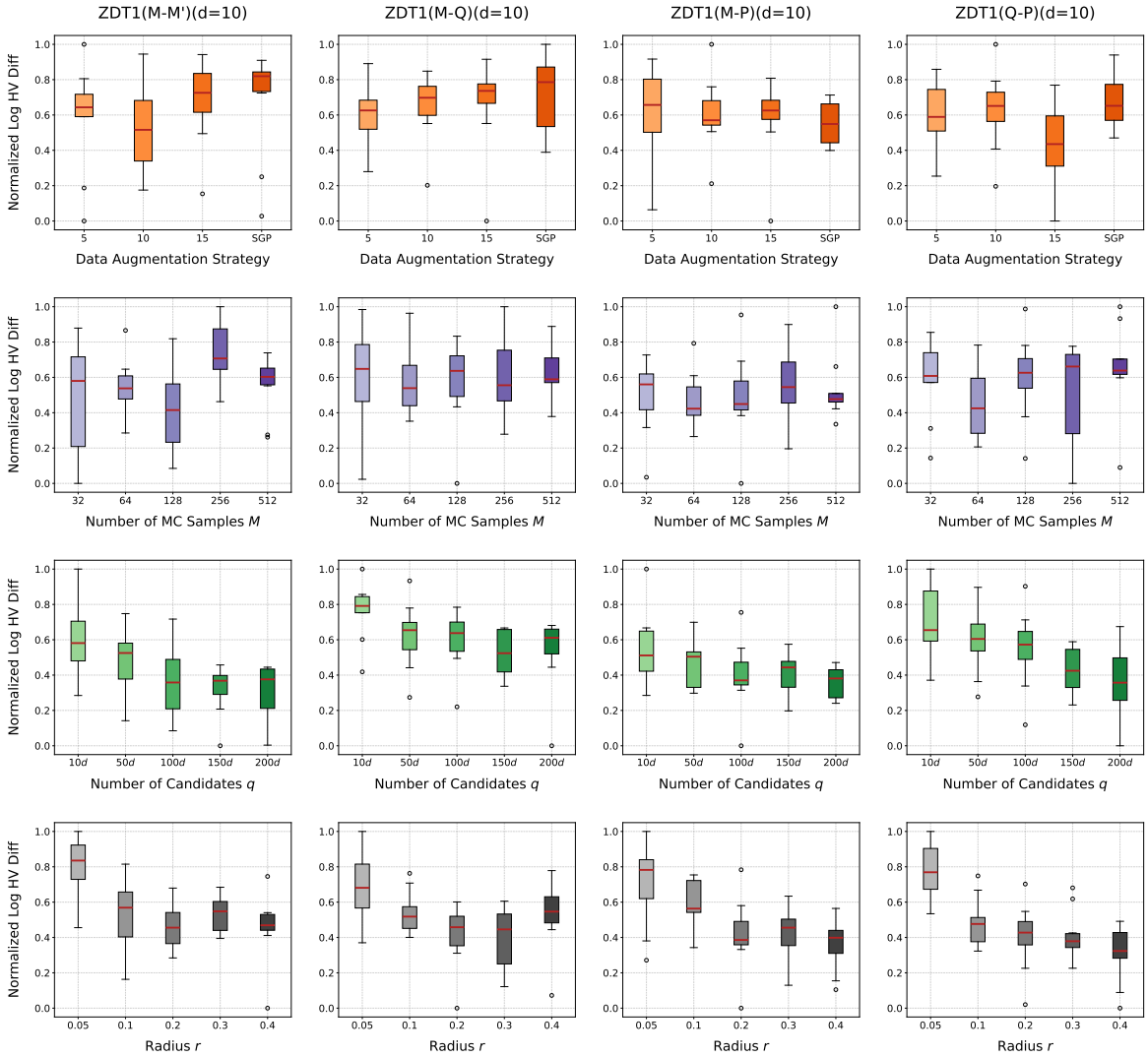

Figure D.2: Sensitivity analysis on key parameters and components of TMOBO on synthetic problems generated from ZDT1 after 100 algorithm iterations. Each row varies one factor, namely (1) data augmentation strategy, using GP trained with 5, 10, or 15 observations per trajectory, or using Sparse GP (SGP); (2) number of MC samples $M \in \{32, 64, 128, 256, 512\}$; (3) number of candidates $q \in \{10d, 50d, 100d, 150d, 200d\}$, where d is the problem dimension; and (4) search radius $r \in \{0.05, 0.1, 0.2, 0.3, 0.4\}$. Each box plot shows the logarithm of HV difference computed over 10 trials and is normalized to [0, 1]. (A lower value indicates better performance.)

Table D.1: Mean (standard deviation) of normalized log HV differences for key parameters and components of TMOBO on synthetic problems generated from ZDT1 after 100 algorithm iterations. For each parameter, performance is evaluated against the default algorithm setting (in bold) using Wilcoxon rank-sum test at a 95% confidence level. "+" sign indicates that the parameter option performs significantly better than the baseline, "−" indicates significantly worse performance, and "≈" denotes statistically comparable results.

		ZDT1(M-M')		ZDT1(M-Q)		ZDT1(M-P)		ZDT1(Q-P)	
		Mean (std)	Sign	Mean (std)	Sign	Mean (std)	Sign	Mean (std)	Sign
Data augmentation strategy	5	0.589 (0.277)	≈	0.603 (0.172)	≈	0.613 (0.244)	≈	0.587 (0.190)	≈
	10	0.528 (0.234)	≈	0.659 (0.179)	≈	0.602 (0.191)	≈	0.628 (0.206)	≈
	15	0.682 (0.219)	n.a.	0.666 (0.240)	n.a.	0.586 (0.214)	n.a.	0.430 (0.230)	n.a.
	SGP	0.686 (0.282)	≈	0.722 (0.203)	≈	0.554 (0.111)	≈	0.684 (0.157)	−
Number of MC samples M	32	0.493 (0.298)	≈	0.597 (0.289)	≈	0.496 (0.206)	≈	0.593 (0.220)	≈
	64	0.552 (0.147)	≈	0.587 (0.182)	≈	0.470 (0.153)	≈	0.460 (0.197)	≈
	128	0.417 (0.225)	n.a.	0.583 (0.231)	n.a.	0.486 (0.232)	n.a.	0.603 (0.217)	n.a.
	256	0.734 (0.155)	−	0.616 (0.212)	≈	0.542 (0.213)	≈	0.498 (0.278)	≈
	512	0.558 (0.154)	≈	0.631 (0.136)	≈	0.535 (0.183)	≈	0.651 (0.241)	≈
Number of candidates q	$10d$	0.600 (0.194)	−	0.761 (0.156)	−	0.545 (0.202)	≈	0.707 (0.194)	≈
	$50d$	0.475 (0.196)	≈	0.622 (0.167)	≈	0.464 (0.142)	≈	0.595 (0.171)	≈
	100d	0.366 (0.192)	n.a.	0.605 (0.157)	n.a.	0.395 (0.183)	n.a.	0.549 (0.201)	n.a.
	$150d$	0.329 (0.123)	≈	0.526 (0.129)	≈	0.406 (0.129)	≈	0.429 (0.122)	≈
	$200d$	0.314 (0.144)	≈	0.538 (0.194)	≈	0.356 (0.084)	≈	0.349 (0.212)	+
Search radius r	0.05	0.808 (0.162)	−	0.686 (0.181)	−	0.701 (0.225)	−	0.775 (0.152)	−
	0.1	0.529 (0.205)	≈	0.536 (0.119)	≈	0.593 (0.132)	−	0.484 (0.132)	≈
	0.2	0.463 (0.120)	n.a.	0.420 (0.167)	n.a.	0.412 (0.190)	n.a.	0.405 (0.175)	n.a.
	0.3	0.535 (0.097)	≈	0.392 (0.162)	≈	0.429 (0.135)	≈	0.415 (0.129)	≈
	0.4	0.458 (0.177)	≈	0.532 (0.183)	≈	0.363 (0.142)	≈	0.311 (0.152)	≈

D.4 PARAMETER FOR EARLY STOPPING

Regarding the early stopping mechanism, we present a separate sensitivity analysis in Figure D.3 and Table D.2, because the changes in β affect the number of training epochs via early stopping, and it is more meaningful to compare the results after a fixed number of training epochs. As shown in Figure D.3, a moderate value of β (between 0.1 and 0.3) generally results in a lower median in the boxplots; while $\beta = 0.05$ or 0.4 results in a higher median, especially in the first three cases. Intuitively, a lower β accounts for less predictive uncertainty and may cause premature termination of training, thereby encouraging exploration of more hyperparameter settings; conversely, a higher β is likely to prolong training along a given trajectory, leading to excessive exploitation and reduced exploration. This balance between exploitation and exploration is crucial for algorithm performance. The statistical results in Table D.2 show that algorithm performance using the default setting $\beta = 0.2$ is statistically comparable to the other settings. Based on these insights, we recommend choosing a moderate β value for a general TMOBO application.

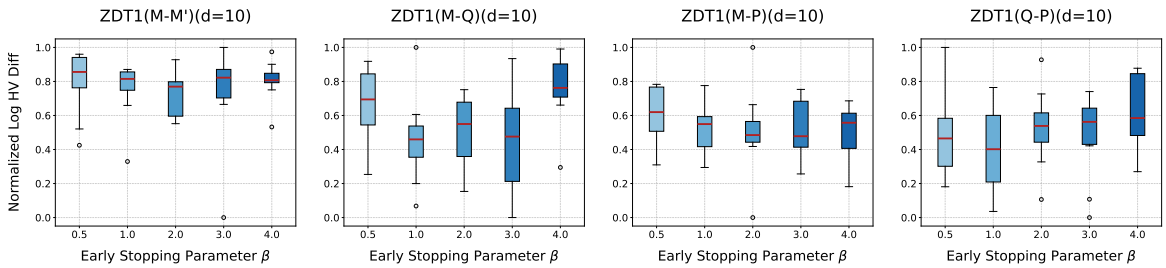

Figure D.3: Sensitivity analysis on early stopping parameter $\beta \in \{0.5, 1.0, 2.0, 3.0, 4.0\}$ of TMOBO on synthetic problems generated from ZDT1 after 3000 training epochs. Each box plot shows the logarithm of HV difference computed over 10 trials and is normalized to [0, 1]. (A lower value indicates better performance.)

Table D.2: Mean (standard deviation) of normalized log HV differences for early stopping parameter β of TMOBO on synthetic problems generated from ZDT1 after 3000 training epochs. For each parameter, performance is evaluated against the default algorithm setting (in bold) using Wilcoxon rank-sum test at a 95% confidence level. "+" sign indicates that the parameter option performs significantly better than the baseline, "−" indicates significantly worse performance, and "≈" denotes statistically comparable results.

		ZDT1(M-M')		ZDT1(M-Q)		ZDT1(M-P)		ZDT1(Q-P)	
		Mean (std)	Sign	Mean (std)	Sign	Mean (std)	Sign	Mean (std)	Sign
Early stopping (β)	0.5	0.802 (0.178)	≈	0.658 (0.225)	≈	0.606 (0.158)	≈	0.498 (0.268)	≈
	1.0	0.758 (0.156)	≈	0.460 (0.238)	≈	0.525 (0.138)	≈	0.404 (0.229)	≈
	2.0	0.735 (0.131)	n.a.	0.496 (0.208)	n.a.	0.502 (0.234)	n.a.	0.530 (0.211)	n.a.
	3.0	0.751 (0.272)	≈	0.459 (0.282)	≈	0.525 (0.161)	≈	0.481 (0.235)	≈
	4.0	0.805 (0.115)	≈	0.761 (0.198)	−	0.501 (0.159)	≈	0.618 (0.210)	≈

E NUMERICAL EXPERIMENTS AND RESULTS

E.1 ADDITIONAL RESULTS FOR SECTION 6.1

In this subsection, we evaluate TMOBO and alternative algorithms on 20 synthetic problems derived from standard multi-objective benchmarks, as discussed in Section 6.1. Figures E.4 and E.5 illustrate the average log HV difference of each algorithm against the number of iterations. Each row of subplots represents problems generated from the same benchmark but with varying trajectory complexities.

On the ZDT benchmarks (Figure E.4), TMOBO achieves rapid convergence, reaching a low HV difference value within a few iterations. It maintains this advantage throughout the optimization process and outperforms the alternatives. While qNEHVI-T and qEHVI-T demonstrate competitive performance on the first two instances of ZDT2, their ability to handle increasing trajectory complexity is limited. In such cases, TMOBO significantly dominates these methods.

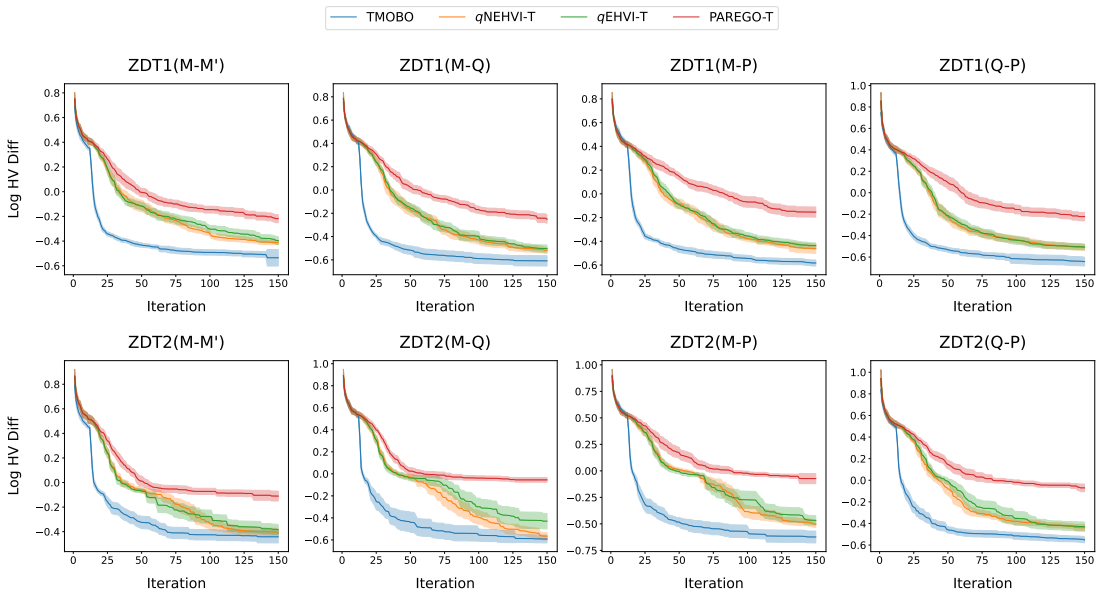

Figure E.4: Average log Hypervolume difference against iterations for each algorithm on synthetic problems generated from ZDT1 and ZDT2. Each algorithm runs for 20 independent trials. The shaded region indicates two standard errors of mean.

In contrast, the variants of the DTLZ benchmarks (Figure E.5) present greater challenges. On synthetic problems derived from DTLZ1, all algorithms struggle to converge adequately within 150 iterations, primarily due to the existence of multiple local optima. However, TMOBO achieves noticeably better HV difference values, which highlights its robustness in handling complex landscapes. DTLZ7 benchmark challenges the capability of optimizers to maintain a diverse set of solutions as it

has a disconnected Pareto-optimal front. Notably, TMOBO emerges as the best choice for handling this type of challenge, as the alternatives tend to stagnate at relatively high HV difference values at an early stage.

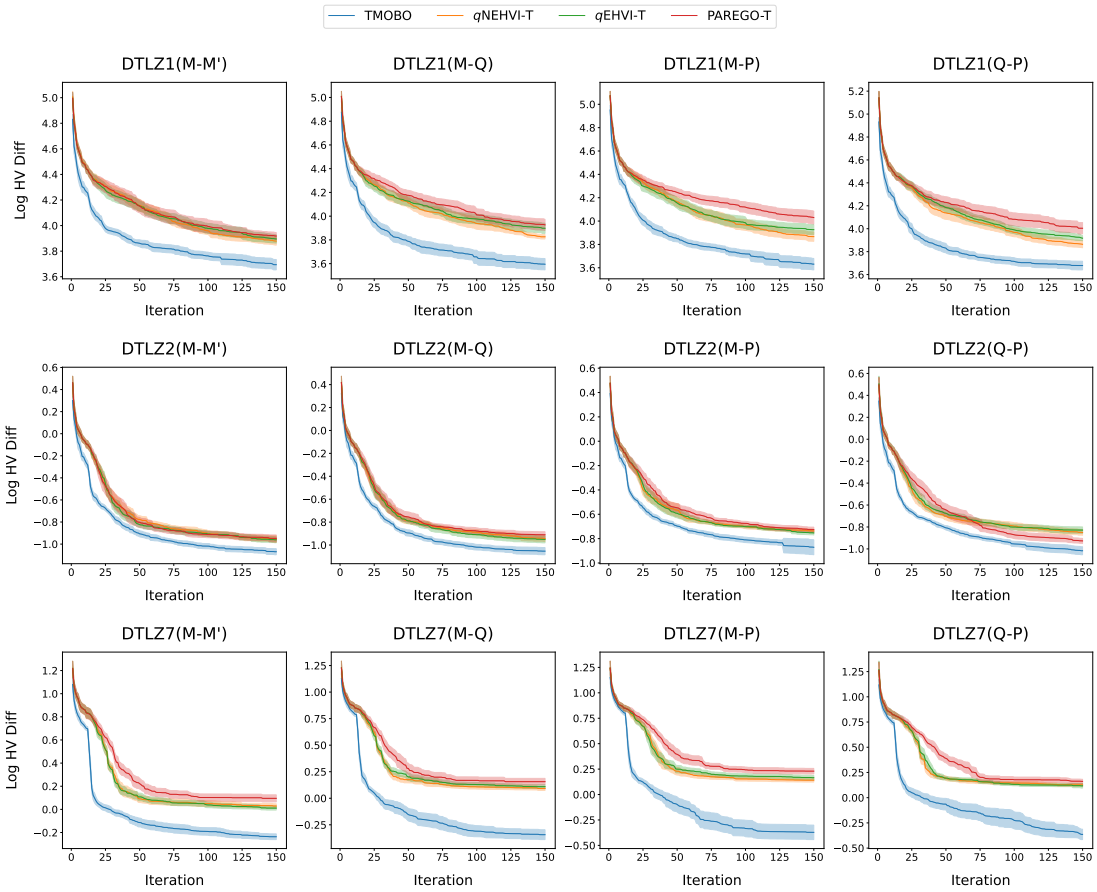

Figure E.5: Average log Hypervolume difference against iterations for each algorithm on synthetic problems generated from DTLZ1, DTLZ2, and DTLZ7. Each algorithm runs for 20 trials. The shaded region indicates two standard errors of mean.

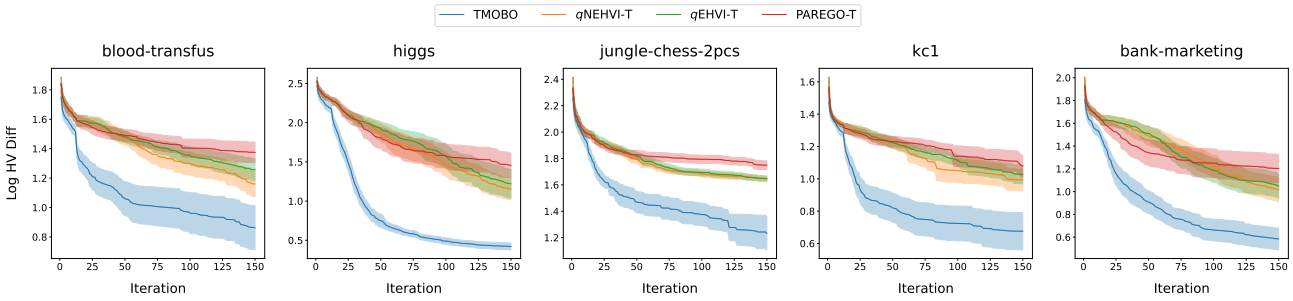

Figure E.6: Average log Hypervolume difference against iterations for each algorithm on five different hyperparameter tuning tasks. Each algorithm runs for 20 independent trials. The shaded region indicates two standard errors of the mean.

E.2 ADDITIONAL RESULTS FOR SECTION 6.2

Next, we assess TMOBO and alternative algorithms on five hyperparameter tuning tasks from LCBench, as described in Section 6.2. These tasks involve optimizing validation loss and training cost for an MLP model. Figure E.6 compares the algorithms in terms of average log HV difference against the number of iterations.

TMOBO consistently outperforms the alternatives across all tasks, achieving better-converged and more diversified Pareto-optimal fronts. Its trajectory-based acquisition function enables it to extract valuable insights from partially trained models. In contrast, qNEHVI-T and qEHVI-T fail to match TMOBO's performance in maintaining convergence speed and solution quality; and ParEGO-T performs the worst. Overall, these results highlight TMOBO's effectiveness in hyperparameter tuning scenarios, particularly in balancing computational efficiency and optimization performance.

E.3 IMPACT OF EARLY STOPPING

On the CNN benchmark, we further take this opportunity to provide readers with a clearer understanding of our algorithm's early stopping mechanism as well as the computational complexity. Figure E.7 first shows that TMOBO and its non-early-stopping variant, TMOBO-nES, achieve comparable performance throughout the trial, from initialization to the end of 100 algorithm iteration. Notably, TMOBO requires less overall runtime (including CNN training time and algorithm overhead) than TMOBO-nES (as reflected by the "Total" boxplots) primarily because its early stopping mechanism prevents unnecessary training of the CNN model, thereby reducing the number of epochs per trajectory and conserving the wall-clock time. This efficiency is evident in the lower "Train" boxplot of TMOBO. The trade-off is that the early stopping mechanism introduces additional computational overhead, since TMOBO needs to determine in real time when to terminate training. However, when compared with the overall time required to train the CNN model, the overhead for both TMOBO and TMOBO-nES is negligible.

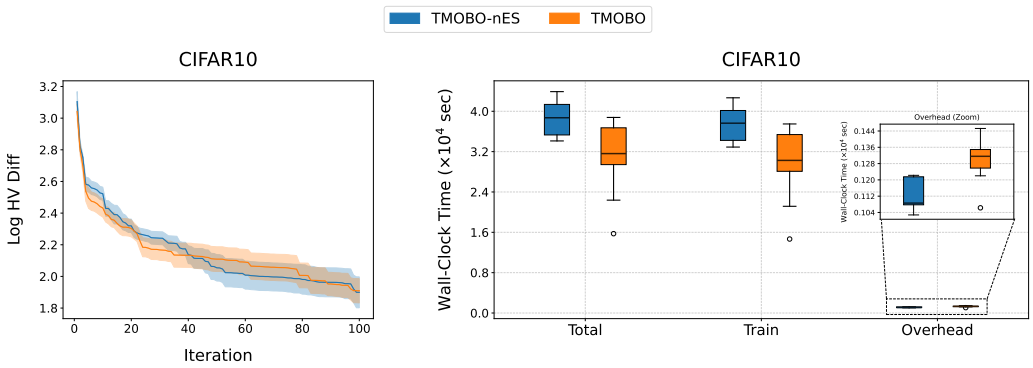

Figure E.7: Comparison of the computational efficiency of TMOBO and TMOBO-nES on the hyperparameter tuning task of MobileNetV2 on the CIFAR-10 dataset. [Left] Average log HV difference against algorithm iterations for each algorithm across 10 independent trials. The shaded region indicates two standard errors of the mean. [Right] Boxplots of wall-clock time for each algorithm, where "Total" indicates the overall runtime, "Train" the overall CNN training time, and "Overhead" the algorithm additional processing time.

E.4 IMPACT OF OBJECTIVE COMPLEXITY

In different ML applications, practitioners may consider optimizing hyperparameters for varying objectives. This makes it essential to evaluate TMOBO's performance across diverse scenarios. To explore this, we modified the LCBench hyperparameter tuning tasks by replacing the training cost objective with validation cross-entropy, while retaining validation accuracy as the other objective. This modification introduces a more complex, non-linear relationship between objectives.

Figure E.8 shows the performance of TMOBO and three alternative algorithms in terms of the average log HV difference against the total number of epochs. As the trajectory characteristics become more complex, each algorithm takes more effort to converge. However, TMOBO consistently outperforms the alternatives in both convergence speed and final HV difference, except for the second task on "higgs" where TMOBO and qNEHVI-T demonstrate comparable performance. Despite the changes in objective complexity, TMOBO maintains its advantage over the alternatives by leveraging information from partially trained models and dynamically adjusting training durations. This capability allows it to efficiently navigate the more challenging optimization tasks.

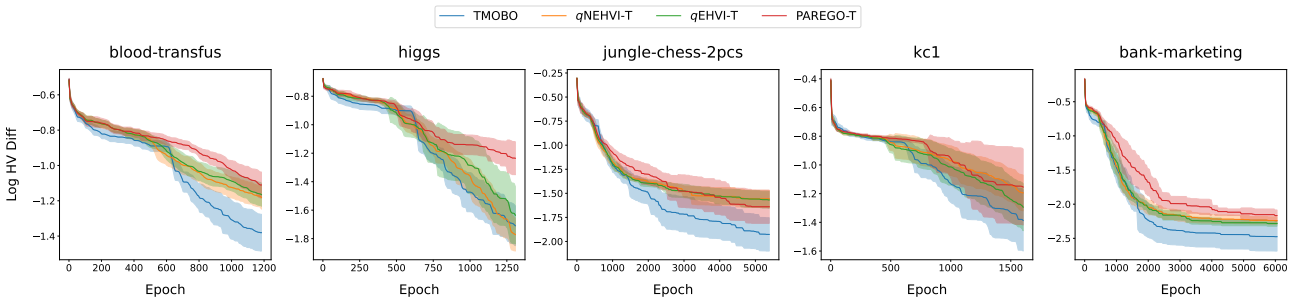

Figure E.8: Average log Hypervolume difference against the number of epochs for each algorithm on five different hyperparameter tuning tasks (with first objective being validation accuracy and the second objective validation cross entropy). Each algorithm runs for 10 independent trials. The shaded region indicates two standard errors of the mean.

E.5 IMPACT OF NOISY LEVELS

We then investigate the impact of stochasticity on the performance of TMOBO. To this end, for each synthetic problem generated from DTLZ, we add to each objective a Gaussian noise with standard deviations of 10%, 1%, and 0.1% of the objective range. Figure E.9 highlights TMOBO's adaptability and performance across different noise scenarios.

At lower noise levels (0.1% and 1%), TMOBO demonstrates stable and consistent performance. The algorithm quickly converges to a low HV difference, showcasing its ability to reliably predict trajectories and identify Pareto-optimal trade-offs. As the noise level increased to 10%, the optimization becomes more challenging due to the amplified variability in objective evaluations, which TMOBO inherently inherits as part of the trade-off modeling. Despite these challenges, TMOBO continues to converge and achieves reasonably low HV differences by the end of trials.

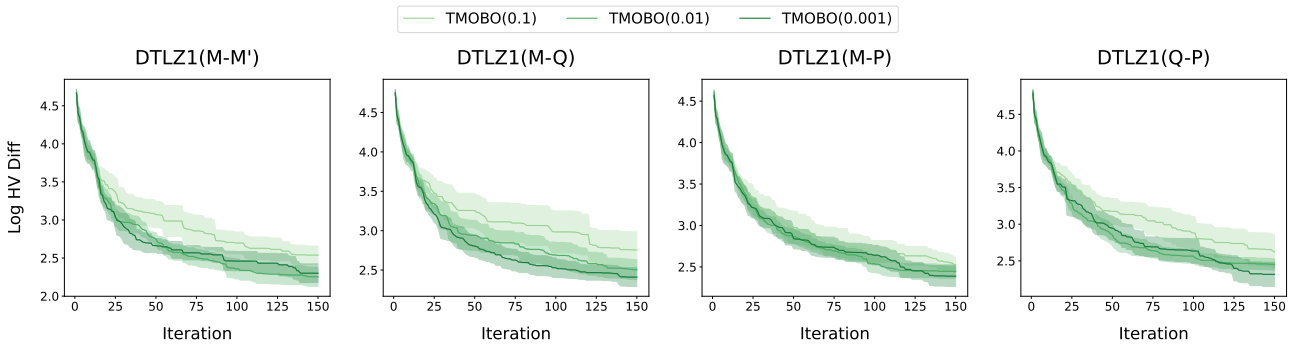

Figure E.9: Average log Hypervolume difference against iterations for TMOBO on synthetic problems generated from DTLZ1 with noise level 0.1, 0.01, and 0.001. For each noise level, TMOBO runs for 10 independent trials. The shaded region indicates two standard errors of the mean.

E.6 IMPACT OF REPLICATION STRATEGY

As a result of the analysis on noise levels, replicated observations at a query $(\boldsymbol{x}, t)$ might be needed to achieve adequate accuracy, especially in the presence of significant noises. While this particular scenario is not the focus of our current study, we outline a straightforward method to extend the proposed TMOBO to accommodate the requirement for replicated observations when the computational budget is sufficient. It is essential to recognize that with each noisy observation obtained at $(\boldsymbol{x}, t)$, the training process must start from scratch and will also output noisy observations from $(\boldsymbol{x}, 1)$ up to $(\boldsymbol{x}, t-1)$. As a consequence, careful maintenance of multiple training procedures for replications is needed to ensure algorithmic efficiency. The pseudo-code for TMOBO-P, a modified version of TMOBO, is presented in Algorithm E.1.

With an additional input P denoting the number of replications, TMOBO-P intends to include P replicated observations at any visited query pair to mitigate the influence of noises. In each iteration, different from TMOBO, TMOBO-P concurrently tracks P independent training procedures for the ML model with the same sampled hyperparameter setting $\boldsymbol{x}'$ and ensures

their progress remains consistent at the same epoch. Therefore, upon visiting a query pair $(\boldsymbol{x}, t')$, we can obtain P replicated observations of it. Intuitively, the collection of sample means $\{\sum_{p=1}^{P} \boldsymbol{y}_p(\boldsymbol{x}', 1)/P, \ldots, \sum_{p=1}^{P} \boldsymbol{y}_p(\boldsymbol{x}', t_{max})/P\}$ form a compressed trajectory of $\boldsymbol{x}'$, i.e., average of P observed trajectory of $\boldsymbol{x}'$. Then, the early stopping mechanism is applied to the compressed trajectory to determine when to stop all P training procedures at the same time. Since the replicated training procedures are independent, TMOBO-P can leverage parallel computing by assigning each training procedure to a specific processor.

Algorithm E.1 Framework of TMOBO-P

Input: Initial sets of inputs Z and observations Y, number of replications P, and initial Pareto-optimal front F identified from Y.

while the computational budget has not been exceeded **do**
  Fit k GP models with $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ based on sets Z and Y.
  Sample a new $\boldsymbol{x}'$ by maximizing the TEHVI acquisition function.
  Initialize $Z' \leftarrow \emptyset$ and $Y' \leftarrow \emptyset$.
  for $t' = 1$ **to** $t_{\max}$ **do**
    for $p = 1$ **to** P **do**
      Continue model training for the t'-th epoch to obtain observation $\boldsymbol{y}_p(\boldsymbol{x}', t')$ on the p-th processor.
    end for
    Let $Z' \leftarrow Z' \cup \{(\boldsymbol{x}', t')\}$ and $Y' \leftarrow Y' \cup \{\sum_{p=1}^{P} \boldsymbol{y}_p(\boldsymbol{x}', t')/P\}$ and update front F.
    Fit k GP models with $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ based on sets $Z \cup Z'$ and $Y \cup Y'$.
    if EarlyStopping$(\boldsymbol{x}', t', \boldsymbol{\mu}, \boldsymbol{\Sigma}, F)$ is triggered **then**
      Break;
    end if
  end for
  Augment Z' and Y' into Z and Y respectively.
end while

Figure E.10 shows the performance of TMOBO-P with $P = 1$, 4, 16, and 64 on the synthetic problems derived from ZDT1. TMOBO proposed in the main paper can be considered a special case of TMOBO-P with $P = 1$. This time, we add Gaussian noise to each objective with a standard deviation of 10% of the objective range in order to emphasize the influence of noises. It can be observed that the performance of TMOBO-P improves significantly in terms of the HV difference as more replications are allowed. This improvement is evident not only in rapid convergence during the initial stages but also in obtaining high-quality results at the end when comparing TMOBO with $P \geq 16$ to TMOBO with $P < 16$. In the meantime, benefiting from a large number of replications, TMOBO-64 has stable performance and its standard error is relatively small.

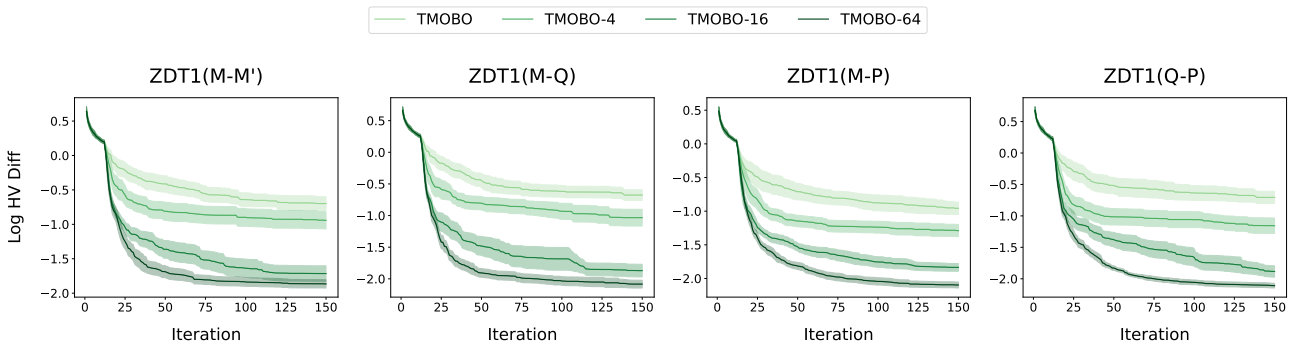

Figure E.10: Average log Hypervolume difference against iterations for each algorithm on synthetic problems generated from ZDT1 with 10% noise level. Each algorithm runs for 20 independent trials. The shaded region indicates two standard errors of the mean.