Speeding Up Multi-Objective Hyperparameter Optimization by Task Similarity-Based Meta-Learning for the Tree-Structured Parzen Estimator

Shuhei Watanabe [*1], **Noor Awad** [1], **Masaki Onishi** [2] and **Frank Hutter**[1]

[1] Department of Computer Science, University of Freiburg, Germany
[2] Artificial Intelligence Research Center, AIST, Tokyo, Japan

{watanabs,awad,fh}@cs.uni-freiburg.de, onishi-masaki@aist.go.jp

Abstract

Hyperparameter optimization (HPO) is a vital step in improving performance in deep learning (DL). Practitioners are often faced with the trade-off between multiple criteria, such as accuracy and latency. Given the high computational needs of DL and the growing demand for efficient HPO, the acceleration of multi-objective (MO) optimization becomes ever more important. Despite the significant body of work on meta-learning for HPO, existing methods are inapplicable to MO tree-structured Parzen estimator (MO-TPE), a simple yet powerful MO-HPO algorithm. In this paper, we extend TPE's acquisition function to the meta-learning setting using a task similarity defined by the overlap of top domains between tasks. We also theoretically analyze and address the limitations of our task similarity. In the experiments, we demonstrate that our method speeds up MO-TPE on tabular HPO benchmarks and attains state-of-the-art performance. Our method was also validated externally by winning the *AutoML 2022 competition on "Multiobjective Hyperparameter Optimization for Transformers"*. See https://arxiv.org/abs/2212.06751 for the latest version with Appendix.

1 Introduction

Hyperparameter optimization (HPO) is a critical step in achieving strong performance in deep learning [Chen *et al.*, 2018; Henderson *et al.*, 2018]. Additionally, practitioners are often faced with the trade-off between important metrics, such as accuracy, latency of inference, memory usage, and algorithmic fairness [Schmucker *et al.*, 2020; Candelieri *et al.*, 2022]. However, exploring the Pareto front of multiple objectives is more complex than single-objective optimization, making it particularly important to accelerate multi-objective (MO) optimization.

To accelerate HPO, a large body of work on meta-learning has been actively conducted, as surveyed, e.g., by Vanschoren [2019]. In the context of HPO, meta-learning mainly focuses on the knowledge transfer of metadata in Bayesian optimization (BO) [Swersky *et al.*, 2013; Wistuba *et al.*, 2016; Feurer *et al.*, 2018; Perrone *et al.*, 2018; Salinas *et al.*, 2020; Volpp *et al.*, 2020]. These methods use meta information in Gaussian process (GP) regression to yield more informed surrogates or an improved acquisition function (AF) for the target dataset, making them applicable to existing MO-BO methods, such as ParEGO [Knowles, 2006] and SMS-EGO [Ponweiser *et al.*, 2008]. However, recent works reported that a variant of BO called MO tree-structured Parzen estimator (MO-TPE) [Ozaki *et al.*, 2020; Ozaki *et al.*, 2022] is more effective than the aforementioned GP-based methods in expensive MO settings. Since MO-TPE uses kernel density estimators (KDEs) instead of GPs, existing meta-learning methods are not directly applicable, and a meta-learning procedure for TPE is yet to be explored.

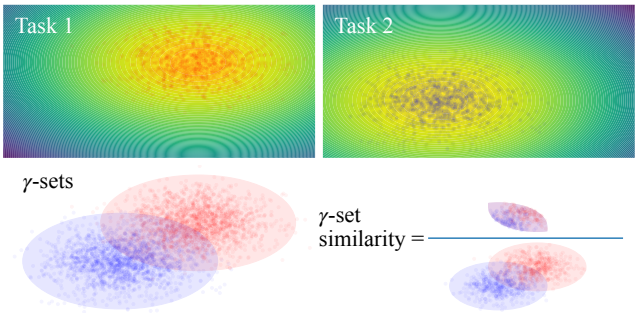

Figure 1: The conceptual visualization of γ-*set similarity* measure. **Top row**: the γ-sets of each task. The dots show the top-γ-quantile observations in both tasks. **Bottom row**: the γ-set similarity is measured via intersection over union of the top-γ-quantile domain, which we define γ-*set*; see Definition 2 in Appendix B.2.

To address this issue, we propose a meta-learning method for TPE on non-hierarchical spaces, i.e. search space does not include any conditional parameters, using a new task kernel. Our method models the joint probability density function (PDF) of an HP configuration $\boldsymbol{x}$ and a task t using a new task kernel $k_t(t_i, t_j)$. We calculate the task kernel by using the intersection-over-union-based new similarity measure, which we call γ-*set similarity*, as visualized in Figure 1. Note that we describe the theoretical details in Appendix A. Although this task kernel successfully works well in many cases, its performance is degraded under some circumstances, such as for high-dimensional spaces or when transferring knowledge

[*]The work was partially done in AIST.

from slightly dissimilar tasks. To alleviate this performance degradation, we analytically discuss and address the issues in this task kernel by (1) dimension reduction based on HP importance (HPI) and (2) an ϵ-greedy algorithm to determine the next HP configuration.

In our experiments, we demonstrate that our method successfully speeds up MO-TPE (or at least recovers the performance of MO-TPE when meta-tasks are not similar). The effectiveness of our method was also validated externally by winning the *AutoML 2022 competition on "Multiobjective Hyperparameter Optimization for Transformers"*. Note that this paper serves as the public announcement of the winner solution as well.

In summary, the main contributions of this paper are to:

1. extend TPE acquisition function (AF) to the meta-learning setting using a new task kernel,
2. discuss the drawbacks of the task kernel and provide the solutions to them, and
3. validate the performance of our method on real-world tabular benchmarks next to the external competition.

To facilitate reproducibility, our source code is available at https://github.com/nabenabe0928/meta-learn-tpe.

2 Related Work

In the context of BO, MO optimization is handled either by reducing MO to a single-objective problem (scalarization) or employing an AF that measures utility of a new configuration in the objective space. ParEGO [Knowles, 2006] is an example of scalarization that enjoys a convergence guarantee to the Pareto front. SMS-EGO [Ponweiser *et al.*, 2008] uses a lower-confidence bound of each objective to calculate hypervolume (HV) improvement, and EHVI [Emmerich *et al.*, 2011] uses expected HV improvement. PESMO [Hernández-Lobato *et al.*, 2016] and MESMO [Wang and Jegelka, 2017] are the extensions for MO settings of predictive entropy search and max-value entropy search. While those methods rely on GP, MO-TPE uses KDE and was shown to outperform the aforementioned methods in expensive MO-HPO settings [Ozaki *et al.*, 2020; Ozaki *et al.*, 2022].

The evolutionary algorithm (EA) community also studies MO actively. MOEAs use either surrogate-assisted EAs (SAEA) [Chugh *et al.*, 2016; Guo *et al.*, 2018; Pan *et al.*, 2018] or non-SAEA methods. Non-SAEA methods, such as NSGA-II [Deb *et al.*, 2002] and MOEA/D [Zhang and Li, 2007], typically require thousands of evaluations to converge [Ozaki *et al.*, 2020], and thus SAEAs are currently more dominant in the EA domain. Since SAEAs combine an EA with a cheap-to-evaluate surrogate, SAEAs are essentially similar to BO, as they can be seen as using EAs to optimize a particular AF in BO.

Meta-learning [Vanschoren, 2019] is a popular method to accelerate optimization and most of them can be classified into either of the following five types in the context of HPO:

1. initialization (or warm-starting) using promising configurations in meta-tasks [Feurer *et al.*, 2015; Nomura *et al.*, 2021],
2. search space reduction [Wistuba *et al.*, 2015; Perrone *et al.*, 2019],
3. learning an AF [Volpp *et al.*, 2020],
4. linear combination of models trained on each task [Wistuba *et al.*, 2016; Feurer *et al.*, 2018], and
5. training of a model jointly with meta-tasks [Swersky *et al.*, 2013; Springenberg *et al.*, 2016; Perrone *et al.*, 2018; Salinas *et al.*, 2020].

Warm-starting helps especially at the early stage of optimizations but does not use knowledge from the metadata afterward. Search space reduction could be applied to any method, but cannot identify the best configurations if the target task's optimum is outside of the optima for the meta-train tasks. The learning of AFs applies an expensive reinforcement learning step and is specific to GP-based methods. The linear combination is empirically demonstrated to outperform most meta-learning BO methods [Feurer *et al.*, 2018] including the search space reduction. The joint model trains a model on both observations and metadata. Although the linear combination of models (Type 4) is simple yet empirically strong, no meta-learning scheme for TPE has been developed so far. For this reason, we introduce a meta-learning method via a joint model (Type 5) inspired by Type 4.

3 Background

3.1 Bayesian Optimization (BO)

Suppose we would like to **minimize** a loss metric $f(\boldsymbol{x})$, then HPO can be formalized as follows:

$$\boldsymbol{x}_{\text{opt}} \in \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmin}}\, f(\boldsymbol{x}) \tag{1}$$

where $\mathcal{X} \coloneqq \mathcal{X}_1 \times \dots \mathcal{X}_D \subseteq \mathbb{R}^D$ is the search space and $\mathcal{X}_d \subseteq \mathbb{R}$ $(d = 1, \dots, D)$ is the domain of the d-th HP. In Bayesian optimization (BO) [Brochu *et al.*, 2010; Shahriari *et al.*, 2016; Garnett, 2022], we assume that $f(\boldsymbol{x})$ is expensive, and we consider the optimization in a surrogate space given observations $\mathcal{D}$. First, we build a predictive model $p(f|\boldsymbol{x}, \mathcal{D})$. Then, the optimization in each iteration is replaced with the optimization of the so-called AF. A common choice for the AF is the following expected improvement [Jones *et al.*, 1998]:

$$\text{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}] = \int_{-\infty}^{f^\star} (f^\star - f) p(f|\boldsymbol{x}, \mathcal{D}) df. \tag{2}$$

Another common choice is the following probability of improvement (PI) [Kushner, 1964]:

$$\mathbb{P}[f \leq f^\star|\boldsymbol{x}, \mathcal{D}] = \int_{-\infty}^{f^\star} p(f|\boldsymbol{x}, \mathcal{D}) df. \tag{3}$$

3.2 Tree-Structured Parzen Estimator (TPE)

TPE [Bergstra *et al.*, 2011; Bergstra *et al.*, 2013] is a variant of BO methods and it uses the expected improvement. See Watanabe [2023b] to better understand the algorithm components. To transform Eq. (2), we define the following:

$$p(\boldsymbol{x}|f, \mathcal{D}) \coloneqq \begin{cases} p(\boldsymbol{x}|\mathcal{D}^{(l)}) & (f \leq f^\gamma) \\ p(\boldsymbol{x}|\mathcal{D}^{(g)}) & (f > f^\gamma) \end{cases} \tag{4}$$

where $\mathcal{D}^{(l)}, \mathcal{D}^{(g)}$ are the observations with $f(\boldsymbol{x}_n) \leq f^\gamma$ and $f(\boldsymbol{x}_n) > f^\gamma$, respectively. f^γ is determined such

that f^γ is the $\lceil \gamma|\mathcal{D}| \rceil$-th best observation in $\mathcal{D}$. Note that $p(\boldsymbol{x}|\mathcal{D}^{(l)}), p(\boldsymbol{x}|\mathcal{D}^{(g)})$ are built by KDE. Combining Eqs. (2), (4) and Bayes' theorem, the AF of TPE is computed as [Bergstra *et al.*, 2011]:

$$\mathrm{EI}_{f^\gamma}[\boldsymbol{x}|\mathcal{D}] \overset{\mathrm{rank}}{\simeq} \frac{p(\boldsymbol{x}|\mathcal{D}^{(l)})}{p(\boldsymbol{x}|\mathcal{D}^{(g)})}. \tag{5}$$

Note that $\phi(\boldsymbol{x}) \overset{\mathrm{rank}}{\simeq} \psi(\boldsymbol{x})$ implies the order isomorphic and $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \phi(\boldsymbol{x}) \leq \phi(\boldsymbol{x}') \Leftrightarrow \psi(\boldsymbol{x}) \leq \psi(\boldsymbol{x}')$ holds. In each iteration, TPE samples configurations from $p(\boldsymbol{x}|\mathcal{D}^{(l)})$ and takes the configuration that satisfies the maximum density ratio among the samples. Note that although our task kernel cannot be computed for tree-structured search space, a.k.a. non-hierarchical search space, we use the name tree-structured Parzen estimator because this name is already recognized as a BO method using the density ratio of KDEs.

3.3 Multi-Objective TPE (MO-TPE)

MO-TPE [Ozaki *et al.*, 2020; Ozaki *et al.*, 2022] is a generalization of TPE with MO settings which falls back to the original TPE in case of single-objective settings. MO-TPE also uses the density ratio $p(\boldsymbol{x}|\mathcal{D}^{(l)})/p(\boldsymbol{x}|\mathcal{D}^{(g)})$ and picks the configuration with the best AF value at each iteration. The only difference from the original TPE is the split algorithm of $\mathcal{D}$ into $\mathcal{D}^{(l)}$ and $\mathcal{D}^{(g)}$. MO-TPE uses the HV subset selection problem (HSSP) [Bader and Zitzler, 2011] to obtain $\mathcal{D}^{(l)}$. HSSP tie-breaks configurations with the same non-domination rank based on the HV contribution. MO-TPE is reduced to the original TPE when we apply it to a single objective problem. In this paper, we replace HSSP with a simple tie-breaking method based on the crowding distance [Deb *et al.*, 2002] as this method does not require HV calculation, which can be highly expensive.

4 Meta-Learning for TPE

In this section, we briefly explain the TPE formulation and then describe the formulation of the AF for the meta-learning setting. Note that our method can be easily extended to MO settings using a rank metric $R : \mathbb{R}^m \to \mathbb{R}$ of an objective vector $\boldsymbol{f} \in \mathbb{R}^m$, and thus we discuss our formulation for the single-objective setting for simplicity; see Appendix B.3 for the theoretical discussion of the extension to MO settings.

Throughout this paper, we denote metadata as $\boldsymbol{D} \coloneqq \{\mathcal{D}_m\}_{m=1}^T$, where $T \in \mathbb{N}$ is the number of tasks and $\mathcal{D}_m$ is the set of observations on the m-th task with size $N_m \coloneqq |\mathcal{D}_m|$. We use the notion of the γ-set, which is, roughly speaking, a set of top-γ quantile configurations as visualized in Figure 1; for more theoretical details, see Appendix B.2. Furthermore, we define $\mathcal{X}_m^\gamma$ as the γ-set of the m-th task. For example, the red regions and the blue regions in Figure 1 correspond to $\mathcal{X}_1^\gamma$ and $\mathcal{X}_2^\gamma$.

4.1 Task-Conditioned Acquisition Function

TPE [Bergstra *et al.*, 2011] first splits a set of observations $\mathcal{D} = \{(\boldsymbol{x}_n, f(\boldsymbol{x}_n))\}_{n=1}^N$ into $\mathcal{D}^{(l)}$ and $\mathcal{D}^{(g)}$ at the top-γ quantile. Then we build KDEs $p(\boldsymbol{x}|\mathcal{D}^{(l)})$ and $p(\boldsymbol{x}|\mathcal{D}^{(g)})$, and compute the AF via $p(\boldsymbol{x}|\mathcal{D}^{(l)})/p(\boldsymbol{x}|\mathcal{D}^{(g)})$. The following proposition provides the multi-task version of the AF:

Proposition 1 *Under the assumption of the conditional shift, the task-conditioned* AF *is computed as*:

$$\mathrm{EI}_{f^\gamma}[\boldsymbol{x}|t, \boldsymbol{D}] \overset{\mathrm{rank}}{\simeq} \frac{p(\boldsymbol{x}, t|\boldsymbol{D}^{(l)})}{p(\boldsymbol{x}, t|\boldsymbol{D}^{(g)})}. \tag{6}$$

The conditional shift means that $p(\boldsymbol{x}|y, t_i) = p(\boldsymbol{x}|y, t_j)$ holds for different tasks, i.e. $t_i \neq t_j$ and it holds in our formulation due to the classification nature of the TPE model. We discuss more details in Appendix A.1. This formulation transfers the knowledge of top domains and weights the knowledge from similar tasks more. To compute the AF, we need to model the joint PDFs $p(\boldsymbol{x}, t|\mathcal{D}^{(l)}), p(\boldsymbol{x}, t|\mathcal{D}^{(g)})$, which we thus discuss in the next section.

4.2 Task Kernel

To compute the task kernel $k_t(t_i, t_j)$, the γ-set similarity visualized in Figure 1 (see Appendix B.2 for the formal definition) is employed. From Theorem 2 in Appendix A.2,

$$\hat{s}(\mathcal{D}_i^{(l)}, \mathcal{D}_j^{(l)}) \coloneqq \frac{1 - d_{\mathrm{tv}}(p_i, p_j)}{1 + d_{\mathrm{tv}}(p_i, p_j)} \tag{7}$$

almost surely converges to the γ-set similarity $s(\mathcal{X}_i^\gamma, \mathcal{X}_j^\gamma)$ if we can guarantee the strong consistency of $p(\boldsymbol{x}|\mathcal{D}_m^{(l)})$ for all $m = 1, \ldots, T$ where we define $p_m \coloneqq p(\boldsymbol{x}|\mathcal{D}_m^{(l)})$, t_m as a meta-task for $m = 2, \ldots, T$ and t_1 as the target task,

$$d_{\mathrm{tv}}(p_i, p_j) \coloneqq \frac{1}{2} \int_{\boldsymbol{x} \in \mathcal{X}} |p(\boldsymbol{x}|\mathcal{D}_i^{(l)}) - p(\boldsymbol{x}|\mathcal{D}_j^{(l)})| d\boldsymbol{x} \tag{8}$$

is the total variation distance, and $p(\boldsymbol{x}|\mathcal{D}_i^{(l)})$ is estimated by KDE. Note that $d_{\mathrm{tv}}(p_i, p_j)$ is approximated simply via Monte-Carlo sampling. In short, we need to compute:

1. KDEs of the top-γ-quantile observations in $\mathcal{D}_m$, and
2. d_{tv} between the target task and each meta-task.

Then we define the task kernel as follows:

$$k_t(t_i, t_j) = \begin{cases} \frac{1}{T} \hat{s}(\mathcal{D}_i^{(l)}, \mathcal{D}_j^{(l)}) & (i \neq j) \\ 1 - \frac{1}{T} \sum_{k \neq i} \hat{s}(\mathcal{D}_i^{(l)}, \mathcal{D}_k^{(l)}) & (i = j) \end{cases}. \tag{9}$$

Note that our task kernel is not strictly a kernel function as our task kernel does not satisfy semi-positive definite although it is still symmetric. The kernel is defined so that the summation over all tasks is 1, and then KDEs are built as follows:

$$\begin{aligned} p(\boldsymbol{x}, t|\boldsymbol{D}') &= \frac{1}{N'_{\mathrm{all}}} \sum_{m=1}^T k_t(t, t_m) \sum_{n=1}^{N'_m} k_x(\boldsymbol{x}, \boldsymbol{x}_{m,n}) \\ &= \frac{1}{N'_{\mathrm{all}}} \sum_{m=1}^T N'_m k_t(t, t_m) p(\boldsymbol{x}|\mathcal{D}'_m), \end{aligned} \tag{10}$$

where $\boldsymbol{D}' \coloneqq \{\mathcal{D}'_m\}_{m=1}^T$ is a set of subsets of the observations on the m-th task $\mathcal{D}'_m = \{(\boldsymbol{x}_{m,n}, f_m(\boldsymbol{x}_{m,n}))\}_{n=i}^{i+N'_m-1}$, and $N'_{\mathrm{all}} = \sum_{m=1}^T N'_m$. In principle, $\mathcal{D}'_m$ could be either $\mathcal{D}_m^{(l)}$ or $\mathcal{D}_m^{(g)}$. The advantages of this formulation are to (1) not be affected by the information from another task t_m if the task is *dissimilar* from the target task t_1, i.e. $\hat{s}(t_1, t_m) = 0$, and (2) asymptotically converge to the original formulation as the sample size goes to infinity, i.e. $\lim_{N'_1 \to \infty} p(\boldsymbol{x}, t|\boldsymbol{D}') = p(\boldsymbol{x}|\mathcal{D}'_1)$.

Algorithm 1 Task kernel (after the modifications)

η (controls the dimension reduction amount), S (Sample size of Monte-Carlo sampling)
$l_m(\boldsymbol{x}) \coloneqq p(\boldsymbol{x}|\mathcal{D}_m^{(l)})$ for $m = 1, \dots, T$
for $d = 1, \dots, D$ **do** ▷ Dimension reduction
  Calculate the average HPI $\bar{\mathbb{V}}_d$ based on Eq. (11)
▷ Pick dimensions from higher $\bar{\mathbb{V}}_d$
Build $\mathcal{S}$ with the top-$\lfloor \log_\eta |\mathcal{D}_1^{(l)}| \rfloor$ dimensions
Re-build $p_m^{\mathrm{DR}}(\boldsymbol{x}|\mathcal{D}_m^{(l)})$ based on Eq. (13)
for $m = 2, \dots, T$ **do**
  ▷ Use S samples for Monte-Carlo sampling
  Calculate d_{tv} in Eq. (8) with $l_1^{\mathrm{DR}}, l_m^{\mathrm{DR}}$
  Calculate $k_t(t_1, t_m)$ based on Eq. (9)
return k_t

4.3 When Does Our Meta-Learning Fail?

In this section, we discuss the drawbacks of our meta-learning method and provide solutions for them.

Case I: γ-Set for Target Task $\mathcal{D}_1^{(l)}$ Does Not Approach $\mathcal{X}^\gamma$

From the assumption of Theorem 2, $\mathcal{D}_1^{(l)}$ must approach $\mathcal{X}^\gamma$ to approximate the γ-set similarity precisely; however, since TPE is not a uniform sampler and it is a local search method due to the fact that the AF of TPE is PI [Watanabe and Hutter, 2022; Watanabe and Hutter, 2023; Song *et al.*, 2022], it does not guarantee that $\mathcal{D}_1^{(l)}$ goes to $\mathcal{X}^\gamma$ and it may even be guided towards non-γ-set domains. In this case, our task similarity measure not only obtains a wrong approximation, but also causes poor solutions. To avoid this problem, we introduce the ϵ-greedy algorithm to pick the next configuration instead of the greedy algorithm. By introducing the ϵ-greedy algorithm, we obtain the following theorem, and thus we can guarantee more correct or tighter similarity approximation:

Theorem 1 *If we use the ϵ-greedy policy ($\epsilon \in (0, 1)$) for* TPE *to choose the next candidate $\boldsymbol{x}$ for a noise-free objective function $f(\boldsymbol{x})$ defined on search space $\mathcal{X}$ with at most a countable number of configurations, and we use a* KDE *whose distribution converges to the empirical distribution as the number of samples goes to infinity, then a set of the top-γ-quantile observations $\mathcal{D}_1^{(l)}$ is almost surely a subset of $\mathcal{X}^\gamma$.*

The proof is provided in Appendix B.5. Intuitively speaking, this theorem states that when we use the ϵ-greedy algorithm, $\mathcal{D}_1^{(l)}$ will not include any configurations worse than the top-γ quantile if we have a sufficiently large number of observations. Therefore, the task similarity is correctly or pessimistically estimated. Notice that we use the bandwidth selection used by Falkner *et al.* [2018], which satisfies the assumption about the KDE in Theorem 1.

Case II: Search Space Dimension D Is High

When the dimensionality D is high, the approximation of the γ-set similarity is easily biased. In Figure 3, we provide a concrete example, where we consider $f(\boldsymbol{x}) = \|R\boldsymbol{x}\|_1$. Note that $\boldsymbol{x} \in [-1/2, 1/2]^D$ and $R \in \mathbb{R}^{D \times D}$ is the rotation matrix in this example. The γ-set of this example is $\mathcal{X}^\gamma = [-\gamma^{1/D}/2, \gamma^{1/D}/2]$. As $\lim_{D \to \infty} \gamma^{1/D} = 1$, which can be seen from the fact that the red lines become longer in 2D case, the marginal γ-set PDF $p_d(x_d|\mathcal{D}^{(l)}) \coloneqq \int_{\boldsymbol{x}_{-d} \in \mathcal{X}_{-d}} p(\boldsymbol{x}|\mathcal{D}^{(l)}) d\boldsymbol{x}_{-d}$ for each dimension (roughly speaking, the red lines for each dimension in Figure 3 show the region where the marginal γ-set PDF exhibits high density) approaches the uniform PDF in this example. However, the marginal γ-set PDF for each dimension will not converge to the uniform PDF when we have only a few observations. In fact, it is empirically known that the effective dimension D_e in HPO is typically much lower than D [Bergstra and Bengio, 2012]. This could imply that the marginal γ-set PDF for most dimensions go to the uniform PDF and only a fraction of dimensions have non-uniform PDF. In such cases, the following holds:

Algorithm 2 Meta-learning TPE

N_{init} (the number of initial samples), N_s (the number of candidates for each iteration), γ (the quantile to split $\mathcal{D}_\cdot$), ϵ (the ratio of random sampling), $\mathcal{D}_m$ (metadata)
$\mathcal{D}_1 \leftarrow \emptyset, \mathcal{D}_{\mathrm{init}} \leftarrow \emptyset$
for $m = 2, \dots, T$ **do** ▷ Create a warm-start set
  Add the top $\lceil N_{\mathrm{init}}/(T-1) \rceil$ in $\mathcal{D}_m$ to $\mathcal{D}_{\mathrm{init}}$
  ▷ Build KDEs for meta-tasks
  Sort $\mathcal{D}_m$ and build KDEs $p(\boldsymbol{x}|\mathcal{D}_m^{(l)}), p(\boldsymbol{x}|\mathcal{D}_m^{(g)})$
for $n = 1, \dots, N_{\mathrm{init}}$ **do** ▷ Initialization by warm-start
  Randomly pick $\boldsymbol{x}$ from $\mathcal{D}_{\mathrm{init}}$
  Pop $\boldsymbol{x}$ from $\mathcal{D}_{\mathrm{init}}$
  $\mathcal{D}_1 \leftarrow \mathcal{D}_1 \cup \{(\boldsymbol{x}, f_1(\boldsymbol{x}))\}$
while Budget is left **do**
  $\mathcal{S} = \emptyset$
  Sort $\mathcal{D}_1$ and build KDEs $p(\boldsymbol{x}|\mathcal{D}_1^{(l)}), p(\boldsymbol{x}|\mathcal{D}_1^{(g)})$
  for $m = 1, \dots, T$ **do**
    $\{\boldsymbol{x}_j\}_{j=1}^{N_s} \sim p(\boldsymbol{x}|\mathcal{D}_m^{(l)}), \mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{x}_j\}_{j=1}^{N_s}$
  Calculate the task kernel k_t by Algorithm 1
  if $r \leq \epsilon$ **then** ▷ $r \sim \mathcal{U}(0, 1)$, ϵ-greedy algorithm
    Randomly sample $\boldsymbol{x}$ and set $\boldsymbol{x}_{\mathrm{opt}} \leftarrow \boldsymbol{x}$
  else
    Pick $\boldsymbol{x}_{\mathrm{opt}} \in \mathrm{argmax}_{\boldsymbol{x} \in \mathcal{S}} \mathrm{EI}_{f^\gamma}[\boldsymbol{x}|t_1, \boldsymbol{D}]$ ▷ Eq. (6)
  $\mathcal{D}_1 \leftarrow \mathcal{D}_1 \cup \{(\boldsymbol{x}_{\mathrm{opt}}, f_1(\boldsymbol{x}_{\mathrm{opt}}))\}$

Proposition 2 *If some dimensions are trivial on two tasks, the γ-set similarity between those tasks is identical irrespective of with or without those dimensions.*

Roughly speaking, a trivial dimension is a dimension that does not contribute to $f(\boldsymbol{x})$ at all. The formal definition of the trivial dimensions and the proof are provided in Appendix B.6. As HP selection does not change the γ-set similarity under such circumstances, we would like to employ a dimension reduction method and we choose an HPI-based dimension reduction. The reason behind this choice is that HPO often has categorical parameters and other methods, such as principle component analysis and singular value decomposition, mix up categorical and numerical parameters. In this paper, we use PED-ANOVA [Watanabe *et al.*, 2023], which

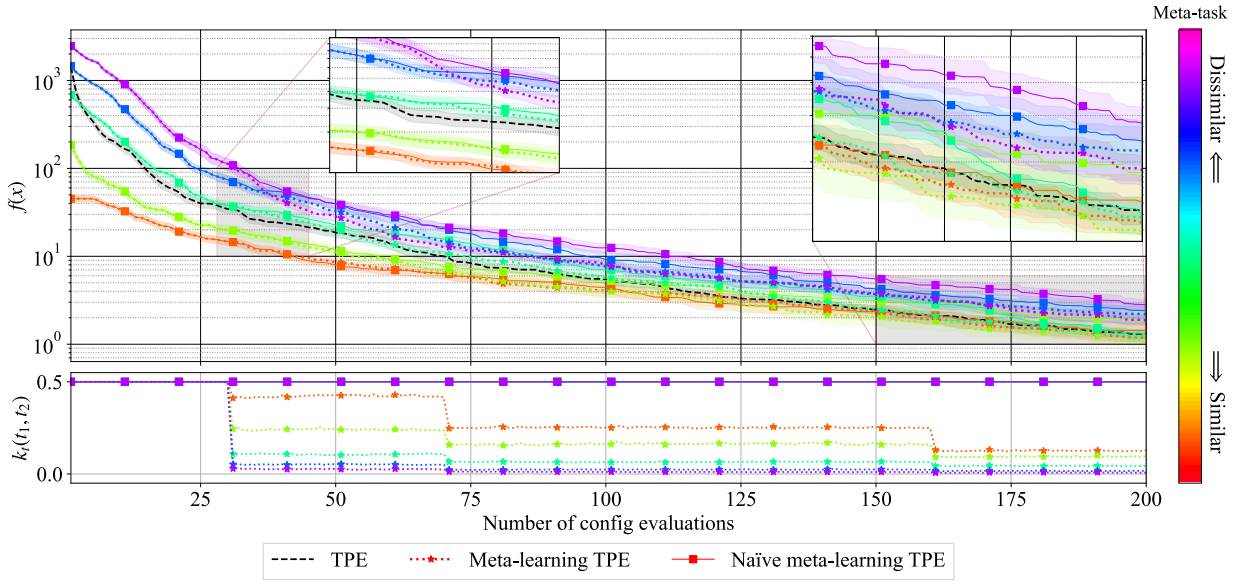

Figure 2: The comparison of the convergence of TPE and meta-learning TPE based on the task similarity. $c^\star = 0$ is identical to the target task and tasks become dissimilar as $c^\star$ becomes larger. **Top**: each line except the black line is the performance curves of meta-learning TPE on differently similar tasks (orange is similar and purple is dissimilar). Dotted lines with $\star$ markers are for meta-learning TPE and solid lines with ■ markers are for naïve meta-learning TPE. Weak-color bands show the standard error of the objective function value over 50 independent runs. **Bottom**: the medians of the task weight on the meta-task (higher is similar).

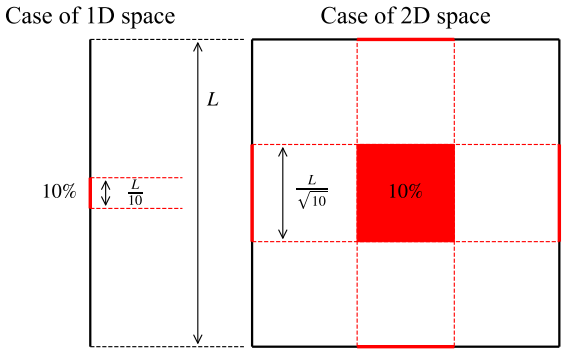

Figure 3: The conceptual visualization where the top-10% domain for each dimension becomes larger when the importance of each dimension is same and there is no interaction between dimensions. The thick black lines are the edges of each domain and the red lines are the important domains for each dimension. The red lines become longer as the dimensionality becomes higher and it implies that the marginal γ-set PDF approaches the uniform PDF as D goes to ∞.

computes HPI for each dimension via Pearson divergence between the marginal γ-set PDF and the uniform PDF.

Algorithm 1 includes the pseudocode of the dimension reduction. We first compute HPIs in Eq. (16) by Watanabe *et al.* [2023] for each dimension and take the average of HPI:

$$\mathbb{V}_{d,m} := \gamma^2 \mathbb{E}_{x_d \sim \mathcal{X}_d}\left[\left(\frac{p_d(x_d|\mathcal{D}_m^{(l)})}{u(\mathcal{X}_d)} - 1\right)^2\right], \qquad \bar{\mathbb{V}}_d := \frac{1}{T}\sum_{m=1}^{T} \mathbb{V}_{d,m} \tag{11}$$

where $u(\mathcal{X}_d)$ is the uniform PDF defined on $\mathcal{X}_d$. Then we pick the top-$\lfloor \log_\eta |\mathcal{D}_1^{(l)}| \rfloor$ dimensions with respect to $\bar{\mathbb{V}}_d$ and define the set of dimensions $\mathcal{I} \in 2^{\{1,\dots,D\}}$. While we compute the original KDE via:

$$p(\boldsymbol{x}|\mathcal{D}') = \frac{1}{N}\sum_{n=1}^{N}\prod_{d=1}^{D} k_d(x_d, x_{d,n}) \tag{12}$$

where $\mathcal{D}' := \{(x_{1,n}, x_{2,n}, \dots, x_{D,n}, f(\boldsymbol{x}_n))\}_{n=1}^N$ and k_d is the kernel function for the d-th dimension, we compute the reduced PDF via:

$$p^{\mathrm{DR}}(\boldsymbol{x}|\mathcal{D}') = \frac{1}{N}\sum_{n=1}^{N}\prod_{d\in\mathcal{I}} k_d(x_d, x_{d,n}). \tag{13}$$

4.4 Algorithm Description

Algorithm 2 presents the whole pseudocode of our meta-learning TPE and the color-coding shows our propositions. To stabilize the approximation of the task kernel, we employ the dimension reduction shown in Algorithm 1 and the ϵ-greedy algorithm at the optimization of the AF in Line 17 of Algorithm 2 as discussed in Section 4.3. Furthermore, we use the warm-start initialization as seen in Lines 3 – 10 of Algorithm 2. The warm-start further speeds up optimizations. Note that we apply the same warm-start initialization to all meta-learning methods for fair comparisons in the experiments. To extend our method to MO settings, all we need is to employ a rank metric $R : \mathbb{R}^m \to \mathbb{R}$ for an objective vector $\boldsymbol{f}$ as mentioned in Appendix B.3. In this paper, we consistently use the non-domination rank and the crowding distance for all methods to realize fair comparisons.

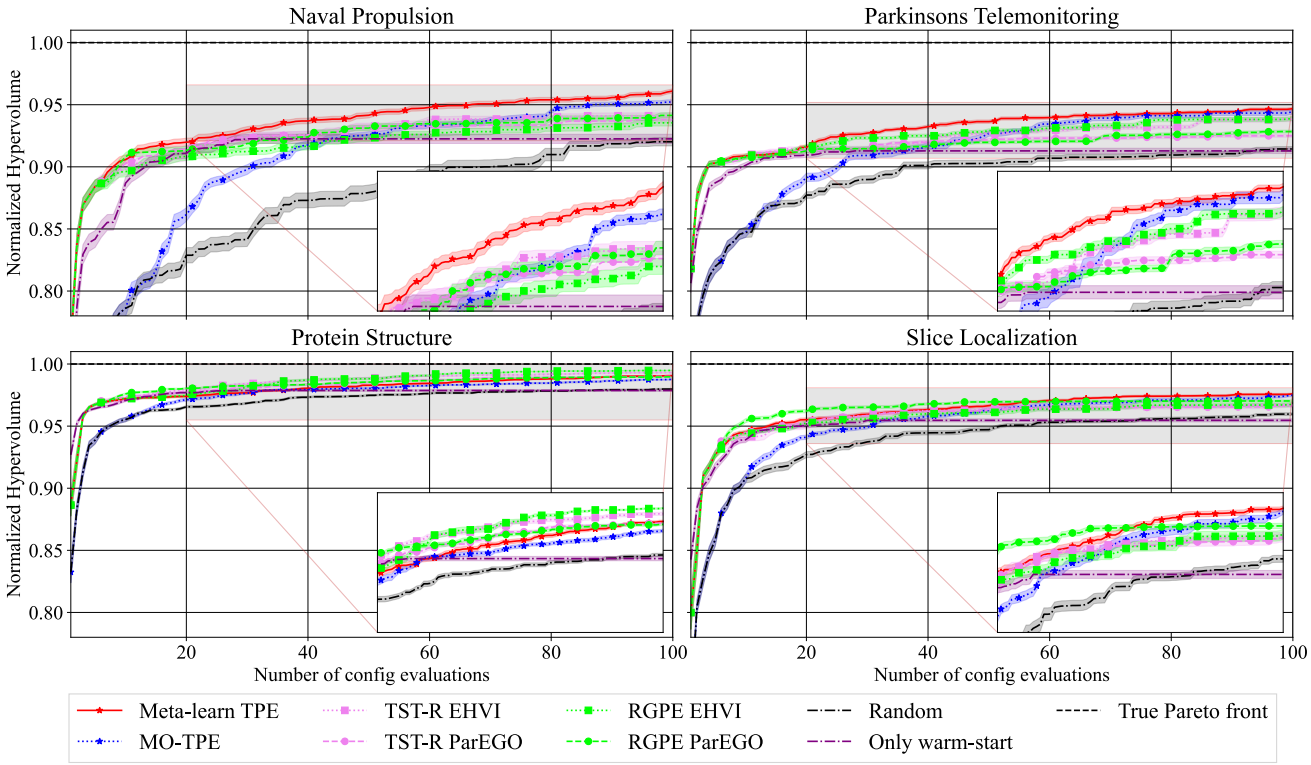

Figure 4: The normalized HV over time on four joint neural architecture search and hyperparameter optimization benchmarks (HPOlib) from HPOBench. Each method was run with 20 different random seeds and the weak-color bands present standard error. The small inset figures in each figure are the magnified gray areas. See Appendix D for the Pareto fronts achieved by 50% [Watanabe, 2023a] of the runs.

4.5 Validation of Modifications

To see the effect of the task kernel, we conduct an experiment using the ellipsoid function $f(\boldsymbol{x}|c) := f(x_1, \ldots, x_4|c) = \sum_{d=1}^{4} 5^{d-1}(x_d - c)^2$ defined on $[-5, 5]^4$. Along with the original TPE, we optimized $f(\boldsymbol{x}|c = 0)$ by meta-learning TPE using the randomly sampled 100 observations from $f(\boldsymbol{x}|c_\star)$ where $c^\star \in [0, 1, \ldots, 4]$ (each run uses only one of $[0, 1, \ldots, 4]$) as metadata. Furthermore, we also evaluated naïve meta-learning TPE that considers $k_t(t_i, t_j) = 1/T$ for all pairs of tasks. All control parameter settings followed Section 5 except we evaluated 200 configurations.

In Figure 2, we present the result. The top figure shows the performance curve and the bottom figure shows the weight ($k_t(t_1, t_2) \in [0, 1]$) on the meta-task. As seen in the figure, the performance rank is proportional to the task similarity in the early stage of the optimizations, and thus meta-learning TPE on the dissimilar tasks performed poorly in the beginning. However, as the number of evaluations increases, the performance curves of dissimilar meta-tasks quickly approach that of TPE after 30 evaluations where $\lfloor \log_\eta |\mathcal{D}_1^{(l)}| \rfloor$ first becomes non-zero for $\eta = 2.5$. We can also see the task weight is also ordered by the similarity between the target task and the meta-task. Thanks to this effect, on the dissimilar tasks (purple, blue lines), our method starts to recover the performance of TPE from that point (see the first inset figure) and our method showed closer performance to the original TPE compared to the naïve meta-learning TPE (see the second inset figure). This result demonstrates the robustness of our method to the knowledge transfer from dissimilar meta-tasks. For similar tasks (light green, orange lines), our method accelerates at the early stage and slowly converges to the performance of TPE. Notice that since we use random search for the metadata and the observations are from the meta-learning TPE sampler, which is obviously a non-i.i.d sampler due to the iterative update nature, the top-γ-quantile in observations obtained from our method is concentrated in a subset of the true top-γ-quantile as stated in Theorem 1. It implies that the weight for meta-task is expected to decrease over time even if the target task is identical to meta-tasks.

5 Experiments

5.1 Setup

In the experiments, we optimize two metrics (a validation loss or accuracy metric, and runtime) on four joint NAS & HPO benchmarks (HPOlib) [Klein and Hutter, 2019] in HPOBench [Eggensperger *et al.*, 2021], as well as NMT-Bench [Zhang and Duh, 2020]. The baselines are as follows:

1. RGPE either with EHVI or ParEGO,
2. TST-R either with EHVI or ParEGO,
3. Random search,
4. Only warm-start (top-10% configurations in metadata),
5. MO-TPE [Ozaki *et al.*, 2020; Ozaki *et al.*, 2022].

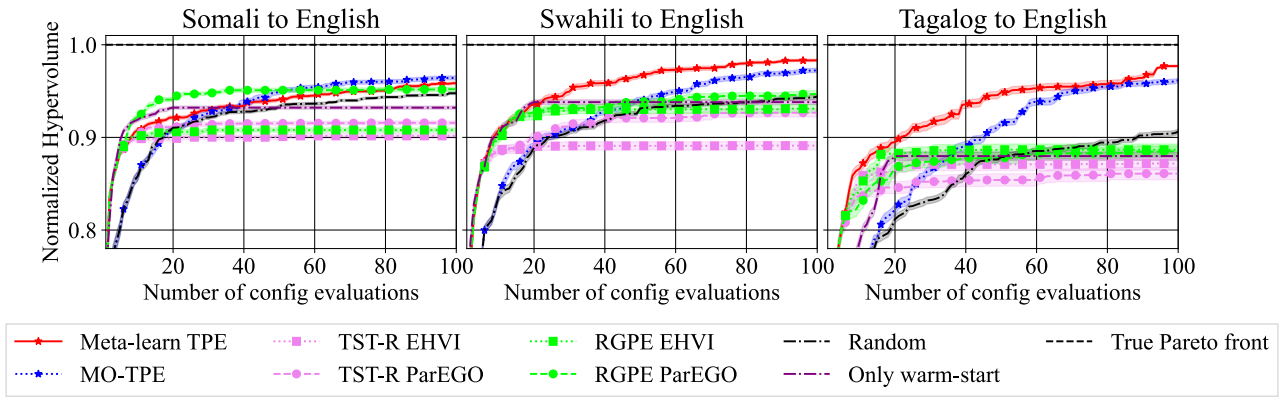

Figure 5: The normalized HV over time on NMT-Bench. Each method was run with 20 different random seeds and the weak-color bands present standard error. See Appendix D for the Pareto fronts achieved by 50% [Watanabe, 2023a] of the runs.

RGPE [Feurer *et al.*, 2018] and TST-R [Wistuba *et al.*, 2016] were reported to show the best average performance in a diverse set of meta-learning BO methods [Feurer *et al.*, 2018]. Note that since RGPE and TST-R require a rank metric to compute the ranking loss, we used non-domination rank and crowding distance, which we use for meta-learning TPE as well. Each meta-learning method uses 100 random configurations from the other datasets in each tabular benchmark and uses the warm-start initialization ($N_{\text{init}} = 5$) in Algorithm 2. `Only warm-start` serves as an indicator of how good the initialization could be and we can judge whether warm-start helps or the meta-learning methods help. In this setup, Algorithm 1 takes 1.0×10^{-4} seconds for a $10D$ space with 200 observations for 5 meta-tasks. In Appendix C, we describe more details about control parameters of each method and tabular benchmarks and discuss the effect of the control parameter η and the number of meta-tasks on the performance.

The performance of each experiment was measured via the normalized HV. When we define the worst (maximum) and the best (minimum) values of each objective as $f_i^{\max}, f_i^{\min}$ for $i \in \{1, \dots, M\}$, the normalized HV is computed as:

$$\prod_{i=1}^{M} \frac{f_i^{\max} - f_i}{f_i^{\max} - f_i^{\min}}. \tag{14}$$

In principle, the normalized HV is better when it is higher and the possible best value is 1, which is shown as *True Pareto front*. Although the normalized HV curve tells us how much each method could improve solution sets, it does not visualize how solutions distribute in the objective space (on average). Therefore, we also provide the 50% empirical attainment surfaces [Fonseca and Fleming, 1996] in Appendix D.

5.2 Results on Real-World Tabular Benchmarks

Figure 4 shows the results on HPOlib. For all datasets, `Only warm-start` quickly yields results slightly worse than Random search with as many as 100 function evaluations. This implies that the knowledge transfer surely helps at the early stage of each optimization, but each method still needs to explore better configurations. Our meta-learning TPE method shows the best performance curves except for `Protein Structure`; however, our method did not exhibit the best performance on `Protein Structure`, where its performance is still competitive. Furthermore, while we found out that `Protein Structure` is relatively dissimilar from the other benchmarks (as discussed in Appendix D), our method could still outperform the non-transfer MO-TPE.

Figure 5 shows the results on NMT-Bench. As in HPOlib, `Only warm-start` quickly yields results slightly worse than Random search with as many as 100 function evaluations for all datasets in NMT-Bench as well. On the other hand, while RGPE and TST-R exhibit performance indistinguishable from `Only warm-start` in most cases, our method still improved until the end. This implies that `Only warm-start` is not sufficient in this task and our method could make use of the knowledge from metadata. However, our meta-learning TPE method struggled in `Somali to English`, which was dissimilar to the other datasets as discussed in Appendix D. Although our method did not exhibit the best performance in this case, it still recovered well with enough observations and got the best performance in the end.

6 Conclusion

In this paper, we introduced a multi-task training method for TPE and demonstrated the performance of our method. Our method measures the similarity between tasks using the intersection over union and computes the task kernel based on the similarity. As the vanilla version of this simple meta-learning method has some drawbacks, we employed the ϵ-greedy algorithm and the dimension reduction method to stabilize our task kernel. In the experiment using a synthetic function, we confirmed that our task kernel correctly ranks the task similarity and our method could demonstrate robust performance over various task similarities. In the real-world experiments, we used two tabular benchmarks of expensive HPO problems. Our method could outperform other methods in most settings and still exhibit competitive performance in the worst case of dissimilar tasks. Our method's strong performance is also backed by winning the *AutoML 2022 competition on "Multi-objective Hyperparameter Optimization for Transformers"*.

Acknowledgments

The authors appreciate the valuable contributions of the anonymous reviewers and helpful feedback from Ryu Minegishi. Robert Bosch GmbH is acknowledged for financial support. The authors also acknowledge funding by European Research Council (ERC) Consolidator Grant "Deep Learning 2.0" (grant no. 101045765). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the ERC. Neither the European Union nor the ERC can be held responsible for them.

Funded by
the European Union

References

[Bader and Zitzler, 2011] J. Bader and E. Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19, 2011.

[Bergstra and Bengio, 2012] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2), 2012.

[Bergstra *et al.*, 2011] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, 2011.

[Bergstra *et al.*, 2013] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, 2013.

[Brochu *et al.*, 2010] E. Brochu, V. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv:1012.2599*, 2010.

[Candelieri *et al.*, 2022] A. Candelieri, A. Ponti, and F. Archetti. Fair and green hyperparameter optimization via multi-objective and multiple information source Bayesian optimization. *arXiv:2205.08835*, 2022.

[Chen *et al.*, 2018] Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas. Bayesian optimization in AlphaGo. *arXiv:1812.06855*, 2018.

[Chugh *et al.*, 2016] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya. A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. *Evolutionary Computation*, 22, 2016.

[Deb *et al.*, 2002] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation*, 6, 2002.

[Eggensperger *et al.*, 2021] K. Eggensperger, P. Müller, N. Mallik, M. Feurer, R. Sass, A. Klein, N. Awad, M. Lindauer, and F. Hutter. HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. In *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[Emmerich *et al.*, 2011] MTM. Emmerich, AH. Deutz, and JW. Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *Congress of Evolutionary Computation*, 2011.

[Falkner *et al.*, 2018] S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, 2018.

[Feurer *et al.*, 2015] M. Feurer, JT. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *AAAI Conference on Artificial Intelligence*, 2015.

[Feurer *et al.*, 2018] M. Feurer, B. Letham, F. Hutter, and E. Bakshy. Practical transfer learning for Bayesian optimization. *arXiv:1802.02219*, 2018.

[Fonseca and Fleming, 1996] CM. Fonseca and PJ. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In *International Conference on Parallel Problem Solving from Nature*, 1996.

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

[Guo *et al.*, 2018] D. Guo, Y. Jin, J. Ding, and T. Chai. Heterogeneous ensemble-based infill criterion for evolutionary multiobjective optimization of expensive problems. *Transactions on Cybernetics*, 49, 2018.

[Henderson *et al.*, 2018] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, 2018.

[Hernández-Lobato *et al.*, 2016] D. Hernández-Lobato, J. Hernandez-Lobato, A. Shah, and R. Adams. Predictive entropy search for multi-objective Bayesian optimization. In *International Conference on Machine Learning*, 2016.

[Jones *et al.*, 1998] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13, 1998.

[Klein and Hutter, 2019] A. Klein and F. Hutter. Tabular benchmarks for joint architecture and hyperparameter optimization. *arXiv:1905.04970*, 2019.

[Knowles, 2006] J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *Evolutionary Computation*, 10, 2006.

[Kushner, 1964] HJ. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Joint Automatic Control Conference*, 1964.

[Nomura *et al.*, 2021] M. Nomura, S. Watanabe, Y. Akimoto, Y. Ozaki, and M. Onishi. Warm starting CMA-ES for hyperparameter optimization. In *AAAI Conference on Artificial Intelligence*, 2021.

[Ozaki *et al.*, 2020] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi. Multiobjective tree-structured Parzen estimator for computationally expensive optimization problems. In *Genetic and Evolutionary Computation Conference*, 2020.

[Ozaki *et al.*, 2022] Y. Ozaki, Y. Tanigaki, S. Watanabe, M. Nomura, and M. Onishi. Multiobjective tree-structured Parzen estimator. *Journal of Artificial Intelligence Research*, 73, 2022.

[Pan *et al.*, 2018] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, and Y. Jin. A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization. *Evolutionary Computation*, 23, 2018.

[Perrone *et al.*, 2018] V. Perrone, R. Jenatton, M. Seeger, and C. Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, 2018.

[Perrone *et al.*, 2019] V. Perrone, H. Shen, MW. Seeger, C. Archambeau, and R. Jenatton. Learning search spaces for Bayesian optimization: Another view of hyperparameter transfer learning. *Advances in Neural Information Processing Systems*, 2019.

[Ponweiser *et al.*, 2008] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection. In *International Conference on Parallel Problem Solving from Nature*, 2008.

[Salinas *et al.*, 2020] D. Salinas, H. Shen, and V. Perrone. A quantile-based approach for hyperparameter transfer learning. In *International Conference on Machine Learning*, 2020.

[Schmucker *et al.*, 2020] R. Schmucker, M. Donini, V. Perrone, MB. Zafar, and C. Archambeau. Multi-objective multi-fidelity hyperparameter optimization with application to fairness. In *Meta-Learning Workshop at Advances in Neural Information Processing Systems*, 2020.

[Shahriari *et al.*, 2016] B. Shahriari, K. Swersky, Z. Wang, R. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *IEEE*, 104, 2016.

[Song *et al.*, 2022] J. Song, L. Yu, W. Neiswanger, and S. Ermon. A general recipe for likelihood-free Bayesian optimization. In *International Conference on Machine Learning*, 2022.

[Springenberg *et al.*, 2016] JT. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[Swersky *et al.*, 2013] K. Swersky, J. Snoek, and R. Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2013.

[Vanschoren, 2019] J. Vanschoren. Meta-learning. In *Automated Machine Learning*, pages 35–61. Springer, 2019.

[Volpp *et al.*, 2020] M. Volpp, LP. Fröhlich, K. Fischer, A. Doerr, S. Falkner, F. Hutter, and C. Daniel. Meta-learning acquisition functions for transfer learning in Bayesian optimization. In *International Conference on Learning Representations*, 2020.

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

[Watanabe and Hutter, 2022] S. Watanabe and F. Hutter. c-TPE: Generalizing tree-structured Parzen estimator with inequality constraints for continuous and categorical hyperparameter optimization. *Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems Workshop at Advances in Neural Information Processing Systems*, 2022.

[Watanabe and Hutter, 2023] S. Watanabe and F. Hutter. c-TPE: Tree-structured Parzen estimator with inequality constraints for expensive hyperparameter optimization. *arXiv:2211.14411*, 2023.

[Watanabe *et al.*, 2023] S. Watanabe, A. Bansal, and F. Hutter. PED-ANOVA: Efficiently quantifying hyperparameter importance in arbitrary subspaces. *arXiv:2304.10255*, 2023.

[Watanabe, 2023a] S. Watanabe. Python tool for visualizing variability of Pareto fronts over multiple runs. 2023.

[Watanabe, 2023b] S. Watanabe. Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv:2304.11127*, 2023.

[Wistuba *et al.*, 2015] M. Wistuba, N. Schilling, and L. Schmidt-Thieme. Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2015.

[Wistuba *et al.*, 2016] M. Wistuba, N. Schilling, and L. Schmidt-Thieme. Two-stage transfer surrogate model for automatic hyperparameter optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2016.

[Zhang and Duh, 2020] X. Zhang and K. Duh. Reproducible and efficient benchmarks for hyperparameter optimization of neural machine translation systems. *Transactions of the Association for Computational Linguistics*, 8, 2020.

[Zhang and Li, 2007] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation*, 11, 2007.