Multi-objective Tree-structured Parzen Estimator Meets Meta-learning

Shuhei Watanabe [*][1]**, Noor Awad** [1]**, Masaki Onishi** [2]**, Frank Hutter**[1, 3]
[1] Department of Computer Science, University of Freiburg, Germany
[2] Artificial Intelligence Research Center, AIST, Tokyo, Japan
[3] Bosch Center for Artificial Intelligence, Renningen, Germany
{watanabs,awad,fh}@cs.uni-freiburg.de

Abstract

Hyperparameter optimization (HPO) is essential for the better performance of deep learning, and practitioners often need to consider the trade-off between multiple metrics, such as error rate, latency, memory requirements, robustness, and algorithmic fairness. Due to this demand and the heavy computation of deep learning, the acceleration of multi-objective (MO) optimization becomes ever more important. Although meta-learning has been extensively studied to speedup HPO, existing methods are not applicable to the 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 in promising regions of each task. In a comprehensive set of experiments, we demonstrate that our method accelerates MO-TPE on tabular HPO benchmarks and yields state-of-the-art performance. Our method was also validated externally by winning the *AutoML 2022 competition on "Multiobjective Hyperparameter Optimization for Transformers"*.

1 Introduction

While deep learning has achieved various breakthrough successes, its performance highly depends on proper settings of its hyperparameters (HP) (Chen *et al.* (2018); Melis *et al.* (2018); Henderson *et al.* (2018)). Furthermore, practical applications often impose important metrics to optimize next to error rate, such as latency of inference, memory requirements, robustness, and algorithmic fairness (Schmucker *et al.* (2020); Cruz *et al.* (2021); Candelieri *et al.* (2022)). However, exploring the Pareto front of several objectives is more complex than single-objective optimization, making it particularly important to accelerate MO optimization.

To accelerate HP optimization (HPO), meta-learning has been actively studied, as surveyed, e.g., by Vanschoren (2019). In the context of HPO, meta-learning mainly focuses on 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)). Those methods use meta information in Gaussian process (GP) regression to yield more informed surrogates 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) (Bergstra *et al.* (2011, 2013)) is stronger than those methods in expensive MO settings (Ozaki *et al.* (2020, 2022)). Since this method uses kernel density estimators (KDEs) rather than GPs, the existing meta-learning methods are not directly applicable and thus a new meta-learning procedure must be invented.

To address this issue, we propose a meta-learning method for TPE using a new task similarity measure. Our method models the joint probability density function (PDF) of a HP configuration $\boldsymbol{x}$ and a task t under the assumption of conditional shift (Zhang *et al.* (2013)). Since the joint

[*] The work was partially done in AIST.

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

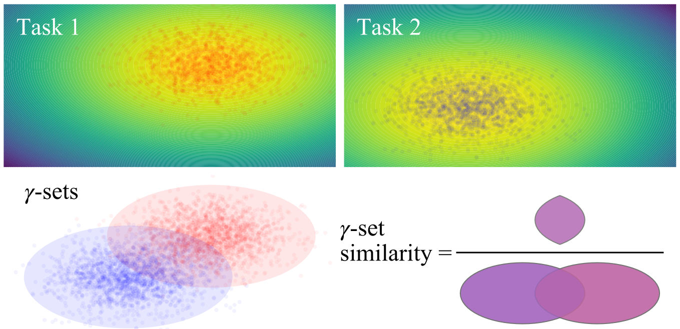

Figure 1: The conceptual visualization of the task similarity measure. The similarity is measured via intersection over union of the γ-set (see Definition 2 in Appendix B) PDFs.

PDF requires a similarity measure between tasks $k_t(t_i, t_j)$, we introduce a task similarity measure using the intersection over union (which we call *γ-set similarity*) as visualized in Figure 1. Note that we discuss more details about components of the task similarity in Appendix C In the series of experiments, we demonstrate that our method successfully speeds up MO-TPE (or at least recovers the performance of MO-TPE in case meta-tasks are not similar). Our method's strong performance was also validated externally by winning the *AutoML 2022 competition on "Multiobjective Hyperparameter Optimization for Transformers"*. Our source code is available at `https://github.com/nabenabe0928/meta-learn-tpe`.

2 Meta-learning for TPE

In this section, we briefly explain the TPE formulation and then describe the formulation of the acquisition function (AF) for the meta-learning setting. For more details about background and related work, see Appendices A and B, respectively. Note that since our method can be easily extended to MO settings, for simplicity we first discuss the single-objective setting and only describe how to extend it to the MO setting later.

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 D_m is the set of observations on the m-th task, of size $N_m \coloneqq |\mathcal{D}_m|$. We use the notion of a γ-set, which, roughly speaking, is the sublevel set of all configurations $\boldsymbol{x} \in \mathcal{X}$ with $f(\boldsymbol{x}) \leq f^\gamma$, with f^γ being the γ quantile of f over the space $\mathcal{X}$. Figure 1 visualizes the concept of the γ-set; we refer to Appendix B for the formal definition. We denote the Lebesgue measure on $\mathcal{X}$ as $\mu(\mathcal{X})$ (more details in Appendix B) and we define $\mathcal{X}_m^\gamma$ as the γ-set of the m-th task in the search space $\mathcal{X}$.

2.1 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 and then builds KDEs $p(\boldsymbol{x}|\mathcal{D}^{(l)})$ and $p(\boldsymbol{x}|\mathcal{D}^{(g)})$. Then TPE computes its AF via $\gamma p(\boldsymbol{x}|\mathcal{D}^{(l)})/(\gamma p(\boldsymbol{x}|\mathcal{D}^{(l)}) + (1-\gamma)p(\boldsymbol{x}|\mathcal{D}^{(g)}))$. In the same vein, the task-conditioned AF is:

$$\mathrm{EI}_{f^\gamma}[\boldsymbol{x}|t, \boldsymbol{D}] \propto \frac{\gamma p(\boldsymbol{x}, t|\boldsymbol{D}^{(l)})}{\gamma p(\boldsymbol{x}, t|\boldsymbol{D}^{(l)}) + (1-\gamma)p(\boldsymbol{x}, t|\boldsymbol{D}^{(g)})}. \tag{1}$$

This formulation transfers the knowledge of well-performing regions and weights the knowledge from similar tasks more. Note that we use the conditional shift for the transformation; see Appendix C.1 for more details. 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.

2.2 Task similarity

To measure the similarity $k_t(t_i, t_j)$ between tasks, we use the γ-set similarity visualized in Figure 1 (see Appendix B for the formal definition). From Thoerem 1 in Appendix C, $\hat{s}(\mathcal{D}_i^{(l)}, \mathcal{D}_j^{(l)}) \coloneqq (1-d_{\mathrm{tv}}(p_i, p_j))/(1+d_{\mathrm{tv}}(p_i, p_j))$ 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 $m = 1, \dots, T$ where we define $p_m \coloneqq p(\boldsymbol{x}|\mathcal{D}_m^{(l)})$, t_m as a meta-task for $m = 2, \dots, T$ and t_1 as the target task, $d_{\mathrm{tv}}(p_i, p_j)$ is the total variation distance $1/2 \int_{\boldsymbol{x}\in\mathcal{X}} |p(\boldsymbol{x}|\mathcal{D}_i^{(l)}) - p(\boldsymbol{x}|\mathcal{D}_j^{(l)})|\mu(d\boldsymbol{x})$, and $p(\boldsymbol{x}|\mathcal{D}_i^{(l)})$ is estimated by KDE. Note that $d_{\mathrm{tv}}(p_i, p_j)$ is approximated via Monte-Carlo sampling. Then we define the task kernel as follows:

$$k_t(t_i, t_j) = \begin{cases} \frac{1}{T}\hat{s}(\mathcal{D}_i^{(l)}, \mathcal{D}_j^{(l)}) & (i \neq j) \\ 1 - \frac{1}{T}\sum_{k\neq i}\hat{s}(\mathcal{D}_i^{(l)}, \mathcal{D}_k^{(l)}) & (i = j) \end{cases}. \tag{2}$$

Algorithm 1 Task similarity (see Appendix C.3 for more details)

```
$\eta$ (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 importance $\bar{\mathbb{V}}_d$ for each dimension over tasks based on Eq. (16)
Choose $\lfloor \log_\eta |\mathcal{D}_1^{(l)}| \rfloor$ dimensions from higher $\bar{\mathbb{V}}_d$ and re-build $l_m^{\mathrm{DR}}(\boldsymbol{x})$ based on Eq. (18)
for $m = 2, \dots, T$ do
    Calculate $d_{\mathrm{tv}}$ in Eq. (15) with $l_1^{\mathrm{DR}}, l_m^{\mathrm{DR}}$       ▷ Use $S$ samples for Monte-carlo sampling
    Calculate $k_t(t_1, t_m)$ based on Eq. (2)
return $k_t$
```

Algorithm 2 Meta-learning TPE

```
1:  $N_{\mathrm{init}}$ (the number of initial samples), $N_s$ (the number of candidates for each iteration), $\gamma$ (the
    quantile to split $\mathcal{D}$), $\epsilon$ (the ratio of random sampling)
2:  $\mathcal{D} \leftarrow \emptyset, \mathcal{D}_{\mathrm{init}} \leftarrow \emptyset$
3:  for $m = 2, \dots, T$ do                                                ▷ Create a warm-start set
4:      Add the top $\lceil N_{\mathrm{init}}/(T-1) \rceil$ in $\mathcal{D}_m$ to $\mathcal{D}_{\mathrm{init}}$
5:      Sort $\mathcal{D}_m$ and build KDEs $p(\boldsymbol{x}|\mathcal{D}_m^{(l)}), p(\boldsymbol{x}|\mathcal{D}_m^{(g)})$          ▷ Build KDEs for meta-tasks
6:  for $n = 1, \dots, N_{\mathrm{init}}$ do                                          ▷ Initialization by warm-start
7:      Randomly pick $\boldsymbol{x}$ from $\mathcal{D}_{\mathrm{init}}$ and pop it from $\mathcal{D}_{\mathrm{init}}$
8:      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}, f_1(\boldsymbol{x}))\}$
9:  while Budget is left do
10:     $\mathcal{S} = \emptyset$
11:     Sort $\mathcal{D}_1$ and build KDEs $p(\boldsymbol{x}|\mathcal{D}_1^{(l)}), p(\boldsymbol{x}|\mathcal{D}_1^{(g)})$
12:     for $m = 1, \dots, T$ do
13:         $\{\boldsymbol{x}_j\}_{j=1}^{N_s} \sim p(\boldsymbol{x}|\mathcal{D}_m^{(l)}), \mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{x}_j\}_{j=1}^{N_s}$
14:     Calculate the task kernel $k_t$ by Algorithm 1
15:     if $r \leq \epsilon$ then                                  ▷ $r \sim \mathcal{U}(0,1)$, $\epsilon$-greedy algorithm
16:         Randomly sample $\boldsymbol{x}$ and set $\boldsymbol{x}_{\mathrm{opt}} \leftarrow \boldsymbol{x}$
17:     else
18:         Pick $\boldsymbol{x}_{\mathrm{opt}} \in \mathrm{argmax}_{\boldsymbol{x} \in \mathcal{S}} \mathrm{EI}_{f^\gamma}[\boldsymbol{x}|t, \boldsymbol{D}]$                       ▷ Eq. (1)
19:     $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}_{\mathrm{opt}}, f_1(\boldsymbol{x}_{\mathrm{opt}}))\}$
```

We define the kernel so that it is symmetric and the summation over all tasks is 1. Using this kernel, we build KDEs as follows:

$$p(\boldsymbol{x}, t|\boldsymbol{D}') = \frac{1}{N'_{\mathrm{all}}} \sum_{m=1}^{T} k_t(t, t_m) \sum_{n=1}^{N'_m} k_x(\boldsymbol{x}, \boldsymbol{x}_{m,n}|\boldsymbol{\alpha}_m) = \frac{1}{N'_{\mathrm{all}}} \sum_{m=1}^{T} N'_m k_t(t, t_m) p(\boldsymbol{x}|\mathcal{D}'_m) \tag{3}$$

where $\boldsymbol{D}' \coloneqq \{\mathcal{D}'_m\}_{m=1}^{T}$ is a set of subsets of the observations on the m-th task $\mathcal{D}'_m = \{(\boldsymbol{x}_{m,n}, f_m(\boldsymbol{x}_{m,n}))\}_{n=1}^{N_m}$, $N'_{\mathrm{all}} = \sum_{m=1}^{T} N'_m$, and $\boldsymbol{\alpha}_m$ is a set of control parameters of the kernel function k_x for the m-th task. 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. $\hat{s}(t_1, t_m) = 0$ (see Appendix C for more details), and (2) asymtotically converge to the original formulation as the sample size goes to infinity, i.e. $\lim_{N'_1 \rightarrow \infty} p(\boldsymbol{x}, t|\boldsymbol{D}') = p(\boldsymbol{x}|\mathcal{D}'_1)$.

2.3 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 similarity, we employ the dimension reduction shown in Algorithm 1 and the ϵ-greedy algorithm at the optimization of the AF in Line 15 of Algorithm 2; see Appendix C for more details about the modifications. Furthermore, we use the warm-start initialization as seen in Lines 3 – 8 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.

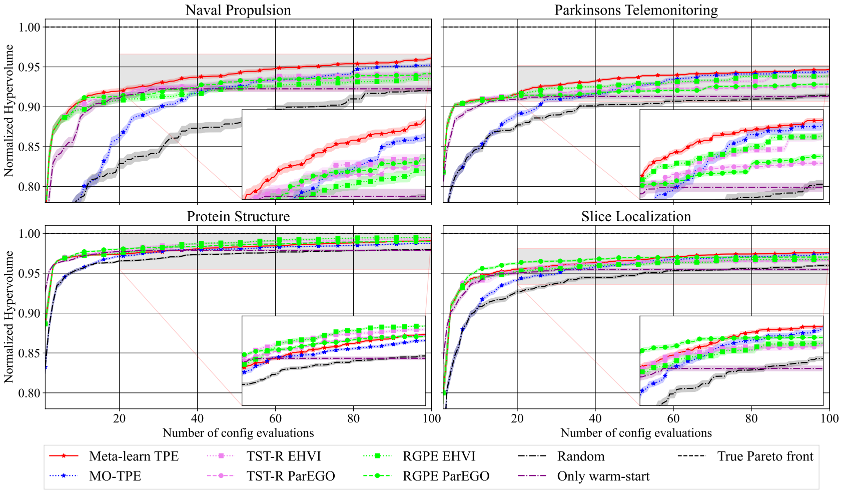

Figure 2: The normalized Hypervolume over time on HPOlib. Each method was run with 20 different random seeds and the weak-color bands present standard error. The small inset figures in each figure are the magnified gray areas.

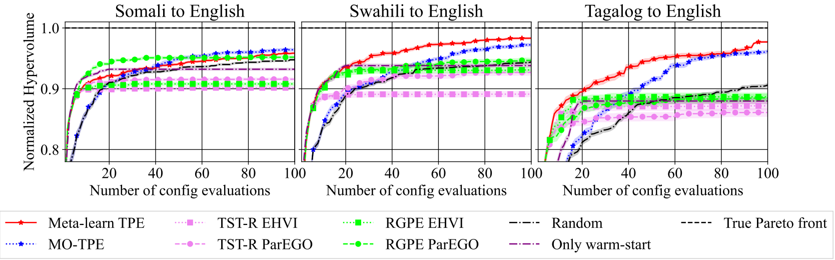

Figure 3: The normalized Hypervolume over time on NMT-Bench. Each method was run with 20 different random seeds and the weak-color bands present standard error.

Finally, we briefly mention how to extend our method to MO settings. As both TPE and MO-TPE consider the ranking among observations, we only need to employ a sorting method such that we can split observations into two parts, i.e. the top-γ-quantile and the other. Then earlier discussion is directly applicable to MO settings as well.

3 Experiments & results

In the experiments, we optimize two metrics (a validation loss or accuracy metric and runtime) in HPOlib (Klein and Hutter (2019)) and NMT-Bench (Zhang and Duh (2020)). For baseline methods, we choose the meta-learning methods that showed the best and the second best average performance reported by Feurer *et al.* (2018): RGPE (Feurer *et al.* (2018)) and TST-R (Wistuba *et al.* (2016)) combined with ParEGO (Knowles (2006)) or EHVI (Emmerich *et al.* (2011)). Furthermore, we evaluate MO-TPE, random search, and only warm-start, which is the top-10% configurations in each meta-task, as baselines. 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. For more details about control parameters of each method and tabular benchmarks, see Appendix E.

Figures 2 and 3 show the normalized Hypervolume over time. As all meta-learning methods use the same warm-start algorithm up to the first five configurations, it makes a difference from random search and MO-TPE. After the initialization (see the magnified inset figures), our method shows better performance except for "Protein Structure" and "Slice Localization" in HPOlib and "Somali to English" in NMT-Bench. This cold-start problem happens if meta-tasks are relatively dissimilar as discussed in Appendix C.2, but our method recovers well from it with enough function evaluations. Our method is also backed by our method also winning the *AutoML 2022 competition on "Multiobjective Hyperparameter Optimization for Transformers"*.

Acknowledgments

The authors acknowledge funding by European Research Council (ERC) Consolidator Grant "Deep Learning 2.0" (grant no. 101045765). Funded by the European Union. Views and opinions expressed are however those of the author(s) 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. The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG. Robert Bosch GmbH is acknowledged for financial support. This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215. The authors also would like to acknowledge the valuable contributions of the anonymous reviewers and helpful feedback from Eddie Bergman and Noor Awad.

Funded by
the European Union

References

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

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.

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.

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

AF. Cruz, P. Saleiro, C. Belém, C. Soares, and P. Bizarro. Promoting fairness through hyperparameter optimization. In *International Conference on Data Mining*, 2021.

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

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

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

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

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

G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*, 2018.

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.

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

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

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.

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

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

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

S. Watanabe and F. Hutter. Revisiting hyperparameter importance assessment: Local importance from the Lebesgue view. *arXiv:4522.459*, 2022.

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.

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.

K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, 2013.

Appendix

A 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 and it has a convergence guarantee to the Pareto front. SMS-EGO (Ponweiser *et al.* (2008)) uses lower-confidence bound of each objective to calculate hypervolume improvement and EHVI (Emmerich *et al.* (2011)) uses expected hypervolume 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 it outperformed aforementioned methods in expensive MO-HPO settings (Ozaki *et al.* (2020, 2022)).

Evolutionary algorithm (EA) is another domain where MO optimization is actively studied. In EA, we 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 in the sense that both SAEAs and BO optimize a surrogate, which is an AF in BO and the surrogate in EA, at each iteration.

Meta-learning (Vanschoren (2019)) attracts attention to accelerate optimizations. In the context of HPO, Most meta-learning methods can be classified into either of the following four types: (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) linear combination of models trained on each task (Wistuba *et al.* (2016); Feurer *et al.* (2018)), and (4) training of a model jointly with meta-tasks (Swersky *et al.* (2013); Springenberg *et al.* (2016); Perrone *et al.* (2018); Salinas *et al.* (2020)). The warm-starting is designed to help especially at the early stage of optimizations; however, it does not provide the knowledge from metadata afterwards. The search space reduction could be also applied to any methods although those methods assume that users have a diverse set of meta-tasks and it is likely to miss some promising regions due to the reduction. 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 and our method is viewed as Type (4) (and is close to Type (3) as well). Since TPE relies on KDE, no joint training scheme has been developed so far.

B Background

In this paper, we consistently use the Lebesgue integral $\int_{\boldsymbol{x}\in\mathcal{X}} g(\boldsymbol{x})\mu(d\boldsymbol{x})$ instead of the Riemann integral $\int_{\boldsymbol{x}\in\mathcal{X}} g(\boldsymbol{x})d\boldsymbol{x}$; however, $\int_{\boldsymbol{x}\in\mathcal{X}} g(\boldsymbol{x})d\boldsymbol{x} = \int_{\boldsymbol{x}\in\mathcal{X}} g(\boldsymbol{x})\mu(d\boldsymbol{x})$ holds if $g(\boldsymbol{x})$ is Riemann integrable. The only reason why we use the Lebesgue integral is that some functions in our discussion cannot be handled by the Riemann integral and thus we encourage readers to replace $\mu(d\boldsymbol{x})$ with $d\boldsymbol{x}$ if they are not familiar to the Lebesgue integral.

Since all definitions can easily be expanded to the MO case, we discuss the single objective case for simplicity and mention how to expand to the MO case in the end of Section 2. Throughout this paper, we use the following terms for the discussion later (see Figure 1 to get the insight):

Definition 1 (γ-quantile value) *Given a quantile $\gamma \in (0, 1]$ and a measurable function $f : \mathcal{X} \rightarrow \mathbb{R}$, γ-quantile value $f^\gamma \in \mathbb{R}$ is a real number such that*

$$\gamma = \int_{\boldsymbol{x}\in\mathcal{X}} \mathbb{1}[f(\boldsymbol{x}) \leq f^\gamma] \frac{\mu(d\boldsymbol{x})}{\mu(\mathcal{X})}. \tag{4}$$

Definition 2 (γ-set) *Given a quantile $\gamma \in (0, 1]$, γ-set $\mathcal{X}^\gamma$ is defined as $\mathcal{X}^\gamma \coloneqq \{\boldsymbol{x} | \boldsymbol{x} \in \mathcal{X}, f(\boldsymbol{x}) \leq f^\gamma\} \in \mathcal{B}_\mathcal{X}$ where $\mathcal{B}_\mathcal{X} \coloneqq \mathcal{B}_D \cap \mathcal{X}$ is the Borel body over $\mathcal{X}$.*

Definition 3 (γ-set similarity) *Given two objective functions f_1, f_2 and $\gamma \in (0, 1]$, let the γ-sets for f_1, f_2 be $\mathcal{X}_1^\gamma, \mathcal{X}_2^\gamma$. Then we define γ-set similarity $s : \mathcal{B}_\mathcal{X} \times \mathcal{B}_\mathcal{X} \rightarrow [0, 1]$ between f_1 and f_2 as* *follows:*

$$s(\mathcal{X}_1^\gamma, \mathcal{X}_2^\gamma) = \frac{\mu(\mathcal{X}_1^\gamma \cap \mathcal{X}_2^\gamma)}{\mu(\mathcal{X}_1^\gamma \cup \mathcal{X}_2^\gamma)}. \tag{5}$$

Note that those definitions rely on some assumptions described in Appendix D.1.

B.1 Bayesian optimization (BO)

Suppose we would like to **minimize** a loss metric $f(\boldsymbol{x})$, then the formulation of HPO is defined as follows:

$$\boldsymbol{x}_{\mathrm{opt}} \in \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmin}} f(\boldsymbol{x}). \tag{6}$$

In Bayesian optimization (BO) (Brochu *et al.* (2010); Shahriari *et al.* (2016); Garnett (2022)), we assume that $f(\boldsymbol{x})$ is expensive and 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)):

$$\mathrm{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}] = \int_{-\infty}^{f^\star} (f^\star - f) p(f|\boldsymbol{x}, \mathcal{D}) df. \tag{7}$$

BO proposes the next configuration to evaluate via the optimization of the AF.

B.2 Tree-structured Parzen Estimator (TPE)

TPE (Bergstra *et al.* (2011, 2013)) is a variant of BO methods and it uses the expected improvement. To transform Eq. (7), we define the following:

$$p(\boldsymbol{x}|f, \mathcal{D}) := \left\{ \begin{array}{ll} p(\boldsymbol{x}|\mathcal{D}^{(l)}) & (f \leq f^\gamma) \\ p(\boldsymbol{x}|\mathcal{D}^{(g)}) & (f > f^\gamma) \end{array} \right. \tag{8}$$

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 top-$\lceil \gamma |\mathcal{D}| \rceil$ observation in $\mathcal{D}$. Note that $p(\boldsymbol{x}|\mathcal{D}^{(l)}), p(\boldsymbol{x}|\mathcal{D}^{(g)})$ are built by KDE. Combining Eqs. (7), (8) and Bayes' theorem, the AF of TPE is computed as (Bergstra *et al.* (2011)):

$$\mathrm{EI}_{f^\gamma}[\boldsymbol{x}|\mathcal{D}] \propto \left(\gamma + (1 - \gamma) \frac{p(\boldsymbol{x}|\mathcal{D}^{(g)})}{p(\boldsymbol{x}|\mathcal{D}^{(l)})} \right)^{-1}. \tag{9}$$

Since the density ratio $p(\boldsymbol{x}|\mathcal{D}^{(l)})/p(\boldsymbol{x}|\mathcal{D}^{(g)})$ preserves the order, TPE uses the density ratio as an AF instead of the RHS of Eq. (9). 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.

B.3 Multi-objective TPE (MO-TPE)

MO-TPE (Ozaki *et al.* (2020, 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 hypervolume subset selection problem (HSSP) (Bader and Zitzler (2011)) to obtain $\mathcal{D}^{(l)}$. HSSP is a method to tie-break configurations with the same non-domination rank based on the hypervolume contribution. MO-TPE with HSSP 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 the hypervolume calculation, which can be highly expensive.

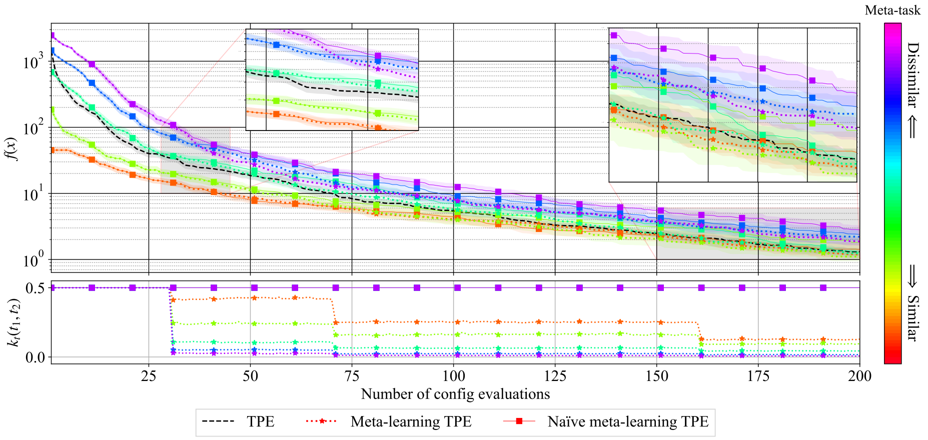

Figure 4: The comparison of the convergence of TPE and meta-learning TPE based on the task similarity. $c^\star = 0$ is identical to the target task and tasks become dissimilar as $c^\star$ becomes larger. Each line except the black line is the performance curves of meta-learning TPE on differently similar tasks. 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. The bottom figure shows the medians of the task weight on the meta-task.

C The details of Meta-learning TPE

C.1 The details of the task-conditioned acquisition function

To begin with, we explain the transformation of the original TPE:

$$\begin{aligned}
\mathrm{EI}_{f^\gamma}[\boldsymbol{x}|\mathcal{D}] &= \int_{-\infty}^{f^\gamma} (f^\gamma - f) p(f|\boldsymbol{x}) df \\
&= \int_{-\infty}^{f^\gamma} (f^\gamma - f) \frac{p(\boldsymbol{x}|f, \mathcal{D}) p(f|\mathcal{D})}{p(\boldsymbol{x}|\mathcal{D})} df \ (\because \mathrm{Bayes' \ theorem}) \\
&= \frac{p(\boldsymbol{x}|\mathcal{D}^{(l)})}{p(\boldsymbol{x}|\mathcal{D})} \underbrace{\int_{-\infty}^{f^\gamma} (f^\gamma - f) p(f|\mathcal{D}) df}_{\mathrm{const \ w.r.t.} \ \boldsymbol{x}} \ (\because \mathrm{Eq.} \ (8)) \\
&\propto \frac{p(\boldsymbol{x}|\mathcal{D}^{(l)})}{\gamma p(\boldsymbol{x}|\mathcal{D}^{(l)}) + (1-\gamma) p(\boldsymbol{x}|\mathcal{D}^{(g)})} \left(\because p(\boldsymbol{x}|\mathcal{D}) = \int_{-\infty}^{\infty} p(\boldsymbol{x}|f, \mathcal{D}) p(f|\mathcal{D}) df \right).
\end{aligned} \tag{10}$$

In the same vein, we transform the task-conditioned AF:

$$\begin{aligned}
\mathrm{EI}_{f^\gamma}[\boldsymbol{x}|t, \boldsymbol{D}] &= \int_{-\infty}^{f^\gamma} (f^\gamma - f) p(f|\boldsymbol{x}, t, \boldsymbol{D}) df \\
&= \int_{-\infty}^{f^\gamma} (f^\gamma - f) \frac{p(\boldsymbol{x}, t|f, \boldsymbol{D}) p(f|\boldsymbol{D})}{p(\boldsymbol{x}, t|\boldsymbol{D})} df \ (\because \mathrm{Bayes' \ theorem}).
\end{aligned} \tag{11}$$

To further transform, we assume the conditional shift (Zhang *et al.* (2013)). In the conditional shift, we assume that all tasks have the same target distribution, i.e. $p(f|t_i, \mathcal{D}_i) = p(f|t_j, \mathcal{D}_j)$ (see Figure 5). Since TPE considers ranks of observations and ignore the scale, the optimization of $f(\boldsymbol{x})$ is equivalent to that of the quantile of $f(\boldsymbol{x})$. For this reason, the only requirement to achieve the conditional shift is to use the same quantile γ for the split of observations $\mathcal{D}_m$ of all meta-tasks. In Figure 5, we visualize two functions and their quantile functions along with their target distributions $p(f)$. As seen in the figure, the quantile functions equalize the target distribution while preserving the order of each point. Let f be already the quantile function for convenience and we define:

$$p(\boldsymbol{x}, t|f, \boldsymbol{D}) := \left\{ \begin{array}{ll} p(\boldsymbol{x}, t|\boldsymbol{D}^{(l)}) & (f \leq \gamma) \\ p(\boldsymbol{x}, t|\boldsymbol{D}^{(g)}) & (f > \gamma) \end{array} \right. \tag{12}$$

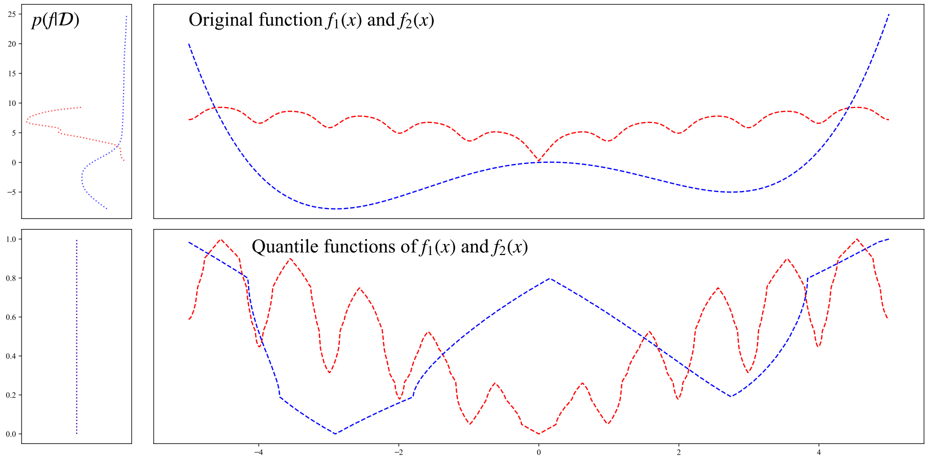

Figure 5: Examples of the quantile function. Left figures show shapes of functions and right figures show the distribution of each function value. Different functions usually do not have the same target distribution (top left); however, if we consider the quantile of functions, their target distributions match (bottom left).

where $\boldsymbol{D}^{(l)} := \{\mathcal{D}_m^{(l)}\}_{m=1}^T$ is a set of the γ-quantile observations and the conditional shift plays a role to enable us to have a unique split point, which is at $f = \gamma$. Then we can further transform as follows:

$$\begin{aligned}\int_{-\infty}^{f^\gamma}(f^\gamma - f)\frac{p(\boldsymbol{x}, t|f, \boldsymbol{D})p(f|\boldsymbol{D})}{p(\boldsymbol{x}, t|\boldsymbol{D})}df &= \int_0^\gamma(\gamma - f)\frac{p(\boldsymbol{x}, t|f, \boldsymbol{D})p(f|\boldsymbol{D})}{p(\boldsymbol{x}, t|\boldsymbol{D})}df \\ &= \frac{p(\boldsymbol{x}, t|\boldsymbol{D}^{(l)})}{p(\boldsymbol{x}, t|\boldsymbol{D})}\underbrace{\int_0^\gamma p(f|\boldsymbol{D})df}_{\text{const w.r.t. }\boldsymbol{x}} \\ &\propto \frac{p(\boldsymbol{x}, t|\boldsymbol{D}^{(l)})}{\gamma p(\boldsymbol{x}, t|\boldsymbol{D}^{(l)}) + (1-\gamma)p(\boldsymbol{x}, t|\boldsymbol{D}^{(g)})}.\end{aligned} \tag{13}$$

As seen in the equations above, TPE always ignores the improvement term $f^\gamma - f$ and thus we can actually view the AF as probability of improvement (PI). Then the conditional shift can be explained in a more straightforward way. Since PI requires the binary classification of $\mathcal{D}$ into $\mathcal{D}^{(l)}$ or $\mathcal{D}^{(g)}$, $p(f|\mathcal{D})$ just depends on our split. More specifically, if we regard f as $\mathbb{1}[f \leq f^\gamma]$ and then we simply get $p(f|\mathcal{D})$ as $p(f = 1|\mathcal{D}) = |\mathcal{D}^{(l)}|/|\mathcal{D}| = \gamma$ and $p(f = 0|\mathcal{D}) = |\mathcal{D}^{(g)}|/|\mathcal{D}| = 1 - \gamma$. For this reason, the assumption is easily satisfied as long as we fix γ for all tasks. Note that although we can guarantee the conditional shift, as TPE usually performs better than random search and thus the γ-quantile in observations is better than the true γ-quantile value f^γ with a high probability. It implies that the quality of knowledge about well-performing regions will be relatively low in meta-tasks if we have much more observations on the target task and our method might not be accelerated drastically. We design our task similarity to tackle the knowledge dillution problem.

C.2 The details of the task similarity

Since the γ-set similarity is intractable due to the unavailability of the true $\mathcal{X}^\gamma$, we describe how to estimate the γ-set similarity. First, we define the measure space $(\Omega, \mathcal{F}, \mu)$ such that $\Omega := \prod_{k=1}^\infty \mathcal{X}$ and $\mathcal{F}$ is the minimum σ-algebra on the cylinder set of $\mathcal{B}_\mathcal{X}$. Then, we prove the following theorem:

Theorem 1 *Given a measure space $(\Omega, \mathcal{F}, \mu)$ and a strongly-consistent estimator of the γ-set PDF $p : \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$, then the following approximation almost surely converges to the γ-set similarity:*

$$\hat{s}(\mathcal{D}_i^{(l)}, \mathcal{D}_j^{(l)}) := \frac{1 - d_{\mathrm{tv}}(p_i, p_j)}{1 + d_{\mathrm{tv}}(p_i, p_j)} \overset{a.s.}{\rightarrow} s(\mathcal{X}_i^\gamma, \mathcal{X}_j^\gamma) \tag{14}$$

where $\overset{a.s.}{\to}$ implies almost sure convergence and d_{tv} is the total variation distance:

$$d_{\mathrm{tv}}(p_i, p_j) \coloneqq \frac{1}{2}\int_{\boldsymbol{x}\in\mathcal{X}} |p(\boldsymbol{x}|\mathcal{D}_i^{(l)}) - p(\boldsymbol{x}|\mathcal{D}_j^{(l)})|\mu(d\boldsymbol{x}). \tag{15}$$

The proof is provided in Appendix D.2.

To see the effect of the task similarity, we conduct an experiment using the ellipsoid function $f(\boldsymbol{x}|c) \coloneqq f(x_1, \dots, x_4|c) = \sum_{d=1}^{4} 5^{d-1}(x_d - c)^2$ defined on $[-5, 5]^4$. Along with the original TPE, we optimized the ellipsoid function $f(\boldsymbol{x}|c = 0)$ by meta-learning TPE using the randomly sampled 100 observations from $f(\boldsymbol{x}|c_\star)$ where $c^\star \in [0, 1, \dots, 4]$ (each run uses only one of $[0, 1, \dots, 4]$) 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 3 except we evaluated 200 configurations.

In Figure 4, 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, our method starts to recover the performance of TPE from that point (see the first inset figure) and our method showed closer performance 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, 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 not from i.i.d of the meta-learning TPE sampler, the true quantile of our method is likely to be better. It implies that the weight for meta-task is expected to decrease over time even if the target task is identical to meta-tasks.

C.3 When does meta-learning fail? – Solutions to the problems

Case I: The γ-set for the target task $\mathcal{D}_1^{(l)}$ does not approach $\mathcal{X}^\gamma$

From the assumption of Theorem 1, $\mathcal{D}_1^{(l)}$ 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 as discussed in Appendix C.1, it does not guarantee that $\mathcal{D}_1^{(l)}$ goes to $\mathcal{X}^\gamma$ and it may even be guided towards non-γ-set regions. In this case, our task similarity measure not only obtains wrong approximation, but also causes poor solutions. To avoid the 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 2 *Suppose 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 at most countable search space $\mathcal{X}$ and we use a KDE such that its distribution converges to the empirical distribution as the number of samples goes to infinity, then the top-γ-quantile observations $\mathcal{D}_N^{(l)}$ is almost surely a subset of $\mathcal{X}^\gamma$.*

The proof is provided in Appendix D.3. 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 when we have a sufficiently large number of observations and thus the task similarity is correctly or pessimistically estimated. Notice that we use the bandwidth selection used in Falkner *et al.* (2018) and it satisfies the assumption about the KDE in Theorem 2.

Case II: The dimension of the search space D is high

When D is high, the approximation of the γ-set similarity is easily biased. For example, when we consider $f(\boldsymbol{x}) = \|R\boldsymbol{x}\|_1$ where $\boldsymbol{x} \in [-1/2, 1/2]^D$ and $R \in \mathbb{R}^{D\times D}$ is the rotation matrix to

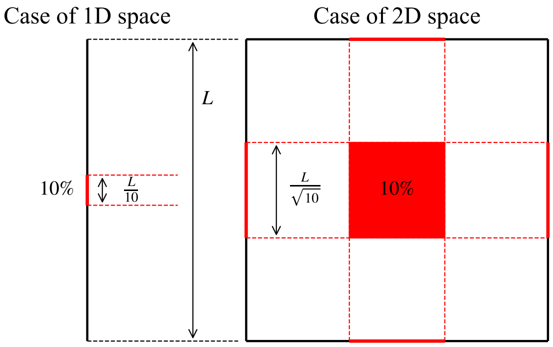

Figure 6: The conceptual visualization where the top-10% domain for each dimension becomes larger when the importance of each dimension is same and there is no interaction between dimensions. The thick black lines are the edges of each domain and the red lines are the important domains for each dimension. The red lines become longer as the dimension becomes higher.

obtain the example in Figure 6, the γ-set is $\mathcal{X}^{\gamma} = [-\gamma^{1/D}/2, \gamma^{1/D}/2]$. As $\lim_{D \rightarrow \infty} \gamma^{1/D} = 1$, the marginal γ-set PDF for each dimension approaches the uniform PDF. However, the marginal γ-set PDF $p_d(x_d|\mathcal{D}^{(l)})$ 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 according to Bergstra and Bengio (2012). It implies 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:

Proposition 1 *If some dimensions are trivial on two tasks, the γ-set similarity between those tasks is identical irrespective of with or without those dimensions.*

The formal definition of the trivial dimensions and the proof are provided in Appendix D.4. 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 major methods such as principle component analysis and singular value decomposition mix up categorical and numerical parameters. In this paper, we use Pearson divergence based f-ANOVA (Watanabe and Hutter (2022)) and it computes HPI for each dimension via the 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 for each dimension and take the average of HPI:

$$
\begin{aligned}
\mathbb{V}_{d,m} &\coloneqq \gamma^2 \int_{x_d \in \mathcal{X}_d} \left(\frac{p_d(x_d|\mathcal{D}_m^{(l)})}{1/\mu(\mathcal{X}_d)} - 1 \right)^2 \frac{\mu(dx_d)}{\mu(\mathcal{X}_d)} \\
\bar{\mathbb{V}}_d &\coloneqq \frac{1}{T} \sum_{m=1}^{T} \mathbb{V}_{d,m}
\end{aligned}
\tag{16}
$$

where the first equation is Eq (8) in Watanabe and Hutter (2022). 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 $S \in 2^{\{1,\dots,D\}}$. While we compute the original KDE via:

$$
p(\boldsymbol{x}|\mathcal{D}') = \frac{1}{N} \sum_{n=1}^{N} \prod_{d=1}^{D} k_d(x_d, x_{d,n}) \tag{17}
$$

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

$$
p^{\mathrm{DR}}(\boldsymbol{x}|\mathcal{D}') = \frac{1}{N} \sum_{n=1}^{N} \prod_{d \in S} k_d(x_d, x_{d,n}). \tag{18}
$$

D Proofs

D.1 Assumptions

In this paper, we assume the following:

1. Objective function $f : \mathcal{X} \rightarrow \mathbb{R}$ is Lebesgue integrable and measurable defined over the compact convex measurable subset $\mathcal{X} \subseteq \mathbb{R}^D$,
2. The probability measure $\mathbb{1}[\boldsymbol{x} \in \mathcal{X}^\gamma]$ is Riemann integrable over $\mathcal{X}$,
3. The PDF of the probability measure $\mathbb{1}[\boldsymbol{x} \in \mathcal{X}^\gamma]$ always exists, and
4. The target PDFs are identical, i.e. $p(f|t_i) = p(f|t_j)$ for all task pairs, while the conditional PDFs can be different, i.e. $p(\boldsymbol{x}|f, t_i) \neq p(\boldsymbol{x}|f, t_j)$ where t_i for $i \in [1, T] \cap \mathbb{N}$ is the symbol for the i-th task, respectively.

Strictly speaking, we cannot guarantee that the PDF of the probability measure $\mathbb{1}[\boldsymbol{x} \in \mathcal{X}^\gamma]$ always exists; however, we formally assume that the PDF exists by considering the formal derivative of the step function as the Dirac delta function. The last assumption is known as **conditional shift** (Zhang *et al.* (2013)). Note that a categorical parameter with K categories is handled as $\mathcal{X}_i = [1, K]$ as in the TPE implementation (Bergstra *et al.* (2011)) as we do not require the continuity of f with respect to hyperparameters in our theoretical analysis, this definition is valid as long as the employed kernel for categorical parameters treats different categories to be equally similar such as Aitchison Kernel proposed by Aitchison and Aitken (1976). In this definition, $x, x' \in \mathcal{X}_i$ are viewed as equivalent as long as $\lfloor x \rfloor = \lfloor x' \rfloor$ and it leads to the random sampling of each category to be uniform and the Lebesgue measure of $\mathcal{X}$ to be non-zero.

D.2 Proof of Theorem 1

Throughout this section, we denote $\mathcal{D}^{(l)}_{m,N}$ as the $\mathcal{D}^{(l)}_m$ with the size of N for notational clarity and we define the measure space $(\Omega, \mathcal{F}, \mu)$ such that $\Omega := \prod_{k=1}^{\infty} \mathcal{X}$ and $\mathcal{F}$ is the minimum σ-algebra on the cylinder set of $\mathcal{B}_{\mathcal{X}}$. Then we first prove the following lemma:

Lemma 1 *The γ-set similarity between two functions f_i, f_j is computed as:*

$$s(\mathcal{X}_i^\gamma, \mathcal{X}_j^\gamma) = \frac{1 - d_{\mathrm{tv}}(p_i, p_j)}{1 + d_{\mathrm{tv}}(p_i, p_j)}. \tag{19}$$

where $P_i = \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_i^\gamma], P_j = \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_j^\gamma]$ are the probability measure of $p_i = p(\boldsymbol{x}|\mathcal{X}_i^\gamma), p_j = p(\boldsymbol{x}|\mathcal{X}_j^\gamma)$ and

$$d_{\mathrm{tv}}(p_i, p_j) = \frac{1}{2}\|P_i - P_j\|_1 = \frac{1}{2}\int_{\boldsymbol{x} \in \mathcal{X}} |p(\boldsymbol{x}|\mathcal{X}_i^\gamma) - p(\boldsymbol{x}|\mathcal{X}_j^\gamma)|\mu(d\boldsymbol{x}) \tag{20}$$

is the total variation distance.

Note that we used the Scheffe's lemma (Tsybakov (2008)) for the transformation in Eq. (20).

Proof 1 *Since $\boldsymbol{x}$ is either in or not in $\mathcal{X}^\gamma$ and $\mathcal{X}^\gamma$ is fixed, the probability measure takes either* 0 *or* 1. *For this reason, the following holds:*

$$\begin{aligned} \mathbb{1}[\boldsymbol{x} \in \mathcal{X}_i^\gamma \cap \mathcal{X}_j^\gamma] &= \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_i^\gamma \cap \mathcal{X}_j^\gamma] = \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_i^\gamma] \wedge \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_j^\gamma], \\ \mathbb{1}[\boldsymbol{x} \in \mathcal{X}_i^\gamma \cup \mathcal{X}_j^\gamma] &= \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_i^\gamma \cup \mathcal{X}_j^\gamma] = \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_i^\gamma] \vee \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_j^\gamma] \end{aligned} \tag{21}$$

where $\wedge, \vee$ are equivalent to the min and max operator, respectively. It leads to the following equations:

$$\begin{aligned} \mu(\mathcal{X}_i^\gamma \cap \mathcal{X}_j^\gamma) &= \int_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_i^\gamma \cap \mathcal{X}_j^\gamma]\mu(d\boldsymbol{x}), \\ \mu(\mathcal{X}_i^\gamma \cup \mathcal{X}_j^\gamma) &= \int_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_i^\gamma \cup \mathcal{X}_j^\gamma]\mu(d\boldsymbol{x}). \end{aligned} \tag{22}$$

Using Eqs. (21), (22), we obtain the following equation:

$$s(\mathcal{X}_i^\gamma, \mathcal{X}_j^\gamma) = \frac{\int_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_i^\gamma] \wedge \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_j^\gamma] \mu(d\boldsymbol{x})}{\int_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_i^\gamma] \vee \mathbb{P}[\boldsymbol{x} \in \mathcal{X}_j^\gamma] \mu(d\boldsymbol{x})} = \frac{1 - d_{\mathrm{tv}}(p_i, p_j)}{1 + d_{\mathrm{tv}}(p_i, p_j)}. \tag{23}$$

This completes the proof.

Then we need to prove the following lemma as well:

Lemma 2 *Given the assumptions above and the measure space $(\Omega, \mathcal{F}, \mu)$, the following holds under the assumption (denote Assumption $(\star)$) of $\exists \epsilon > 0, \lim_{N \to \infty} \|(p(\boldsymbol{x}|\mathcal{D}_{m,N}^{(l)}) - \mathbb{1}[\boldsymbol{x} \in \mathcal{X}_m^\gamma]/\mu(\mathcal{X}_m^\gamma))\mathbb{1}[\boldsymbol{x} \in \epsilon \boldsymbol{B}_N]\|_\infty = 0$ for all m where $\boldsymbol{B}_N \subseteq \mathbb{R}^D$ is a ball with radius N centered at $\mathbf{0}$:*

$$\int_{\boldsymbol{x} \in \mathcal{X}} |p(\boldsymbol{x}|\mathcal{D}_{i,N}^{(l)}) - p(\boldsymbol{x}|\mathcal{D}_{j,N}^{(l)})| \mu(d\boldsymbol{x}) \overset{a.s.}{\to} \int_{\boldsymbol{x} \in \mathcal{X}} \left| \frac{\mathbb{1}[\boldsymbol{x} \in \mathcal{X}_i^\gamma]}{\mu(\mathcal{X}_i^\gamma)} - \frac{\mathbb{1}[\boldsymbol{x} \in \mathcal{X}_j^\gamma]}{\mu(\mathcal{X}_j^\gamma)} \right| \mu(d\boldsymbol{x}) \; (N \to \infty). \tag{24}$$

Proof 2 *From the assumption of the strongly-consistent estimator, the following uniform convergence holds (Wied and Weißbach (2012)) for all $m \in \{1, \dots, T\}$ and $\forall \epsilon > 0$ at each point of continuity $\boldsymbol{x}$ of $\mathbb{1}[\boldsymbol{x} \in \mathcal{X}_m^\gamma]$:*

$$p(\boldsymbol{x}|\mathcal{D}_{m,N}^{(l)}) \overset{a.s.}{\to} \frac{\mathbb{1}[\boldsymbol{x} \in \mathcal{X}_m^\gamma]}{\mu(\mathcal{X}_m^\gamma)} \; (N \to \infty). \tag{25}$$

Note that the uniform convergence almost surely happens with respect to an arbitrary sample from Ω. Thus the following holds using the continuous mapping theorem:

$$|p(\boldsymbol{x}|\mathcal{D}_{i,N}^{(l)}) - p(\boldsymbol{x}|\mathcal{D}_{j,N}^{(l)})| \overset{a.s.}{\to} \left| \frac{\mathbb{1}[\boldsymbol{x} \in \mathcal{X}_i^\gamma]}{\mu(\mathcal{X}_i^\gamma)} - \frac{\mathbb{1}[\boldsymbol{x} \in \mathcal{X}_j^\gamma]}{\mu(\mathcal{X}_j^\gamma)} \right| \; (N \to \infty). \tag{26}$$

From the assumptions, all functions are defined over the compact convex measurable subset $\mathcal{X} \subseteq \mathbb{R}^D$ and all convergences in this proof are uniform and almost sure; therefore, we obtain the following result based on Theorem 7.16 (Rudin and others (1976)):

$$\begin{aligned} \lim_{N \to \infty} \int_{\boldsymbol{x} \in \mathcal{X}} |p(\boldsymbol{x}|\mathcal{D}_{i,N}^{(l)}) - p(\boldsymbol{x}|\mathcal{D}_{j,N}^{(l)})| \mu(d\boldsymbol{x}) &= \int_{\boldsymbol{x} \in \mathcal{X}} \lim_{N \to \infty} |p(\boldsymbol{x}|\mathcal{D}_{i,N}^{(l)}) - p(\boldsymbol{x}|\mathcal{D}_{j,N}^{(l)})| \mu(d\boldsymbol{x}) \\ &\overset{a.s.}{\to} \int_{\boldsymbol{x} \in \mathcal{X}} \left| \frac{\mathbb{1}[\boldsymbol{x} \in \mathcal{X}_i^\gamma]}{\mu(\mathcal{X}_i^\gamma)} - \frac{\mathbb{1}[\boldsymbol{x} \in \mathcal{X}_j^\gamma]}{\mu(\mathcal{X}_j^\gamma)} \right| \mu(d\boldsymbol{x}). \end{aligned} \tag{27}$$

Note that the last transformation requires Assumption $(\star)$ to be true with the probability of 1. This completes the proof.

From Eq. (23) in Lemma 1, Eq. (24) in Lemma 2 and the continuous mapping theorem, the statement of Theorem 1 is proved and this completes the proof.

D.3 Proof of Theorem 2

Theorem 3 *Suppose 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 at most countable search space $\mathcal{X}$ and we use a KDE such that its distribution converges to the empirical distribution as the number of samples goes to infinity, then the top-γ-quantile observations $\mathcal{D}_N^{(l)}$ is almost surely a subset of $\mathcal{X}^\gamma$.*

Proof 3 *First, we define a function defined on at most countable set as $f_k(\boldsymbol{x}) = f(k\lfloor \boldsymbol{x}/k \rfloor)$ where $f(\boldsymbol{x})$ so that we can still define the PDF on $\mathcal{X}$. In this proof, we assume $\mathcal{X} = [0, 1]^D$ and we choose the next configuration from $\{0, 1/k, \dots, (k-1)/k, 1\}^D$. We first prove the finite case and then take $k \to \infty$ to prove the countable case. First, we list the facts that are immediately drawn from the assumptions:*

1. *The KDE with a set of control parameters $\boldsymbol{\alpha}_N$ at the N-th iteration converges to the empirical distribution, i.e.*

$$\lim_{N \to \infty} \frac{1}{N} \sum_{n'=1}^{N} k(\boldsymbol{x}, \boldsymbol{x}_{n'} | \boldsymbol{\alpha}_N) \to \lim_{N \to \infty} \frac{1}{N} \sum_{n'=1}^{N} \delta(\boldsymbol{x}, \boldsymbol{x}_{n'}), \tag{28}$$

2. *Due to the random part in the ϵ-greedy policy, we almost surely covers all possible configurations $\boldsymbol{x} \in \mathcal{X}$ with a sufficiently large number of samples,*

3. $\forall \boldsymbol{x} \in \mathcal{D}_N^{(l)}, p(\boldsymbol{x}|\mathcal{D}_N^{(l)}) > 0$,

4. $\forall \epsilon' > 0, \exists N > 0, \forall \boldsymbol{x} \notin \mathcal{D}_N^{(l)}, p(\boldsymbol{x}|\mathcal{D}_N^{(l)}) < \epsilon'$,

5. $\forall \boldsymbol{x} \in \mathcal{D}_N^{(g)}, p(\boldsymbol{x}|\mathcal{D}_N^{(g)}) > 0$, *and*

6. $\forall \epsilon' > 0, \exists N > 0, \forall \boldsymbol{x} \notin \mathcal{D}_N^{(g)}, p(\boldsymbol{x}|\mathcal{D}_N^{(g)}) < \epsilon'$.

Since $\forall \epsilon' > 0, \exists N > 0, \mathcal{D}_N^{(l)} \overset{a.s.}{\subseteq} \mathcal{X}^{\gamma + \epsilon'}$ *holds from the weak law of large numbers if* $\epsilon = 1$*, the main strategy of this proof is to show "**the probability of drawing** $\boldsymbol{x} \in \mathcal{X}^\gamma$ **is higher than** γ **as** N **goes to infinity**". We use the following fact in the AF of the TPE formulation:*

$$\mathrm{EI}_{f^\gamma}[\boldsymbol{x}] \propto \mathrm{PI}_{f^\gamma}[\boldsymbol{x}] = \frac{\gamma p(\boldsymbol{x}|\mathcal{D}^{(l)})}{\gamma p(\boldsymbol{x}|\mathcal{D}^{(l)}) + (1-\gamma) p(\boldsymbol{x}|\mathcal{D}^{(g)})}. \tag{29}$$

Since we use the ϵ-greedy algorithm, $\lim_{N \to \infty} p(\boldsymbol{x}|\mathcal{D}_N^{(l)}) \overset{a.s.}{\geq} \epsilon/(k+1)^D$ *and* $\lim_{N \to \infty} p(\boldsymbol{x}|\mathcal{D}_N^{(g)}) \overset{a.s.}{\geq} \epsilon/(k+1)^D$ *hold for all* $\boldsymbol{x} \in \mathcal{X}$*. It implies that* $p(\boldsymbol{x}|\mathcal{D}_N^{(l)})$ *and* $p(\boldsymbol{x}|\mathcal{D}_N^{(g)})$ *dominate each other if* $\boldsymbol{x} \notin \mathcal{D}_N^{(g)}$ *or* $\boldsymbol{x} \notin \mathcal{D}_N^{(l)}$*, respectively. Thus, when we take a sufficiently large N, the following holds:*

1. $\forall \epsilon', \exists N > 0, \forall \boldsymbol{x} \notin \mathcal{D}_N^{(g)}, |\mathrm{PI}_{f^\gamma}[\boldsymbol{x}] - 1| < \epsilon'$, *and*

2. $\forall \epsilon', \exists N > 0, \forall \boldsymbol{x} \notin \mathcal{D}_N^{(l)}, \mathrm{PI}_{f^\gamma}[\boldsymbol{x}] < \epsilon'$.

Notice that we cannot determine $\mathrm{PI}_{f^\gamma}[\boldsymbol{x}]$ *for* $\boldsymbol{x} \in \mathcal{D}_N^{(l)} \cap \mathcal{D}_N^{(g)}$ *and we assume that a sufficiently large N allows TPE to cover all possible configurations at least once. For this reason, the greedy algorithm always picks a configuration* $\boldsymbol{x} \in \mathcal{D}_N^{(l)}$*. Since* $\mathcal{D}_N^{(l)}$ *is determined based on the objective function* f_k *and all possible configurations are covered due to the random policy, either of the following is satisfied:*

1. $\mathcal{D}_N^{(l)} \subseteq \mathcal{X}^{\gamma'}$ *such that* $\gamma' \leq \gamma$*, or*

2. $\mathcal{X}^\gamma \subseteq \mathcal{D}_N^{(l)}$.

For the first case, we obtain $\boldsymbol{x} \in \mathcal{X}^\gamma$ *with the probability of* 1 *and we obtain* $\boldsymbol{x} \in \mathcal{X}^\gamma$ *with the probability of* $\gamma(k+1)^D / |\mathrm{set}(\mathcal{D}_N^{(l)})| > \gamma$ *for the second case. Since the random policy obviously picks a configuration* $\boldsymbol{x} \in \mathcal{X}^\gamma$ *with the probability of* γ*, the probability of drawing* $\boldsymbol{x} \in \mathcal{X}^\gamma$ *is larger than* $\epsilon\gamma + (1-\epsilon)\min(1, \gamma(k+1)^D / |\mathrm{set}(\mathcal{D}_N^{(l)})|) > \gamma$*; therefore, it concludes that* $\mathcal{D}_N^{(l)} \overset{a.s.}{\subseteq} \mathcal{X}^\gamma$ *and this completes the proof for the finite case. The countable case is completed by* $n \to \infty$*.*

Using this lemma, we prove the theorem of interest.

Proof 4 *Let* $\{\mathcal{D}_m\}_{m=1}^T$ *with* $|\mathcal{D}_1| = N$ *be* $\boldsymbol{D}_N$ *and the splits of* $\boldsymbol{D}_N$ *be* $\boldsymbol{D}_N^{(l)}$ *and* $\boldsymbol{D}_N^{(g)}$*. We also define the observations of the target task with the size of N as* $\mathcal{D}_N$ *for the simplicity. Notice that we assume that N is sufficiently large to cover all possible configurations as in the proof of Theorem 3. Due to the fact that the limit of the joint PDF converges to the original PDF, i.e.*

$\lim_{N\to\infty} p(\boldsymbol{x}, t|\boldsymbol{D}_N^{(l)}) = p(\boldsymbol{x}|\mathcal{D}_N^{(l)})$ *and* $\lim_{N\to\infty} p(\boldsymbol{x}, t|\boldsymbol{D}_N^{(g)}) = p(\boldsymbol{x}|\mathcal{D}_N^{(g)})$*, the following holds:*

$$\begin{aligned}
&\text{Property (1). } \exists N > 0, \forall \boldsymbol{x} \in \mathcal{D}_N^{(l)}, p(\boldsymbol{x}, t|\boldsymbol{D}_N^{(l)}) > 0,\\
&\text{Property (2). } \forall \epsilon' > 0, \exists N > 0, \forall \boldsymbol{x} \in \mathcal{D}_N^{(l)} \setminus \mathcal{D}_N^{(g)}, p(\boldsymbol{x}, t|\boldsymbol{D}_N^{(g)}) < \epsilon',\\
&\text{Property (3). } \exists N > 0, \forall \boldsymbol{x} \in \mathcal{D}_N^{(g)}, p(\boldsymbol{x}, t|\boldsymbol{D}_N^{(g)}) > 0,\\
&\text{Property (4). } \forall \epsilon' > 0, \exists N > 0, \forall \boldsymbol{x} \in \mathcal{D}_N^{(g)} \setminus \mathcal{D}_N^{(l)}, p(\boldsymbol{x}, t|\boldsymbol{D}_N^{(l)}) < \epsilon'.
\end{aligned}$$

From Properties (1) and (2), the AF converges to 1 if $\boldsymbol{x} \in \mathcal{D}_N^{(l)} \setminus \mathcal{D}_N^{(g)}, N \to \infty$. *From Properties (3) and (4), the AF converges to 0 if* $\boldsymbol{x} \in \mathcal{D}_N^{(g)} \setminus \mathcal{D}_N^{(l)}, N \to \infty$. *Furthermore, the AF may take non-zero if* $\forall \boldsymbol{x} \in \mathcal{D}_N^{(l)} \cap \mathcal{D}_N^{(g)}$. *Since those conditions are the same for Theorem 3, the same discussion applies to this theorem and thus this completes the proof.*

D.4 Proof of Proposition 1

We first note that the definition of the Lebesgue measure μ changes based on the input for simplicity in this section. For example, when we take $A \in \mathcal{B}_D \cap \mathcal{X}$ as an input, μ is defined on $\mathcal{X}$ and when we take $A \in \mathcal{B}_s \cap \mathcal{X}_s$ as an input, μ is defined on $\mathcal{X}_s$.

First we formally define the trivial dimensions:

Definition 4 (Trivial dimension) *Given a quantile* $\gamma \in (0, 1]$ *and a measure space* $(\mathcal{X}, \mathcal{B}_{\mathcal{X}}, \mu)$*, the* d*-th dimension is trivial if and only if* $\mathbb{P}[\boldsymbol{x} \in \mathcal{X}^\gamma | \boldsymbol{x}_{-d}] = 1 \cup \mathbb{P}[\boldsymbol{x} \notin \mathcal{X}^\gamma | \boldsymbol{x}_{-d}] = 1$ *where* $\boldsymbol{x}_{-d}$ *is* $\boldsymbol{x}$ *without the* d*-th dimension,* $\mathcal{X}_d$ *is the domain of the* d*-th dimension, and the probability measure* $\mathbb{P}$ *is defined on* $(\mathcal{X}_d, \mathcal{B} \cap \mathcal{X}_d)$*,*

The definition implies the d-th dimension is trivial if and only if the binary function $b(\boldsymbol{x}|\mathcal{X}^\gamma) := \mathbb{1}[\boldsymbol{x} \in \mathcal{X}^\gamma]$ does not change almost everywhere (μ-a.e.) due to the variation in the d-th dimension when the other dimensions are fixed. Note that trivial dimension is a stronger concept than less important dimension in f-ANOVA. More specifically, f-ANOVA computes the marginal variance of each dimension and judges important dimensions based on the variance; however, zero variance, which implies the dimension is not important, does not guarantee that such dimension is trivial. For example, $f(x, y) = \mathrm{sign}(x) \sin y$ leads to zero variance in both dimensions, but both dimensions obviously change the function values.

We define $b(\boldsymbol{x}|\mathcal{X}^\gamma, \boldsymbol{x}_{-d})$ as $b(\boldsymbol{x}|\mathcal{X}^\gamma)$ such that each element, except for the d-th dimension, of $\boldsymbol{x}$ is fixed to $\boldsymbol{x}_{-d}$ and then we prove Proposition 1.

Proof 5 *From the definition of the trivial dimension, the following holds when we assume the* d*-th dimension is trivial:*

$$\int_{\boldsymbol{x}_{-d} \in \mathcal{X}_{-d}} b(\boldsymbol{x}|\mathcal{X}^\gamma, \boldsymbol{x}_{-d}) \mu(d\boldsymbol{x}_{-d}) = \frac{\mu(\mathcal{X}^\gamma)}{\mu(\mathcal{X}_d)} \ \mu\text{-a.e..} \tag{30}$$

Additionally, as the d*-th dimension of two tasks are trivial, we obtain the following for* $\forall \boldsymbol{x}_{-d} \in \mathcal{X}_{-d}$*:*

$$\begin{aligned}
b(\boldsymbol{x}|\mathcal{X}_1^\gamma \cap \mathcal{X}_2^\gamma, \boldsymbol{x}_{-d}) &= \min(b(\boldsymbol{x}|\mathcal{X}_1^\gamma, \boldsymbol{x}_{-d}), b(\boldsymbol{x}|\mathcal{X}_2^\gamma, \boldsymbol{x}_{-d})),\\
b(\boldsymbol{x}|\mathcal{X}_1^\gamma \cup \mathcal{X}_2^\gamma, \boldsymbol{x}_{-d}) &= \max(b(\boldsymbol{x}|\mathcal{X}_1^\gamma, \boldsymbol{x}_{-d}), b(\boldsymbol{x}|\mathcal{X}_2^\gamma, \boldsymbol{x}_{-d}))
\end{aligned} \tag{31}$$

where $\mathcal{X}_1^\gamma, \mathcal{X}_2^\gamma$ *are the* γ*-sets of tasks 1, 2. As both* $b(\boldsymbol{x}|\mathcal{X}_1^\gamma, \boldsymbol{x}_{-d}), b(\boldsymbol{x}|\mathcal{X}_2^\gamma, \boldsymbol{x}_{-d})$ *are also almost everywhere* 0 *or* 1 *(*μ*-a.e.),* $b(\boldsymbol{x}|\mathcal{X}_1^\gamma \cup \mathcal{X}_2^\gamma, \boldsymbol{x}_{-d}), b(\boldsymbol{x}|\mathcal{X}_1^\gamma \cap \mathcal{X}_2^\gamma, \boldsymbol{x}_{-d})$ *are almost everywhere* 0 *or* 1*. Therefore, we obtain:*

$$\begin{aligned}
\int_{\boldsymbol{x}_{-d} \in \mathcal{X}_{-d}} b(\boldsymbol{x}|\mathcal{X}_1^\gamma \cap \mathcal{X}_2^\gamma, \boldsymbol{x}_{-d}) \mu(d\boldsymbol{x}_{-d}) &= \frac{\mu(\mathcal{X}_1^\gamma \cap \mathcal{X}_2^\gamma)}{\mu(\mathcal{X}_d)} \ \mu\text{-a.e.,}\\
\int_{\boldsymbol{x}_{-d} \in \mathcal{X}_{-d}} b(\boldsymbol{x}|\mathcal{X}_1^\gamma \cup \mathcal{X}_2^\gamma, \boldsymbol{x}_{-d}) \mu(d\boldsymbol{x}_{-d}) &= \frac{\mu(\mathcal{X}_1^\gamma \cup \mathcal{X}_2^\gamma)}{\mu(\mathcal{X}_d)} \ \mu\text{-a.e..}
\end{aligned} \tag{32}$$

Table 1: The search space of HPOlib. All tasks have the same search space by default. Each benchmark has performance metrics of 62208 possible configurations with 4 random seeds.

Hyperparameter	Choices
Number of units 1	$\{2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$
Number of units 2	$\{2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$
Dropout rate 1	$\{0.0, 0.3, 0.6\}$
Dropout rate 2	$\{0.0, 0.3, 0.6\}$
Activation function 1	{ReLU, tanh}
Activation function 2	{ReLU, tanh}
Batch size	$\{2^3, 2^4, 2^5, 2^6\}$
Learning rate scheduler	{cosine, constant}
Initial learning rate	{5e-4, 1e-3, 5e-3, 1e-2, 5e-2, 1e-1}

Using those results, the following holds:

$$
\begin{aligned}
s(\mathcal{X}_1^\gamma, \mathcal{X}_2^\gamma) = \frac{\mu(\mathcal{X}_1^\gamma \cap \mathcal{X}_2^\gamma)}{\mu(\mathcal{X}_1^\gamma \cup \mathcal{X}_2^\gamma)} &= \frac{\int_{\boldsymbol{x}_{-d} \in \mathcal{X}_{-d}} b(\boldsymbol{x} | \mathcal{X}_1^\gamma \cap \mathcal{X}_2^\gamma, \boldsymbol{x}_{-d}) \mu(d\boldsymbol{x}_{-d})}{\int_{\boldsymbol{x}_{-d} \in \mathcal{X}_{-d}} b(\boldsymbol{x} | \mathcal{X}_1^\gamma \cup \mathcal{X}_2^\gamma, \boldsymbol{x}_{-d}) \mu(d\boldsymbol{x}_{-d})} \\
&= \frac{\int_{\boldsymbol{x} \in \mathcal{X}} b(\boldsymbol{x} | \mathcal{X}_1^\gamma \cap \mathcal{X}_2^\gamma) \mu(d\boldsymbol{x})}{\int_{\boldsymbol{x} \in \mathcal{X}} b(\boldsymbol{x} | \mathcal{X}_1^\gamma \cup \mathcal{X}_2^\gamma) \mu(d\boldsymbol{x})}
\end{aligned}
\tag{33}
$$

and this completes the proof.

E Details of the experiments

E.1 Dataset description

In the experiments, we used the following tabular benchmarks:

1. HPOlib (slice localization, naval propulsion, parkinsons telemonitoring, protein structure) (Klein and Hutter (2019)), and
2. NMT-Bench (Somali to English, Swahili to English, Tagalog to English) (Zhang and Duh (2020)).

The search spaces for each benchmark are presented in Tables 1, 2.

HPOlib is an HPO tabular benchmark for neural networks on regression tasks. This benchmark has four regression tasks and provides the number of parameters, runtime, and training and validation mean squared error (MSE) for each configuration. In the experiments, we optimized validation MSE and runtime.

NMT-Bench is an HPO tabular benchmark for neural machine translation (NMT) and we optimize hyperparameters of transformer. This benchmark provides three tasks: NMT for Somali to English, Swahili to English, and Tagalog to English. Since the original search space of "Swahili to English" is slightly different from the others, we reduced its search space to be identical to the others. In the experiments, we optimized translation accuracy (BLEU) Papineni *et al.* (2002) and decoding speed. Note that as mentioned in Footnote 7 in the original paper, some hyperparameters do not have both BLEU and decoding speed due to training failures and thus we pad the worst possible values for such cases.

E.2 Details of optimization methods

Baseline methods

In the experiments, we used the following:

1. MO-TPE (Ozaki *et al.* (2022)),
2. Random search (Bergstra and Bengio (2012)),

Table 2: The search space of NMT-Bench. To make the experiments easier, we reduce the choices of "Number of layers" in the "Swahili to English" dataset and thus all tasks have the same search space. Each benchmark has performance metrics of 648 possible configurations with 1 random seed.

Hyperparameter	Choices
Number of BPE symbols ($\times 10^3$)	$\{2^0, 2^1, 2^2, 2^3, 2^4, 2^5\}$
Number of layers	$\{1, 2, 4\}$
Embedding size	$\{2^8, 2^9, 2^{10}\}$
Number of hidden units in each layer	$\{2^{10}, 2^{11}\}$
Number of heads in self-attention	$\{2^3, 2^4\}$
Initial learning rate ($\times 10^{-4}$)	$\{3, 6, 10\}$

3. TST-R (Wistuba *et al.* (2016)),
4. RGPE (Feurer *et al.* (2018)).

Note that since TST-R and RGPE are introduced as meta-learning methods for single-objective optimization problems, we extended them to MO settings using ParEGO (Knowles (2006)) and EHVI (Emmerich *et al.* (2011)). Both extensions require only ranking loss between configurations and thus we defined the ranking of each configuration using the non-dominated rank and crowding distance as in NSGA-II (Deb *et al.* (2002)). Although TST-R and RGPE implementations were provided in `https://github.com/automl/transfer-hpo-framework`, since the implementations are based on SMAC3 [2], which does not support EHVI, we re-implemented RGPE and TST-R using BoTorch [3]. Our BoTorch implementations are also available in `https://github.com/nabenabe0928/meta-learn-tpe`. We chose $\gamma = 0.1$ for MO-TPE by following the original paper (Ozaki *et al.* (2020)) and modified the tie-break method from HSSP to crowding distance as TST-R and RGPE also use crowding distance for tie-breaking in our experiments. Furthermore, we employed multivariate kernel to improve the performance of TPE (Falkner *et al.* (2018)).

More details about the settings of meta-learning TPE

As seen in Algorithm 2, meta-learning TPE has several control parameters:

1. (TPE) the number of initial samples N_{init},
2. (TPE) the number of candidates for each iteration N_s,
3. (TPE) the quantile for the observation split γ,
4. (New) dimension reduction factor η,
5. (New) sample size of Monte-Carlo sampling for the task similarity approximation S, and
6. (New) the ratio of random sampling in the ϵ-greedy algorithm ϵ.

Note that (TPE) means the parameters already exist from the original version and (New) means the parameters added in our proposition. Since the original TPE paper (Bergstra *et al.* (2011, 2013)) uses 5% of total evaluations for the initialization and 100 candidates for each iteration, we set $N_{\mathrm{init}} = 5$ and $\epsilon = 0.05$ in our experiments and used $N_s = 100$. $\gamma = 0.1$ followed Ozaki *et al.* (2020). For the other parameters, we used $S = 1000$ and $\eta = 5/2$. Furthermore, we employed multivariate kernel as in Falkner *et al.* (2018) and used crowding distance to tie-break non-dominated rank as mentioned in the previous section.

F Additional results of the experiments

Figures 7 and 8 show the 50% empirical attainment surfaces (Fonseca and Fleming (1996)) for each method on each benchmark [4]. The 50% empirical attainment surface represents the Pareto front of

[2] SMAC3: https://github.com/automl/SMAC3

[3] BoTorch: https://github.com/pytorch/botorch

[4] For the visualization of emprical attainment surface, we used the following package: `https://github.com/nabenabe0928/empirical-attainment-func/`

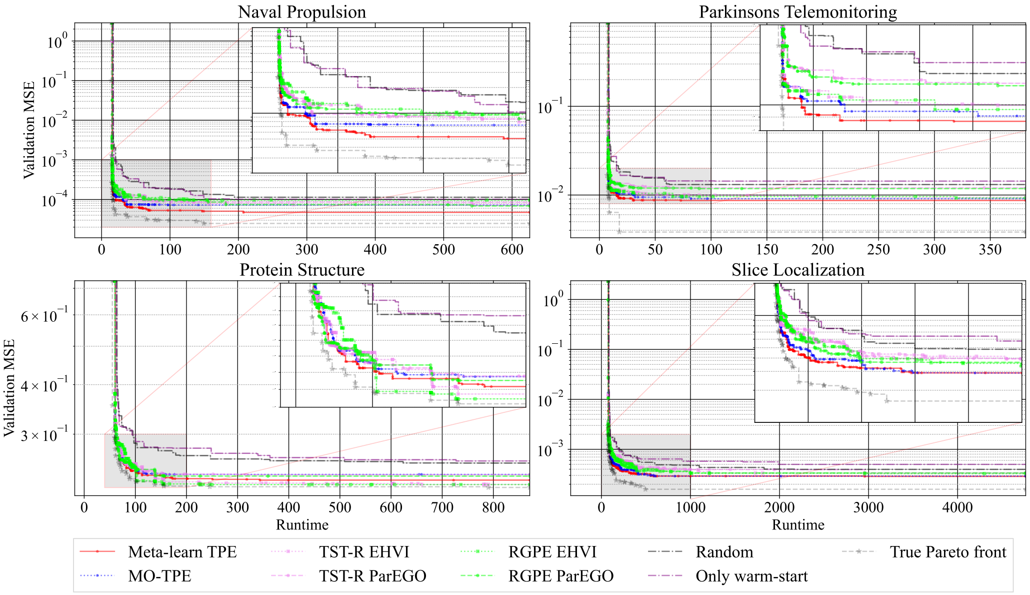

Figure 7: The 50% empirical attainment surfaces of each method over 20 different random seeds on HPOlib. The objectives are to minimize runtime and validation MSE.

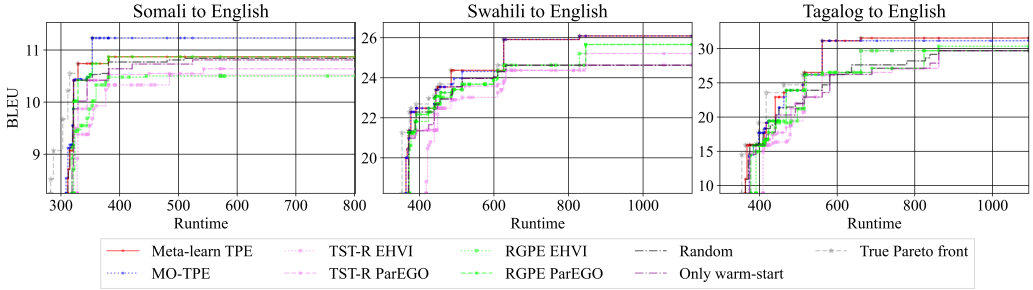

Figure 8: The 50% empirical attainment surfaces of each method over 20 different random seeds on NMT-Bench. The objectives are to minimize runtime and maximize BLEU.

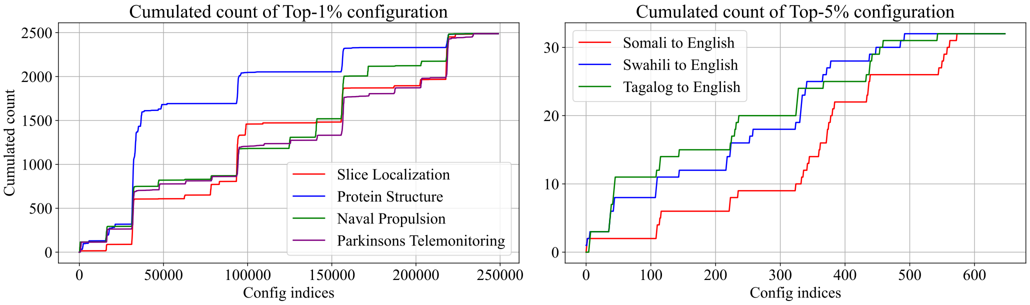

Figure 9: The cumulated count of top-1% (HPOlib) and top-5% (NMT-Bench) over configuration indices. Although subsequent indices do not necessarily imply that those configurations are close in the search space, each index identifies the same configuration for each dataset and thus we can see trends by tracking the variations of each curve.

the observations achieved by 50% of independent runs. As discussed in Section 3, we mentioned that our method did not yield the best hypervolume on “Somali to English” and “Protein Structure”. According to the figures, validation MSE in “Protein Structure” and BLEU in “Somali to English” are underexplored by our method.

In Figure 9, we plot the cumulated count of top configurations by configuration indices for each dataset and we can see that both “Protein Structure” and “Somali to English” have different distribution, which may imply the γ-set for each dataset is not close to that for the other datasets. In fact, we could find out that only “Somali to English” requires high number of BPE symbols compared to other datasets. From the results, we could conclude that while our method is somewhat affected by knowledge transfer from not similar meta-tasks, our method, at least, exhibits indistinguishable performance from that of MO-TPE in our settings.

References

J. Aitchison and CG. Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3), 1976.

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

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

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

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.

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.

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.

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

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

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

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

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

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.

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

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.

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.

D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

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

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

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.

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.

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

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.

K. Papineni, S. Roukos, T. Ward, and WJ. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics*, 2002.

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

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.

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.

W. Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1976.

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

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.

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

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

AB. Tsybakov. *Introduction to Nonparametric Estimation*. Springer Publishing Company, 2008.

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

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

S. Watanabe and F. Hutter. Revisiting hyperparameter importance assessment: Local importance from the Lebesgue view. *arXiv:4522.459*, 2022.

D. Wied and R. Weißbach. Consistency of the kernel density estimator: a survey. *Statistical Papers*, 53, 2012.

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.

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.

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.

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

K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, 2013.