c-TPE: Generalizing Tree-structured Parzen Estimator with Inequality Constraints for Continuous and Categorical Hyperparameter Optimization

Paper under double-blind review

Abstract

Hyperparameter optimization (HPO) is crucial for strong performance of deep learning algorithms. A widely-used versatile HPO method is a variant of Bayesian optimization called tree-structured Parzen estimator (TPE), which splits data into good and bad groups and uses the density ratio of those groups as an acquisition function (AF). However, real-world applications often have some constraints, such as memory requirements, or latency. In this paper, we present an extension of TPE to constraint optimization (c-TPE) via simple factorization of AFs. The experiments demonstrate c-TPE is robust to various constraint levels and c-exhibits the best average rank performance among existing methods with statistical significance on search spaces with categorical parameters on 81 settings.

1 Introduction

While deep learning has achieved various breakthrough successes, its performance highly depends on proper settings of its hyperparameters (Chen *et al.* (2018); Melis *et al.* (2018)). Furthermore, practical applications often impose several constraints on computational memory or latency of inference. In such cases, we need to extend the usual hyperparameter optimization (HPO) task to constraint optimization settings.

Recently, many open source softwares (OSS) for HPO have been developed such as Optuna (Akiba *et al.* (2019)), Hyperopt (Bergstra *et al.* (2013)), and Ray (Liaw *et al.* (2018)). All of these support a variant of Bayesian optimization (BO) called the Tree-structured Parzen estimator (TPE) (Bergstra *et al.* (2011, 2013)), and Optuna, which uses TPE as its main algorithm, was core to reaching the second place in the Open Images Object Detection Competition (Akiba *et al.* (2019)). As a BO method, TPE determines the next configuration using a surrogate model and an acquisition function (AF) that judges the promise of a configuration based on the surrogate model. While standard BO uses Gaussian process for the surrogate, TPE uses the ratio of Parzen estimators for good and bad observations (Bergstra *et al.* (2011)). Although TPE has been used widely due to its versatility and stable performance even when the search space contains categorical parameters, it has not been extended to constraint optimization and thus practitioners facing constraints need to use alternatives.

In this paper, we show how to extend TPE to constraint optimization settings. Since the AF of TPE is probability of improvement (PI), but not expected improvement (EI), we can simply take the product of AFs in both the objective and constraints based on the AF proposed by Gelbart *et al.* (2014). Note that if we naïvely combine those ideas, c-TPE will not work well and thus we provide an in-depth analysis on how to enhance c-TPE in Appendix B. In the series of experiments, we demonstrate (1) the strong performance of c-TPE with statistical significance on search spaces that include categorical parameters and (2) robustness to changes in the constraint level. As used tabular benchmarks do not represent all possible tasks, we discuss the limitations of our work in Appendix J. The full results of the experiments are available at `https://anonymous.4open.science/r/constraint_tpe-342C/appendix.pdf`.

2022 NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems.

Algorithm 1 c-TPE algorithm

1: N_{init} (The number of initial configurations), N_s (The number of candidates to consider in the optimization of the AF)
2: $\mathcal{D} \leftarrow \emptyset$
3: **for** $n = 1, \ldots, N_{\text{init}}$ **do**
4: Randomly pick $\boldsymbol{x}$
5: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}, f(\boldsymbol{x}), c_1(\boldsymbol{x}), \ldots, c_C(\boldsymbol{x}))\}$
6: **while** Budget is left **do**
7: $\mathcal{S} = \emptyset$
8: **for** $i = 0, \ldots, C$ **do**
9: Split $\mathcal{D}$ into $\mathcal{D}_i^{(l)}$ and $\mathcal{D}_i^{(g)}$, $\hat{\gamma}_i \leftarrow |\mathcal{D}_i^{(l)}|/|\mathcal{D}|$
10: Build $l(\boldsymbol{x}|\mathcal{D}_i^{(l)}), g(\boldsymbol{x}|\mathcal{D}_i^{(g)})$
11: $\{\boldsymbol{x}_j\}_{j=1}^{N_s} \sim l(\boldsymbol{x}|\mathcal{D}_i^{(l)}), \mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{x}_j\}_{j=1}^{N_s}$
12: Pick $\boldsymbol{x}_{\text{opt}} \in \text{argmax}_{\boldsymbol{x} \in \mathcal{S}} \prod_{i=0}^{C} r_i^{\text{rel}}(\boldsymbol{x}|\mathcal{D})$
13: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}_{\text{opt}}, f(\boldsymbol{x}_{\text{opt}}), c_1(\boldsymbol{x}_{\text{opt}}), \ldots, c_C(\boldsymbol{x}_{\text{opt}}))\}$

2 Constraint TPE (c-TPE)

In this section, we briefly describe the AF of c-TPE and its algorithm. We provide the theoretical background and the analysis for the algorithm construction in Appendices A, B.

2.1 The acquisition function

In single-objective optimization problems, 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 $\mathcal{D}^{(g)}$ by a threshold f^{γ} that is the top $\lceil \gamma |\mathcal{D}| \rceil$-quantile objective value in $\mathcal{D}$. Then we build Parzen estimators $l(\boldsymbol{x}|\mathcal{D}^{(l)}), g(\boldsymbol{x}|\mathcal{D}^{(g)})$ and compute the AF via $r(\boldsymbol{x}|\mathcal{D}) \coloneqq l(\boldsymbol{x}|\mathcal{D}^{(l)})/g(\boldsymbol{x}|\mathcal{D}^{(g)})$. This density ratio is known to be EI (Bergstra *et al.* (2011)). The extension of EI to constraint settings is ECI invented by Gardner *et al.* (2014) and Gelbart *et al.* (2014) and this AF takes the product of EI of the objective and PIs of constraints. Using the formulations, we show that the AF of c-TPE is equivalent to $\prod_{i=0}^{C} r_i^{\text{rel}}(\boldsymbol{x}|\mathcal{D}) \coloneqq \prod_{i=0}^{C} (\hat{\gamma}_i + (1 - \hat{\gamma}_i r_i(\boldsymbol{x}|\mathcal{D})^{-1}))^{-1}$ where $r_i(\boldsymbol{x}|\mathcal{D}) \coloneqq l(\boldsymbol{x}|\mathcal{D}_i^{(l)})/g(\boldsymbol{x}|\mathcal{D}_i^{(g)})$ is the density ratio of the i-th constraint for $i \in \{1, \ldots, C\}$ and that of the objective for $i = 0$, $\mathcal{D}_i^{(l)}, \mathcal{D}_i^{(g)}$ are obtained by splitting $\mathcal{D}$ based on the algorithm discussed in Section 2.2, and $\hat{\gamma}_i \coloneqq |\mathcal{D}_i^{(l)}|/|\mathcal{D}|$. We describe more details about the background knowledge in Appendix A and those about the validity of the formulation in Appendix B.

2.2 Algorithm modification details from a naïve combination

Algorithm 1 is the pseudocode of c-TPE with color-coded modifications. The modifications of the algorithm compared to a naïve combination of TPE and ECI are as follows:

1. **Split algorithm of** $\mathcal{D}$ (Line 9; more details in Appendix B.3)
2. **Computation of** $\text{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}]$ (Line 12)

Note that the benefits of those modifications are dicussed in Appendices B.2.1 and B.2.2.

The split algorithm in the original TPE by Bergstra *et al.* (2013) first sorts the observations $\mathcal{D}$ by f and takes the first $\lceil \sqrt{N}/4 \rceil$ observations as $\mathcal{D}_0^{(l)}$ and the rest as $\mathcal{D}_0^{(g)}$. On the other hand, our method includes all the observations until the $\lceil \sqrt{N}/4 \rceil$-th **feasible** observation into $\mathcal{D}_0^{(l)}$ and the rest into $\mathcal{D}_0^{(g)}$. As mentioned earlier, this split guarantees $\mathcal{D}_0^{(l)}$ to have at least one feasible solution unless we have no feasible solutions. For the split of constraints, we first check the upper bound of $\{c_{i,n}\}_{n=1}^{|\mathcal{D}|}$ that satisfies a given threshold $c_i^\star$ and let this value be c_i'. Note that $c_{i,n}$ is the i-th constraint value in the n-th observation. If such values do not exist, we take the best value $\min\{c_{i,n}\}_{n=1}^{|\mathcal{D}|}$ so that the optimization of this constraint will be strengthened (see Theorem 1 in Appendix). Then we split $\mathcal{D}$ into $\mathcal{D}_i^{(l)}$ and $\mathcal{D}_i^{(g)}$ so that $\mathcal{D}_i^{(l)}$ includes only observations that satisfy $c_{i,n} \leq c_i'$ and vice versa.

The computation of the AF uses $\prod_{i=0}^{C} r_i^{\text{rel}}(\boldsymbol{x}|\mathcal{D})$ instead of a naïve computation $\prod_{i=0}^{C} r_i(\boldsymbol{x}|\mathcal{D})$. Its validity is described in Appendix 2.1.

Table 1: The table shows (Wins/Loses/Ties) of c-TPE against each method for optimizations with different constraint levels. Bold numbers indicate $p < 0.01$ of the hypothesis "The other method is better than c-TPE" by the Wilcoxon signed-rank test.

Quantiles	$\gamma_i^{\text{true}} = 0.1$				$\gamma_i^{\text{true}} = 0.5$				$\gamma_i^{\text{true}} = 0.9$			
Methods / # of configs	50	100	150	200	50	100	150	200	50	100	150	200
Naïve c-TPE	**26/0/1**	**27/0/0**	**27/0/0**	**27/0/0**	**25/0/2**	**25/0/2**	**25/1/1**	**25/0/2**	**21/5/1**	**23/1/3**	**21/1/5**	**24/1/2**
Vanilla TPE	**27/0/0**	**27/0/0**	**27/0/0**	**27/0/0**	**25/0/2**	**26/0/1**	**26/1/0**	**24/0/3**	14/11/2	**18/8/1**	**15/5/7**	**16/7/4**
Random	**25/0/2**	**26/1/0**	**27/0/0**	**27/0/0**	**27/0/0**	**26/0/1**	**26/0/1**	**27/0/0**	**27/0/0**	**27/0/0**	**27/0/0**	**27/0/0**
CNSGA-II	**25/0/2**	**27/0/0**	**24/0/3**	**24/0/3**	**26/0/1**	**26/0/1**	**26/0/1**	**25/0/2**	**26/1/0**	**27/0/0**	**27/0/0**	**26/0/1**
NEI	**24/1/2**	**27/0/0**	**27/0/0**	**27/0/0**	**27/0/0**	**26/0/1**	**26/0/1**	**27/0/0**	**27/0/0**	**27/0/0**	**27/0/0**	**27/0/0**
HM2	**23/2/2**	**26/1/0**	**25/2/0**	**25/2/0**	**22/3/2**	**23/2/2**	**25/1/1**	**23/0/4**	**27/0/0**	**27/0/0**	**23/0/4**	**26/0/1**

3 Experiments

3.1 Setup

In the experiments, we report a comprehensive evaluation of c-TPE on tabular benchmarks. The reason behind this choice is that tabular benchmarks enable us to control the quantiles of each constraint γ_i^{true}. The quantile γ_i^{true} can be obtained by looking up all the results and we use 9 different γ_i^{true} to observe the performance variation on different difficulties.

The evaluations were performed on (1) HPOlib (4 benchmarks by Klein and Hutter (2019)), (2) NAS-Bench-101 (3 benchmarks by Ying *et al.* (2019)), and (3) NAS-Bench-201 (3 benchmarks by Dong and Yang (2020)), and the search space for each benchmark followed Awad *et al.* (2021). As constraints, we use network size, runtime, or both.

As the baseline methods, we chose (1) random search (Bergstra and Bengio (2012)), (2) CNSGA-II (Deb *et al.* (2002)), (3) NEI provided in Facebook Ax (Letham *et al.* (2019)), (4) Hypermapper2.0 (HM2) (Nardi *et al.* (2019)), (5) vanilla TPE (optimize only loss as if we do not have constraints) (6) naïve c-TPE (the naïve combination discussed in Section 2.2). Note that we could not perform the optimization of CIFAR10C in NAS-Bench-101 using NEI and HM2 due to too long computation time and thus we show the average ranks using only 9 benchmarks (other than CIFAR10C).

For more details about how to calculate γ_i^{true}, tabular benchmark datasets, and the baseline methods, see Appendix F. The source code is available at `https://anonymous.4open.science/r/constraint_tpe-342C` along with complete scripts to reproduce the experiments, tables, and figures.

3.2 Results

In Figure 1, we show how c-TPE performance improves given various levels of constraints and we chose $\gamma_i^{\text{true}} \in \{0.1, 0.5, 0.9\}$ for this figure; see Appendix G for the complete results of $\gamma_i^{\text{true}} \in \{0.1, 0.2, \ldots, 0.9\}$. Table 1 presents the numbers of wins/loses/ties and statistical significance by the Wilcoxon signed-rank test over 27 settings (9 benchmarks $\times$ 3 constraint choices). Note that we used the results of the optimizations on all datasets in the statistical test and the number of wins was counted by comparing medians of performance between two methods. In Appendix H, we provide the full results of the average rank over time for each constraint level.

As seen in the bottom figures, the naïve c-TPE is completely defeated by other methods while c-TPE achieves the best or at least indistinguishable performance from the best. This gap is caused by small overlaps of the promising regions in the objective and constraints as discussed in Appendix B.2.2. In fact, 41% of the top-10% configurations belong to infeasible domains in NAS-Bench-201 of $\gamma_i^{\text{true}} = 0.9$ although only 16% and 23% are infeasible in HPOlib and NAS-Bench-101 of $\gamma_i^{\text{true}} = 0.9$, respectively. This fact also affects the discrepancy between c-TPE and the vanilla TPE. As TPE is not a uniform sampler and tries to sample from promising regions, $\hat{\gamma}_i$ will not necessarily approach γ_i^{true}. For example, it is natural to consider $\hat{\gamma}_i$ to be closer to $100 - 41 = 59\%$ rather than 90% as 59% of configurations are feasible in the top-10% domain. Since this ratio is much lower than γ_i^{true}, the performance of c-TPE is much better than the vanilla TPE due to Theorem 1 in Appendix.

For the middle figures of $\gamma_i^{\text{true}} = 0.1, 0.5$, most methods exhibit indistinguishable performance from random search especially in the beginning because of the combination of the high dimensional search space ($D = 26$) and little information on feasible domains in the early stage of optimizations

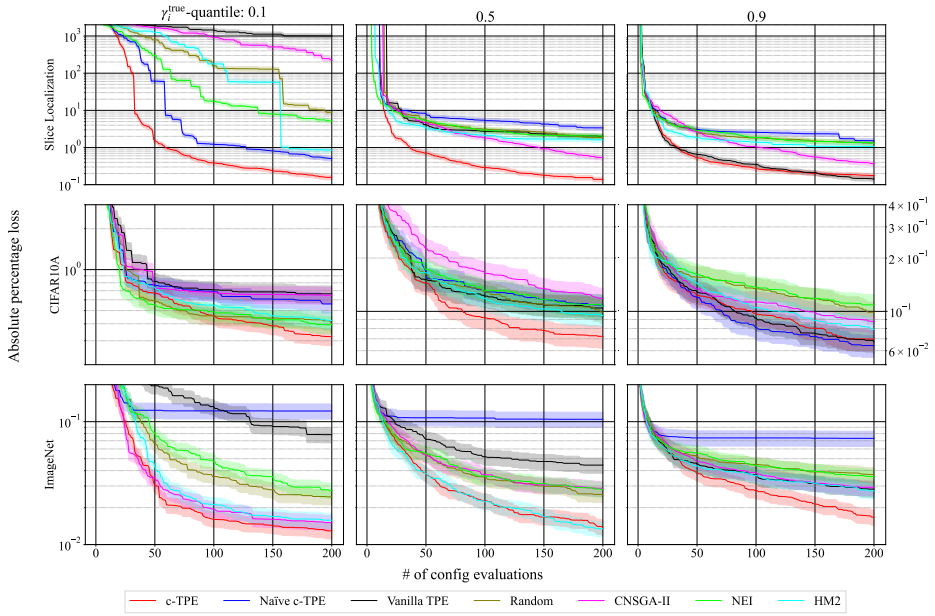

Figure 1: Figures show the performance curves on Slice Localization in HPOlib, CIFAR10A in NAS-Bench-101, and ImageNet16-120 in NAS-Bench-201 with constraints of runtime and network size. We picked $\gamma_i^{\text{true}} = 0.1$ (left), 0.5 (center), 0.9 (right). The vertical axis shows the absolute percentage loss $(f_{\text{observed}} - f_{\text{oracle}})/f_{\text{oracle}}$ where f_{oracle} is determined by looking up all feasible configurations in each benchmark. Note that each row shares the vertical axis except NAS-Bench-101. For $\gamma_i^{\text{true}} = 0.1$ in NAS-Bench-101, we separately scaled for the readability. All results and further discussion are available in Appendix G.

although c-TPE outperforms in the end. In $\gamma_i^{\text{true}} = 0.9$, the naïve c-TPE is slightly better than c-TPE due to large overlaps (84% of the top-10% configurations are feasible). It implies that if search space is high dimension and overlaps in promising regions and feasible domains are large, it might be better to greedily optimize the objective rather than regularizing the optimization of the objective by considering constraints.

For the top figures, c-TPE outperforms other methods and its performance almost coincides with that of the vanilla TPE in $\gamma_i^{\text{true}} = 0.9$. Since the naïve c-TPE does not consider the priority of each constraint and the objective, it does not exhibit stability when the constraint level changes.

As seen in the figures, while the performance of c-TPE, in fact, is stable across all constraint levels, the performance of NEI, HM2, and CNSGA-II variate depending on constraint levels. Furthermore, Table 1 shows that c-TPE is significantly better than other methods in almost all settings. This experimentally validates the robustness of c-TPE to the variations in constraint levels.

4 Conclusion

In this paper, we introduced c-TPE, a new Bayesian constraint optimization method. The AF of c-TPE is naturally extended from the original formulation. In Section 2, we provided the naïve combination of constraint BO and TPE and described why the naïve extension fails in some circumstances. In the experiments, we first showed that the performance of c-TPE is not degraded over various constraint levels while the other BO methods we evaluated (HM2 and NEI) degraded as constraints became looser. Furthermore, the proposed method outperformed the other methods with statistical significance; however, since we focus only on the tabular benchmarks to enable the stability analysis of the performance variations depending on constraint levels, we discuss other possible situations where c-TPE might not perform well in Appendix J. Since TPE is very versatile and prominently used in several active OSS tools, such as Optuna and Ray, and since c-TPE is an extension to the constraint setting that showed significant performance among existing methods available in those packages, c-TPE is likely to be integrated into those packages quickly and yield direct positive impact to practitioners in the future. To this end, we also discuss potential societal impacts that our method might cause in Appendix K.

Acknowledgments

Paper under double-blind review

References

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *International Conference on Knowledge Discovery & Data Mining*, 2019.

N. Awad, N. Mallik, and F. Hutter. DEHB: Evolutionary hyberband for scalable, robust and efficient hyperparameter optimization. *arXiv:2105.09821*, 2021.

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.

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

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

X. Dong and Y. Yang. NAS-bench-201: Extending the scope of reproducible neural architecture search. *arXiv:2001.00326*, 2020.

J. Gardner, M. Kusner, ZE. Xu, K. Weinberger, and J. Cunningham. Bayesian optimization with inequality constraints. In *International Conference on Machine Learning*, 2014.

M. Gelbart, J. Snoek, and R. Adams. Bayesian optimization with unknown constraints. *arXiv:1403.5607*, 2014.

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

B. Letham, B. Karrer, G. Ottoni, and E. Bakshy. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. *arXiv:1807.05118*, 2018.

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.

L. Nardi, D. Koeplinger, and K. Olukotun. Practical design space exploration. In *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 347–358. IEEE, 2019.

C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Please see Section 2 for the discussion of the modifications we made and Section 3 for the discussion of the performance analysis.
   (b) Did you describe the limitations of your work? [Yes] Please see Appendix J.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see Appendix K.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Please see Section A.2.
   (b) Did you include complete proofs of all theoretical results? [Yes] Please see Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please check `https://anonymous.4open.science/r/constraint_tpe-342C/`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see Section 2 for the algorithm construction and Appendix F for the hyperparameter choices.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Please see Appendix G. For the average rank, we do not put error bars because the average ranks is the mean over all experiment settings of ranks by median with respect to random seeds. We could technically add standard deviation bands of the mean of ranks by median, but this is not really common. We, instead, provide the statistical test using 50 random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] The computational time is out of scope in this paper.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Please check Appendix F.
   (b) Did you mention the license of the assets? [No] We did not mention it in the paper, but we already added Apache2.0 to the repository. `https://anonymous.4open.science/r/constraint_tpe-342C/`.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Background

A.1 Preliminaries

We use the following definitions to make the discussion of constraint levels simpler:

Definition 1 (γ-quantile value) *Given a function $f : \mathcal{X} \to \mathbb{R}$ and a quantile $\gamma \in (0, 1]$, f^γ is said to be the γ-quantile value of the given function f if $\mathcal{X}' = \{\boldsymbol{x} \in \mathcal{X} | f(\boldsymbol{x}) \leq f^\gamma\}$ satisfies $\mu(\mathcal{X}')/\mu(\mathcal{X}) = \gamma$ where $\mu : \mathbb{R}^D \to \mathbb{R}_{\geq 0}$ is the Lebesgue measure.*

Definition 2 *Given a constraint $c : \mathcal{X} \to \mathbb{R}$ and a constraint threshold $c^\star \in \mathbb{R}$, $\gamma_{c^\star}$ is defined as the quantile of the constraint c such that $c^\star = c^{\gamma_{c^\star}}$.*

Definition 3 (Γ-feasible domain) *Given a set of constraints $c_i^\star \in \mathbb{R}$ (for $i \in \{1, \ldots, C\}$) and a quantile $\Gamma \in (0, 1]$, the domain is said to be the Γ-feasible domain if $\mathcal{X}' = \{\boldsymbol{x} \in \mathcal{X} | \forall i, c_i(\mathcal{X}) \leq c_i^\star\}$ satisfies $\Gamma = \mu(\mathcal{X}')/\mu(\mathcal{X})$.*

where $\boldsymbol{x} \in \mathbb{R}^D$ is a hyperparameter configuration, $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$ is the search space of the hyperparameter configurations, $\mathcal{X}_i$ (for $i = 1, \ldots, D$) is the domain of the i-th hyperparameter, Note that we consider two assumptions mentioned in the next section and those assumptions allow the whole discussion to be extended to search spaces with categorical parameters.

A.2 Assumptions

In this paper, we assume the following:

1. Objective function $f : \mathcal{X} \to \mathbb{R}$ and constraint functions $c_i : \mathcal{X} \to \mathbb{R}$ are Lebesgue integrable and are measurable functions defined over the compact measurable subset $\mathcal{X} \subseteq \mathbb{R}^D$,
2. The support of the probability of improvement for the objective $\mathbb{P}(f \leq f^\star | \mathcal{X}, \mathcal{D})$ and each constraint $\mathbb{P}(c_i \leq c_i^\star | \boldsymbol{x}, \mathcal{D})$ covers the whole domain $\mathcal{X}$ for an arbitrary choice of $f^\star, c_i^\star \in \mathbb{R}$,

where $\mathcal{D} = \{(\boldsymbol{x}_n, f_n, \boldsymbol{c}_n)\}_{n=1}^N$ is a set of observations, and $\boldsymbol{c}_n = [c_{1,n}, \ldots, c_{C,n}] \in \mathbb{R}^C$ is the n-th observation of each constraint. The Lebesgue integrability easily holds for TPE as TPE only considers the order of each configuration and almost all functions are measurable unless they are constructive. Note that we also assume a categorical parameter to be $\mathcal{X}_i = [1, K]$ as in the TPE implementation (Bergstra *et al.* (2011)) where K is a number of categories. As we do not require the continuity of f and c_i 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-Aitken 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.

A.3 Bayesian optimization (BO)

Suppose we would like to **minimize** a loss metric $f(\boldsymbol{x}) = \mathcal{L}(\boldsymbol{x}, \mathcal{A}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})$ of a supervised learning algorithm $\mathcal{A}$ given training and validation datasets $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}$, then the formulation of HPO is defined as follows:

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

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 $p(f|\boldsymbol{x}, \mathcal{D})$ where $\mathcal{D} = \{(\boldsymbol{x}_n, f_n)\}_{n=1}^N$ is a set of observations. Then, the optimization in each iteration is replaced with the optimization of the so-called acquisition function (AF). A common choice for the AF is the following EI (Jones *et al.* (1998)):

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

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

A.4 Tree-structured Parzen estimator (TPE)

TPE (Bergstra *et al.* (2011, 2013)) is a variant of BO methods and it uses the EI. To transform Eq. (2), we assume the following:

$$p(\boldsymbol{x}|f,\mathcal{D}) = \begin{cases} l(\boldsymbol{x}|\mathcal{D}^{(l)}) & (f \leq f^{\gamma}) \\ g(\boldsymbol{x}|\mathcal{D}^{(g)}) & (f > f^{\gamma}) \end{cases} \tag{3}$$

where $\mathcal{D}^{(l)}, \mathcal{D}^{(g)}$ are the observations with $f_i \leq f^{\gamma}$ and $f_i > f^{\gamma}$, respectively. Note that f^{γ} is the top-γ quantile objective value in $\mathcal{D}$ at each iteration and $l(\boldsymbol{x}|\mathcal{D}^{(l)}), g(\boldsymbol{x}|\mathcal{D}^{(g)})$ are built by the kernel density estimation (Bergstra *et al.* (2011, 2013); Falkner *et al.* (2018)). Combining Eqs. (2), (3) and Bayes' theorem, the AF of TPE is computed as (Bergstra *et al.* (2011)):

$$\mathrm{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}] \overset{\text{affine}}{\propto} r(\boldsymbol{x}|\mathcal{D}) \coloneqq l(\boldsymbol{x}|\mathcal{D}^{(l)})/g(\boldsymbol{x}|\mathcal{D}^{(g)}). \tag{4}$$

where $\phi(\boldsymbol{x}) \overset{\text{affine}}{\propto} \psi(\boldsymbol{x})$ means there exist $a \in \mathbb{R}_+, b \in \mathbb{R}$ such that $\forall \boldsymbol{x} \in \mathcal{X}, \phi(\boldsymbol{x}) = a\psi(\boldsymbol{x}) + b$ and we use $f^\star = f^{\gamma}$ at each iteration. In each iteration, TPE samples configurations from $l(\boldsymbol{x}|\mathcal{D}^{(l)})$ and takes the configuration that satisfies the maximum $r(\boldsymbol{x}|\mathcal{D})$ among the samples.

A.5 BO with unknown constraints

We consider unknown constraints $c_i(\boldsymbol{x}) = \mathcal{C}_i(\boldsymbol{x}, \mathcal{A}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})$ $(\forall i \in [1, C])$, e.g. memory consumption of algorithm $\mathcal{A}$ given a configuration $\boldsymbol{x}$. Then the optimization is formulated as follows:

$$\begin{gathered} \boldsymbol{x}_{\text{opt}} \in \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) \\ \text{subject to } \forall i \in [1, C], c_i(\boldsymbol{x}) \leq c_i^\star \end{gathered} \tag{5}$$

where $c_i^\star \in \mathbb{R}$ is a threshold for the i-th constraint. Note that we reverse the sign of inequality in the case where constraints must be larger than a given threshold. To extend BO to constraint optimization, the following expected constraint improvement (ECI) has been proposed (Gelbart *et al.* (2014)):

$$\mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}] = \mathrm{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}] \prod_{i=1}^{C} \mathbb{P}(c_i \leq c_i^\star|\mathcal{D}), \tag{6}$$

where $\boldsymbol{c}^\star = [c_1^\star, \ldots, c_C^\star] \in \mathbb{R}^C$ and $\mathcal{D} = \{(\boldsymbol{x}_n, f_n, \boldsymbol{c}_n)\}_{n=1}^N$ is a set of observations, and $\boldsymbol{c}_n = [c_{1,n}, \ldots, c_{C,n}] \in \mathbb{R}^C$ is the n-th observation of each constraint. When we model the conditional dependence, we obtain

$$\mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}] = \mathrm{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}]\mathbb{P}(c_1 \leq c_1^\star, \ldots, c_C \leq c_C^\star|\mathcal{D}). \tag{7}$$

However, the simplified factorized form of Eq. (6) is the common choice.

B More details about constraint TPE (c-TPE)

In this section, we first prove that EI and PI are intrinsically equivalent in the TPE formulation and thus TPE can be extended to the constraint settings via the simple product of the AFs. Then we describe an extension naïvely inspired by the original TPE and discuss two pitfalls that prevent the naïve extension from efficient search. We propose modifications for those issues and present how the modifications make difference on synthetic problems.

B.1 The acquisition function

Suppose we would like to solve constraint optimization problems formalized in Eq. (5) with the ECI. To realize the ECI in TPE, we first show the following proposition.

Proposition 1 *Under the TPE formulation,* $\mathrm{EI}_{f^\star}[\boldsymbol{x} \mid \mathcal{D}] \propto \mathbb{P}(f \leq f^\star \mid \boldsymbol{x}, \mathcal{D})$ *holds.*

The proof is provided in Appendix C.1. Since PI and EI are equivalent under the TPE formulation, we obtain the following by combining Proposition 1 and Eq. (6) under the TPE formulation:

$$\begin{aligned} \mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}] &= \mathrm{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}] \prod_{i=1}^{C} \mathbb{P}(c_i \leq c_i^\star|\boldsymbol{x}, \mathcal{D}) \\ &\propto \mathrm{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}] \prod_{i=1}^{C} \mathrm{EI}_{c_i^\star}[\boldsymbol{x}|\mathcal{D}] \propto \underbrace{\mathbb{P}(f \leq f^\star|\boldsymbol{x}, \mathcal{D})}_{\overset{\text{affine}}{\propto} r_0(\boldsymbol{x}|\mathcal{D})} \prod_{i=1}^{C} \underbrace{\mathbb{P}(c_i \leq c_i^\star|\boldsymbol{x}, \mathcal{D})}_{\overset{\text{affine}}{\propto} r_i(\boldsymbol{x}|\mathcal{D})}. \end{aligned} \tag{8}$$

Note that we provide the definition of $r_i(\boldsymbol{x}|\mathcal{D})$ for $i \in \{0, 1, \ldots, C\}$ in the next section.

B.2 Two pitfalls in a naïve extension and their solutions

From the discussion above, we could naïvely extend the original TPE to the constraint settings using the split in Eq. (3) and the AF in Eq. (4). More specifically, the naïve extension computes the AF as follows:

1. Pick the top-$\lceil \gamma|\mathcal{D}| \rceil$ objective value $f^\star$ in $\mathcal{D}$
2. Split $\mathcal{D}$ into $\mathcal{D}_0^{(l)}$ and $\mathcal{D}_0^{(g)}$ by $f^\star$, and $\mathcal{D}$ into $\mathcal{D}_i^{(l)}$ and $\mathcal{D}_i^{(g)}$ by $c_i^\star$ for $i \in \{1, \ldots, C\}$
3. Build Parzen estimators $l(\boldsymbol{x}|\mathcal{D}_i^{(l)})$, $g(\boldsymbol{x}|\mathcal{D}_i^{(g)})$ for $i \in \{1, \ldots, C\}$
4. Take the product of density ratios $\prod_{i=0}^C r_i(\boldsymbol{x}|\mathcal{D}) \coloneqq \prod_{i=0}^C l(\boldsymbol{x}|\mathcal{D}_i^{(l)})/g(\boldsymbol{x}|\mathcal{D}_i^{(l)})$ as the AF

Note that $c_i^\star$ is a user-defined threshold and thus $c_i^\star$ is fixed during the optimization. Although this implementation could be naturally inspired by the original TPE, Operations 1 and 4 could incur performance degradation under (1) small overlaps in promising regions for the objective and feasible domains, or (2) vanished constraints. For this reason, we change Operations 1 and 4 into "Pick the top-$\lceil \gamma|\mathcal{D}| \rceil$ **feasible** objective value $f^\star$ in $\mathcal{D}$ " and "Take the product of **relative** density ratios $\prod_{i=0}^C r_i^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D}) \coloneqq \prod_{i=0}^C (\hat{\gamma}_i + (1-\hat{\gamma}_i) r_i(\boldsymbol{x}|\mathcal{D})^{-1})^{-1}$ as the AF ", where we define $\hat{\gamma}_i \coloneqq |\mathcal{D}_i^{(l)}|/|\mathcal{D}|$. Notice that the original TPE by Bergstra *et al.* (2013) computes γ as $\lceil \sqrt{N}/4 \rceil / N$ by default and $\hat{\gamma}_0$ is not necessarily equal to γ after the modification as $|\mathcal{D}^{(l)}| \geq \lceil \sqrt{N}/4 \rceil$. Furthermore, the following corollary, in fact, holds and thus the modified version is valid:

Corollary 1 *Under the TPE formulation,* $\mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}] \propto \prod_{i=0}^C r_i^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D})$.

We provide the proof in Appendix C.2. Then we discuss why those modifications mitigate the issues below. We start from the discussion of "vanished constraints" for more clarity.

B.2.1 Issue I: Vanished constraints

We refer to constraints that are satisfied in almost all configurations as *vanished constraints*. In other words, if the i-th constraint c_i is a vanished constraint, its quantile is $\hat{\gamma}_{c_i^\star} \coloneqq \hat{\gamma}_i \simeq 1$. In this case, $r_i(\boldsymbol{x}|\mathcal{D})$ should be a constant value as $\mathbb{P}(c_i \leq c_i^\star|\boldsymbol{x}, \mathcal{D}) = 1$ holds for almost all configurations $\boldsymbol{x}$; however, $\mathbb{P}(c_i \leq c_i^\star|\boldsymbol{x}, \mathcal{D}) = 1$ cannot be exactly obtained with a finite sample $\mathcal{D}$. On the other hand, when we use the relative density ratio $r_i^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D})$, this issue will be resolved, and it can be written more formally as follows:

Theorem 1 *Given a pair of constraint thresholds $c_i^\star, c_j^\star$ and the corresponding quantiles $\hat{\gamma}_i, \hat{\gamma}_j (\hat{\gamma}_i \leq \hat{\gamma}_j)$, if $r_i + \frac{\hat{\gamma}_i}{1-\hat{\gamma}_i} r_i^2 \leq r_j + \frac{\hat{\gamma}_j}{1-\hat{\gamma}_j} r_j^2$ holds, then*

$$\frac{\partial \prod_{k=0}^C r_k^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D})}{\partial r_i} \geq \frac{\partial \prod_{k=0}^C r_k^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D})}{\partial r_j} \geq 0 \tag{9}$$

holds where the first equality holds if $\hat{\gamma}_i = \hat{\gamma}_j$ and $r_i = r_j$ and the second one holds iff $\hat{\gamma}_j = 1$.

The proof is provided in Appendix C.3 and we discuss the intuition using Figure 2 later. Roughly speaking, Theorem 1 implies that our modified AF puts more priority in the variations of the density ratios with lower quantiles. Note that as $x/(1-x)$ is a monotonically increasing function in $x \in [0, 1)$ and r_i, r_j are always non-negative, the condition is always satisfied when $r_i \leq r_j$. The special case of Theorem 1 is the following corollary:

Corollary 2 *Assuming $\Gamma = 1$ and the TPE formulation, then $\prod_{i=0}^C r_i^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D}) \overset{\mathrm{affine}}{\propto} r_0(\boldsymbol{x}|\mathcal{D})$ holds.*

Recall that we previously defined $r_0(\boldsymbol{x}|\mathcal{D}) \coloneqq l(\boldsymbol{x}|\mathcal{D}_0^{(l)})/g(\boldsymbol{x}|\mathcal{D}_0^{(g)})$ where we obtain $\mathcal{D}_0^{(l)}, \mathcal{D}_0^{(g)}$ by splitting $\mathcal{D}$ based on f. The proof is provided in Appendix C.4. Corollary 2 indicates that the AF of c-TPE is equivalent to that of the original TPE when $\Gamma = 1$ and thus our formulation achieves $\mathbb{P}(c_i \leq c_i^\star|\boldsymbol{x}, \mathcal{D}) = 1$ if $\hat{\gamma}_i = 0$.

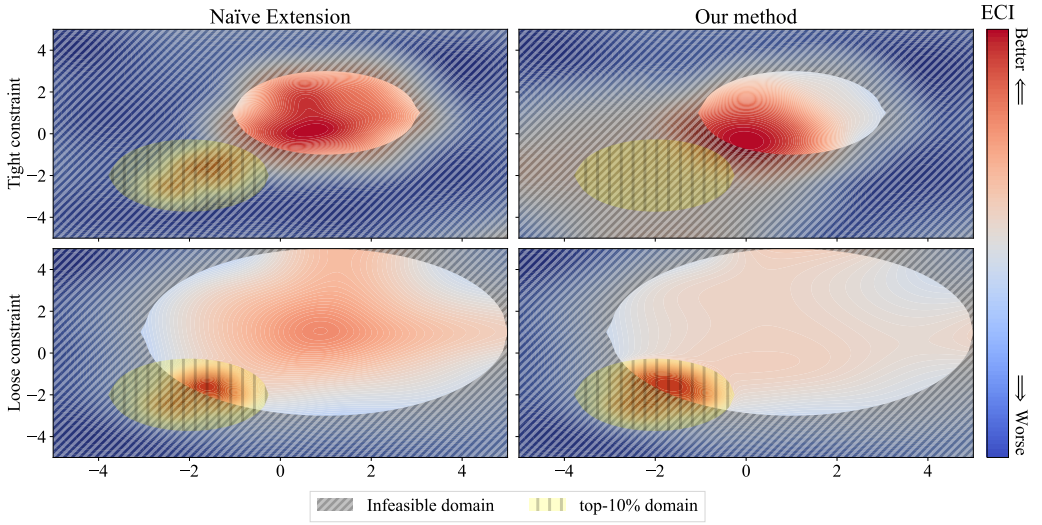

Figure 2: Heat maps of the AF in the naïve extension and our c-TPE. For fair comparisons, we use a fixed set of 200 randomly sampled configurations to compute the AF for all settings. In principle, red regions have higher AF values and the next configuration is likely to be picked from here.

We empirically present the effect of Theorem 1 in Figure 2. We used the objective function $f(x, y) = (x + 2)^2 + (y + 2)^2$ and the constraint $c_1(x, y) = (x - 1)^2 + (y - 1)^2 \leq c_1^\star \in \{4, 16\}$ and visualize the heat maps of the AF using exactly the same observations for each figure. Note that all used parameters are described in Appendix F. As mentioned earlier, the naïve extension does not decay the contribution from the objective or the constraint with a large $\hat{\gamma}_i$ and thus it has two peaks, where we have higher $r_i(\boldsymbol{x}|\mathcal{D})$, in both cases. For our algorithm, however, we only have one peak between the top-10% domain and the feasible domain because our AF decays the contribution from either the objective or the constraint based on their quantiles $\hat{\gamma}_i$ as mentioned in Theorem 1. More specifically, for the case of the tight constraint (top), since the feasible domain quantile $\hat{\gamma}_1 \simeq 0.12$ is relatively small compared to the estimated quantile $\hat{\gamma}_0 \simeq 0.3$, the peak in the top-10% domain vanishes. Notice that we discuss why we have the peak not at the center of the feasible domain, but between the feasible domain and top-10% domain in the next section. For the case of the loose constraint (bottom), $\hat{\gamma}_1 \simeq 0.50$ is much larger than $\hat{\gamma}_0 \simeq 0.02$ and this decays the contribution from the center of the feasible domain where we have the largest $r_1(\boldsymbol{x}|\mathcal{D})$. When $\Gamma = 1$, $r_i^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D})$ for $i \in \{1, \ldots, C\}$ takes 1 as mentioned in Corollary 2 and thus the AF coincides with that for the single objective optimization.

B.2.2 Issue II: Small overlaps in promising regions and feasible domains

Since the original TPE algorithm just takes the top-γ quantile observations, it does not guarantee that $\mathcal{D}^{(l)}$ has feasible solutions. For example, Figure 2 (top) does not have an overlap between the feasible domain and the top-10% domain. That is why if we employ the split algorithm from the original TPE, we will have two peaks in the AF as seen in Figure 2 (top left). On the other hand, as constraint optimization typically requires intensive sampling within feasible domains, we modify the split algorithm to include a certain number of feasible solutions. This leads to the large white circle that embraces the top-10% domain for our method while the naïve extension has only the small red circle that embraces the top-10% domain as in Figure 2 (top). As a result, our algorithm yields a peak at the overlap between the large white circle and the feasible domain.

We will demonstrate how our algorithm and the naïve extension samples configurations using a toy example. We used the objective function $f(x, y) = x^2 + y^2$ and the constraint $c_1(x, y) = (x - z)^2 + (y - z)^2 \leq c_1^\star = 3$ where $z \in \{0.5, 2.3\}$. This experiment also follows the parameters described in Appendix F and both algorithms share the initial configurations. Figure 3 (top) shows a case of a large overlap and both algorithms search similarly in this case. In contrast to this setting, the small overlap setting obtained different behaviors. While our algorithm samples intensively at the boundary of the feasible domain, the naïve extension does not. Furthermore, we can see a

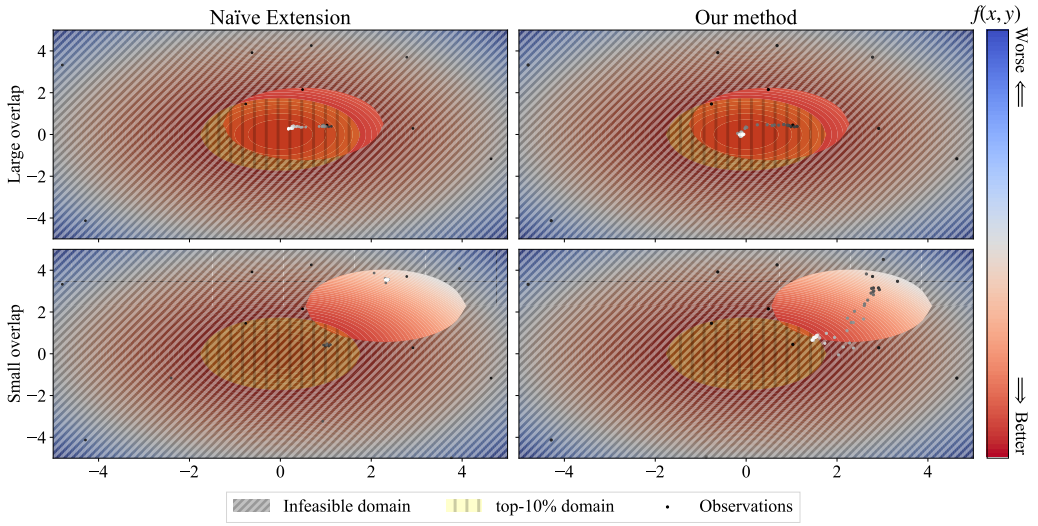

Figure 3: Scatter plots of observations obtained by the naïvely extended TPE and our c-TPE. Each figure shows the 2D search space for each task and the observations obtained during optimization are plotted. Earlier observations are colored black and later observations are colored white. Each figure has 50 observations.

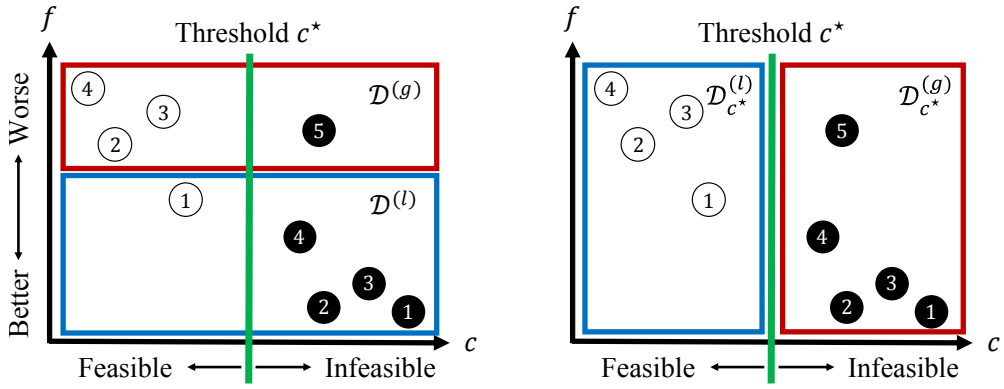

Figure 4: The conceptual visualizations of the split algorithm for the objective (left) and for each constraint (right). The black circles in the figures represent infeasible solutions and the white circles represent feasible solutions. The numberings for white and black objects stand for the ranking of the objective value in feasible and infeasible domains, respectively. The configurations enclosed by the red rectangle belong to the bad group and those enclosed by the blue rectangle belong to the good group.

trajectory from the top right of the feasible domain to the boundary between the feasible domain and the top-10% domain for our algorithm. This happens only to our algorithm although both methods have some observations in the top right of the feasible domain. Based on Figure 2 (top right), we can infer that this is because we include some feasible solutions in $\mathcal{D}_0^{(l)}$ and the peak of the AF will be shifted towards the top-10% domain in our algorithm.

B.3 Further details of the split algorithms

B.3.1 Split algorithm of objective

Figure 4 presents how to split observations into good and bad groups. The left figure shows the split for the objective. In this example there are $N = 9$ observations and thus we will include $\lceil \sqrt{N}/4 \rceil = \lceil \sqrt{9}/4 \rceil = 1$ feasible solution in $\mathcal{D}^{(l)}$. For this reason, we first need to find the feasible observation with the best objective value and the white-circled observation 1 in the figure is the

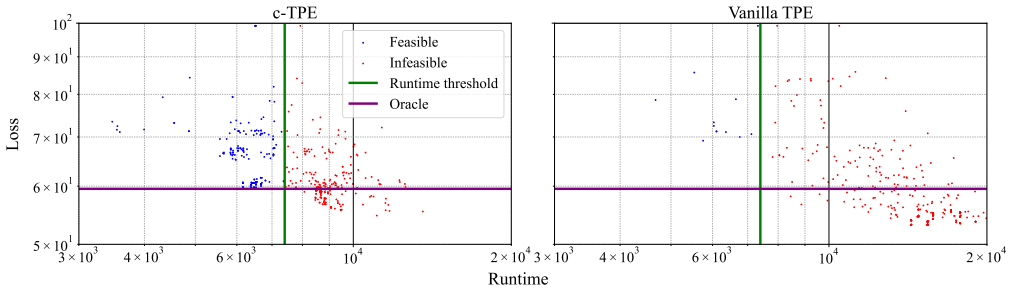

Figure 5: The visualization of the observations obtained by c-TPE (left) and the vanilla TPE (right) in the optimization of NAS-Bench-201 on ImageNet16-120 with $\gamma_i^{\text{true}} = 0.1$. We include the observations in the latter half of each optimization to see the pure learning effect of each method and each optimization was run five times. Runtime threshold is chosen so that $\gamma_i^{\text{true}} = 0.1$ will hold and oracle is the best loss value that can be achieved given a constraint value. The red dots are the observations that belong to the infeasible domain and the blue dots are the observations that belong to the feasible domain.

corresponding observation in this example. Then we split observations at the white observation 1 along the horizontal axis and $\mathcal{D}^{(l)}$ and $\mathcal{D}^{(g)}$ are obtained. The reasons behind this modification are from the fact that observations with the best objective values are often, especially for tighter constraints, far from the feasible domain (e.g. the black-circled observations 1 and 3 in Figure 4) and it guarantees at least one feasible observation to be in the good group unless there is no feasible observation. For example, Figure 5 visualizes the observations by c-TPE and the vanilla TPE on ImageNet16-120 of NAS-Bench-201 with $\gamma_i^{\text{true}} = 0.1$. As seen in the figure, there are many observations with better performance than the oracle that are far from the feasible domain in the result of the vanilla TPE. When c-TPE prioritizes only such observations, c-TPE ends up searching the infeasible domain. For this reason, we consider the splits by the number of feasible observations.

B.3.2 Split algorithm of each constraint

The right figure of Figure 4 shows the split of each constraint. Note that for simplicity, we show the 1D example and abbreviate $c_i, c_i^\star, \mathcal{D}_i^{(l)}, \mathcal{D}_i^{(g)}$ as $c, c^\star, \mathcal{D}_{c^\star}^{(l)}, \mathcal{D}_{c^\star}^{(g)}$, respectively. As illustrated in the figure, we take the observations with constraint values less than $c^\star$ into $\mathcal{D}_{c^\star}^{(l)}$ and vice versa. When the observations in the feasible domain do not exist, we only take the observation with the best constraint value among all the observations into $\mathcal{D}_{c^\star}^{(l)}$ and the rest into $\mathcal{D}_{c^\star}^{(g)}$. This selection increases the priority of this constraint as mentioned in Theorem 1 and thus raises the probability of yielding feasible solutions quickly.

C Proofs

C.1 Proof of Proposition 1

Proof 1 *Using the definition of $p(\boldsymbol{x}|f, \mathcal{D})$ in Eq. (3), PI is computed as:*

$$\mathbb{P}(f \leq f^\star \mid \boldsymbol{x}, \mathcal{D}) = \int_{-\infty}^{f^\star} p(f|\boldsymbol{x}, \mathcal{D}) df = \int_{-\infty}^{f^\star} \frac{p(\boldsymbol{x}|f, \mathcal{D}) p(f|\mathcal{D})}{p(\boldsymbol{x}|\mathcal{D})} df = \frac{l(\boldsymbol{x}|\mathcal{D}^{(l)})}{p(\boldsymbol{x}|\mathcal{D})} \int_{-\infty}^{f^\star} p(f|\mathcal{D}) df. \tag{10}$$

Notice that $\mathcal{D}$ is split by $f^\star$. Since the EI for TPE is computed as:

$$\mathrm{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}] = \frac{l(\boldsymbol{x}|\mathcal{D}^{(l)})}{p(\boldsymbol{x}|\mathcal{D})} \int_{-\infty}^{f^\star} (f^\star - f) p(f|\mathcal{D}) df \tag{11}$$

and both equations have the common part $l(\boldsymbol{x}|\mathcal{D}^{(l)})/p(\boldsymbol{x}|\mathcal{D})$. This part cancels out when we take the ratio. For this reason, the ratio of the two equations is computed as:

$$\frac{\int_{-\infty}^{f^\star} (f^\star - f) p(f|\mathcal{D}) df}{\int_{-\infty}^{f^\star} p(f|\mathcal{D}) df} = \text{const w.r.t. } \boldsymbol{x}, \tag{12}$$

where, since we assume that the support of $\mathbb{P}(f \leq f^\gamma|\boldsymbol{x}, \mathcal{D})$ covers the whole domain $\mathcal{X}$, i.e. $\forall \boldsymbol{x} \in \mathcal{X}, \mathbb{P}(f \leq f^\star|\boldsymbol{x}, \mathcal{D}) \neq 0$ and f is Lebesgue integrable, i.e. the expectation of f exists and $\int |f|\mu(d\boldsymbol{x}) < \infty$, both numerator and denominator always take a positive finite value and thus the LHS of Eq. (12) takes a finite positive constant value.

C.2 Proof of Corollary 1

Proof 2 *Under the TPE formulation, $\mathrm{EI}_{f^\star}(\boldsymbol{x}|\mathcal{D})$ is proportional to $r_0^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D})$ and $\mathrm{EI}_{c_i^\star}(\boldsymbol{x}|\mathcal{D})$ is proportional to $r_i^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D})$ as shown by Bergstra* et al. *(2011). Furthermore, since EI and PI are equivalent in the TPE formulation from Proposition 1, ECI for TPE satisfies the following using Eq. (8):*

$$\mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}] \propto \mathbb{P}(f \leq f^\star|\boldsymbol{x}, \mathcal{D}) \prod_{i=1}^{C} \mathbb{P}(c_i \leq c_i^\star|\boldsymbol{x}, \mathcal{D}) \propto \prod_{i=0}^{C} r_i^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D}). \tag{13}$$

This completes the proof.

C.3 Proof of Theorem 1

Proof 3 *From Corollary 1, $\mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}] \propto \prod_{k=0}^{C} r_k^{\mathrm{rel}}$ holds and thus the partial derivative of the RHS with respect to the density ratio $r_k(\boldsymbol{x})$ for $k \in \{0, \ldots, C\}$ is computed as follows:*

$$\begin{aligned}
\frac{\partial \mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}]}{\partial r_k} &\propto \frac{\partial r_k^{\mathrm{rel}}}{\partial r_k} \prod_{k' \neq k} r_{k'}^{\mathrm{rel}} \\
&= \frac{\partial}{\partial r_k} \frac{1}{\hat{\gamma}_k + (1 - \hat{\gamma}_k) r_k^{-1}} \prod_{k' \neq k} r_{k'}^{\mathrm{rel}} \\
&= \frac{1 - \hat{\gamma}_k}{(\hat{\gamma}_k r_k + 1 - \hat{\gamma}_k)^2} \prod_{k' \neq k} r_{k'}^{\mathrm{rel}} \\
&= \frac{1 - \hat{\gamma}_k}{r_k^2} r_k^{\mathrm{rel}} \prod_{k'=0}^{C} r_{k'}^{\mathrm{rel}} \geq 0 \\
&(\because \forall k' \in \{0, \ldots, C\}, r_{k'}, r_{k'}^{\mathrm{rel}} > 0, 0 \leq 1 - \hat{\gamma}_k < 1).
\end{aligned} \tag{14}$$

For this reason, the LHS takes zero if and only if $\hat{\gamma}_k = 1$. Using the result, the following holds with a positive constant number α:

$$\begin{aligned}
\frac{\partial \mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}]}{\partial r_i} - \frac{\partial \mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}]}{\partial r_j} &= \alpha \left(\frac{1 - \hat{\gamma}_i}{r_i^2} r_i^{\mathrm{rel}} - \frac{1 - \hat{\gamma}_j}{r_j^2} r_j^{\mathrm{rel}} \right) \\
&= \alpha \left(\frac{1 - \hat{\gamma}_i}{\hat{\gamma}_i r_i^2 + (1 - \hat{\gamma}_i) r_i} - \frac{1 - \hat{\gamma}_j}{\hat{\gamma}_j r_j^2 + (1 - \hat{\gamma}_j) r_j} \right) \\
&= \alpha \left(\left(r_i + \frac{\hat{\gamma}_i}{1 - \hat{\gamma}_i} r_i^2 \right)^{-1} - \left(r_j + \frac{\hat{\gamma}_j}{1 - \hat{\gamma}_j} r_j^2 \right)^{-1} \right).
\end{aligned} \tag{15}$$

Since $r_i, r_j > 0$ holds and

$$r_i + \frac{\hat{\gamma}_i}{1 - \hat{\gamma}_i} r_i^2 \geq r_j + \frac{\hat{\gamma}_j}{1 - \hat{\gamma}_j} r_j^2, \tag{16}$$

$$\frac{\partial \mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}]}{\partial r_i} - \frac{\partial \mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}]}{\partial r_j} = \alpha \left(\left(r_i + \frac{\hat{\gamma}_i}{1 - \hat{\gamma}_i} r_i^2 \right)^{-1} - \left(r_j + \frac{\hat{\gamma}_j}{1 - \hat{\gamma}_j} r_j^2 \right)^{-1} \right) \geq 0 \tag{17}$$

holds. When we assume $\hat{\gamma}_i = \hat{\gamma}_j$ and $r_i = r_j$, we get the equality. This completes the proof.

Note that since $x/(1-x)$ is a monotonically increasing function in $x \in [0, 1)$ and $\hat{\gamma}_i \leq \hat{\gamma}_j$ from the assumption,

$$0 \leq \alpha_i \coloneqq \frac{\hat{\gamma}_i}{1-\hat{\gamma}_i} \leq \alpha_j \coloneqq \frac{\hat{\gamma}_j}{1-\hat{\gamma}_j} \tag{18}$$

holds. Furthermore, using $r_i, r_j > 0$, if we assume $r_i < r_j$, then $r_i < r_j, r_i^2 < r_j^2, \alpha_i < \alpha_j$ and thus the partial derivative for the i-th constraint is larger; therefore, r_j must be smaller than r_i for its contribution to be larger than that from r_i. It implies that we will not put more priority on the constraints with large feasible domains unless those constraints are likely to be violated, which means the density ratios for those constraints are small.

C.4 Proof of Corollary 2

To prove Corollary 2, we first show two lemmas.

Lemma 1 *Given a Γ-feasible domain ($\Gamma > 0$) with constraint thresholds of $c_i^\star$ for all $i \in \{1, \ldots, C\}$, each constraint satisfies*

$$\forall i \in \{1, \ldots, C\}, \gamma_i \geq \Gamma. \tag{19}$$

Proof 4 *Let the feasible domain for the i-th constraint be $\mathcal{X}_i' = \{\boldsymbol{x} \in \mathcal{X} | c_i \leq c_i^\star\}$. Then the feasible domain is $\mathcal{X}' = \bigcap_{i=1}^C \mathcal{X}_i'$. Since $\mathcal{X}_i'$ is a measurable set by definition and $\mathcal{X}' \subseteq \mathcal{X}_i'$ holds, $\Gamma/\gamma_i = \mu(\mathcal{X}')/\mu(\mathcal{X}_i') \leq 1$ holds. Γ is a positive number, so $\gamma_i \geq \Gamma$ and this completes the proof.*

Lemma 2 *The domain is $(\Gamma = 1)$-feasible domain iff:*

$$\forall i \in \{1, \ldots, C\}, \gamma_i = 1. \tag{20}$$

Proof 5 *Suppose $\gamma_i < 1$ for some $i \in \{1, \ldots, C\}$, we immediately obtain $\Gamma \leq \gamma_i < 1$ from Lemma 1 and thus the assumption does not hold. For this reason, $\gamma_i \geq 1$ for all $i \in \{1, \ldots, C\}$ and since $\gamma_i \leq 1$ by definition, $\gamma_i = 1$ for all $i \in \{1, \ldots, C\}$.*

Using Lemma 2 and Theorem 1, we prove Corollary 2.

Proof 6 *From Lemma 2, when $\Gamma = 1$, $\gamma_i = 1$ for all $i \in \{1, \ldots, C\}$ and we plug $\gamma_i = 1$ into Theorem 1. Then we obtain:*

$$\forall i \in \{1, \ldots, C\}, \frac{\partial \mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}]}{\partial r_i} = 0. \tag{21}$$

For this reason, $\mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}] \propto \prod_{i=0}^C r_i^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D}) \propto r_0^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D}) \overset{\mathrm{affine}}{\propto} r_0(\boldsymbol{x}|\mathcal{D})$ and this completes the proof.

D Related work and discussion

The ECI was introduced by Gardner *et al.* (2014) and Gelbart *et al.* (2014). Furthermore, there are various extensions of these prior works. For example, Letham *et al.* (2019) (NEI) is more robust to the noise caused in experiments and Eriksson and Poloczek (2021) is scalable to high dimensions. Another technique for constraint BO is entropy search, such as predictive entropy search (Lobato *et al.* (2015)) and max-value entropy search (Perrone *et al.* (2019)). They choose the next configuration by approximating the expected information gain on the value of the constrained minimizer. Note that we could not compare with those methods except NEI in our experiments because the implementations are not provided in the papers. While these settings use Gaussian process (GP) regression to compute the AF, our method uses TPE. The major advantages of TPE over naïve GP-based BOs are more natural handling of categorical and conditional parameters and easier integration of cheap-to-evaluate partial observations due to the linear time complexity with respect to $|\mathcal{D}|$. Notice that we provided the concept of the integration of partial observations and its results, which showed a further acceleration of c-TPE, in Appendix E. Regarding the first advantage, although there are several BO methods that

handle categorical parameters explicitly (Daxberger *et al.* (2019); Deshwal *et al.* (2021); Ru *et al.* (2020)), none of them is available as constraint optimization packages [1]. Another option is to apply one-hot encoding (Garrido-Merchán and Hernández-Lobato (2020)) to categorical parameters as Facebook Ax (Letham *et al.* (2019)) does; however, this approach did not work well in our settings. Considering the fact that c-TPE is easily integratable to actively maintained open source softwares and the results in the experiments, c-TPE is a desirable optimization method.

Evolutionary algorithm (EA) is another domain where constraint optimization has been studied actively, such as genetic algorithms (e.g. CNSGA-II (Deb *et al.* (2002))), CMA-ES (Arnold and Hansen (2012)) or differential evolution (Montes *et al.* (2006)). Although CMA-ES has demonstrated the best performance among more than 100 methods for various black-box optimization problems (Loshchilov *et al.* (2013)), it does not support categorical parameters. Furthermore, since EAs have many control parameters, such as mutation rate and population size, it requires meta-tuning. Another downside of EAs is that it is hard to integrate partial observations because EAs require all the metrics to rank each configuration at each iteration. In general, BO overcomes these difficulties as discussed in Appendix E.

E Integration of partial observations

In this section, we discuss the integration of partial observations for BO and we name the integration "Knowledge augmentation".

E.1 Knowledge augmentation

When some constraints can be precisely evaluated with a negligible amount of time compared to others, practitioners typically would like to use Knowledge augmentation (KA). For example, the network size of deep learning models is trivially computed in seconds while the final validation performance requires several hours to days. In this case, we can obtain many observations only for network size and augment the knowledge of network size prior to the optimization so that the constraint violations will be reduced in the early stage of optimizations.

To validate the effect of KA, all of the additional results in the appendix include the results obtained using c-TPE with KA. In the experiments, we augmented the knowledge only for network size and we did not include runtime as a target of KA because although runtime can be roughly estimated from a 1-epoch training, such estimations are not precise. However, practitioners can include such rough estimations into partial observations as long as they can accept errors caused by them.

E.2 Algorithm of knowledge augmentation

Algorithm 2 is the pseudocode of c-TPE with KA. We first need to specify a set of indices for cheap constraints $I = \{i_j\}_{j=1}^{C_p}$ where $C_p (< C)$ is the number of cheap constraints and I must be an element of the power set of $\{j\}_{j=1}^{C}$. In Lines 4–6, we first collect partial observations $\mathcal{D}_p$. Then we augment observations in Lines 12–15 if partial observations are available for the corresponding constraint. We denote the augmented set of observations $\mathcal{D}_{\mathrm{aug}}$. When the acquisition function follows Eq. (6), the predictive models for each constraint are independently trained due to conditional independence. It enables us to introduce different amounts of observations for each constraint. Since c-TPE follows Eq. (6), we can employ KA. As discussed in Appendix D, it is hard to apply KA to evolutionary algorithms due to their algorithm nature and KA causes a non-negligible bottleneck for GP-based BO as the number of observations grows.

E.3 Empirical results of knowledge augmentation

In this experiment, we optimized each benchmark with a constraint for network size, and constraints for runtime and network size. To see the effect, we measured how much KA increases the chance

[1] Although we could not include those methods in our experiments as they are not extended to constraint optimization, we demonstrated that the vanilla TPE was significantly better than some recent BO methods on our settings as presented in Appendix I. Based on the results, we would expect c-TPE could be better than extensions of those methods, even if they existed, on our settings.

Algorithm 2 c-TPE with knowledge augmentation

$N_{\text{init}}, N_s, N_p$ ▷ Control parameters
$I = \{i_j\}_{j=1}^{C_p}$ ▷ Indices of cheap constraints
$\mathcal{D}_p \leftarrow \emptyset, \mathcal{D} \leftarrow \emptyset$
for $n = 1, \ldots, N_p$ **do** ▷ Collect cheap information
    Randomly pick $\boldsymbol{x}$
    $\mathcal{D}_p \leftarrow \mathcal{D}_p \cup \{(\boldsymbol{x}, f(\boldsymbol{x}), c_{i_1}(\boldsymbol{x}), \ldots, c_{i_{C_p}}(\boldsymbol{x}))\}$
for $n = 1, \ldots, N_{\text{init}}$ **do**
    Randomly pick $\boldsymbol{x}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}, f(\boldsymbol{x}), c_1(\boldsymbol{x}), \ldots, c_C(\boldsymbol{x}))\}$
while Budget is left **do**
    for $i = 0, \ldots, C$ **do**
        if $i \in I$ **then**
            $\mathcal{D}_{\text{aug}} = \mathcal{D} \cup \mathcal{D}_p$
        else
            $\mathcal{D}_{\text{aug}} = \mathcal{D}$
        Split $\mathcal{D}_{\text{aug}}$ into $\mathcal{D}_i^{(l)}$ and $\mathcal{D}_i^{(g)}$, $\gamma_i \leftarrow |\mathcal{D}_i^{(l)}|/|D_{\text{aug}}|$
        Build $l(\boldsymbol{x}|\mathcal{D}_i^{(l)}), g(\boldsymbol{x}|\mathcal{D}_i^{(g)})$
        $\{\boldsymbol{x}_j\}_{j=1}^{N_s} \sim l(\boldsymbol{x}|\mathcal{D}_i^{(l)}), \mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{x}_j\}_{j=1}^{N_s}$
    Pick $\boldsymbol{x}_{\text{opt}} \in \text{argmax}_{\boldsymbol{x} \in \mathcal{S}} \prod_{i=0}^{C} r_i^{\text{rel}}(\boldsymbol{x}|\mathcal{D})$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}_{\text{opt}}, f(\boldsymbol{x}_{\text{opt}}), c_1(\boldsymbol{x}_{\text{opt}}), \ldots, c_C(\boldsymbol{x}_{\text{opt}}))\}$

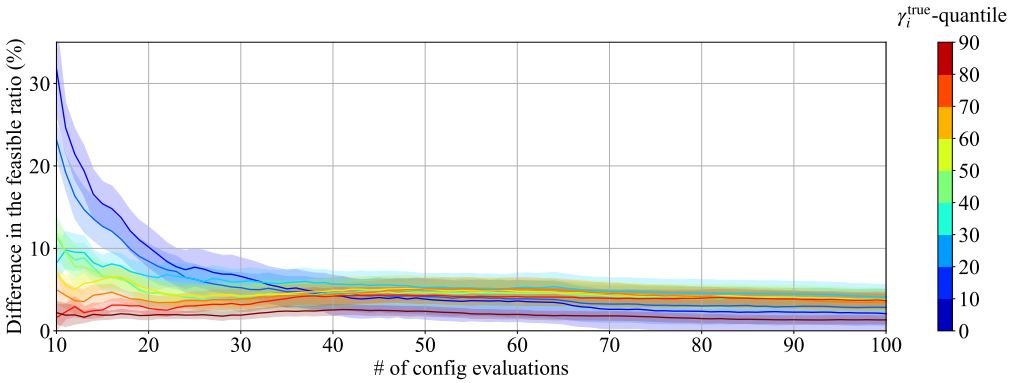

Figure 6: The effect of KA in the optimizations with a constraint for network size. The horizontal axis shows the number of evaluated configurations in optimizations and the vertical axis shows the difference in the cumulated ratio of feasible solutions between c-TPE and c-TPE with KA using 200 randomly sampled configurations. The weak-color bands show the standard error of mean values of 50 runs for 10 benchmarks.

of drawing feasible solutions and tested the performance difference by the Wilcoxon signed-rank test on 18 settings (9 benchmarks $\times$ 2 constraint choices). According to Figure 6, the tighter the constraint becomes, the more KA helps to obtain feasible solutions, especially in the early stage of the optimizations. Additionally, Table 2 shows the statistically significant speedup effects of KA in $\gamma_i^{\text{true}} = 0.1$. Although KA did not exhibit the significant speedup in loose constraint levels, it did not deteriorate the optimization quality significantly. At the later stage of the optimizations, the effect gradually decays as c-TPE becomes competent enough to detect violations. In summary, KA significantly accelerates optimizations with tight constraints and it does not deteriorate the optimization quality in general, so it is practically recommended to use KA as much as possible.

Table 2: The table shows (Wins/Loses/Ties) of c-TPE with KA against c-TPE for optimizations with different constraint levels. Bold numbers indicate $p < 0.05$ of the hypothesis "c-TPE is better than c-TPE with KA" by the Wilcoxon signed-rank test.

Quantiles	$\gamma_i^{\text{true}} = 0.1$				$\gamma_i^{\text{true}} = 0.5$				$\gamma_i^{\text{true}} = 0.9$			
# of configs	50	100	150	200	50	100	150	200	50	100	150	200
Wins/Loses/Ties	**12/5/1**	**11/5/2**	7/5/6	6/6/6	6/12/0	5/11/2	7/6/5	5/5/8	9/9/0	10/6/2	8/5/5	6/9/3

F Experiment settings

F.1 The choice of γ_i^{true}

As mentioned earlier, we chose tabular benchmarks because they enable us to control the quantiles of each constraint γ_i^{true}. In practice, we do not have the access to γ_i^{true} because γ_i^{true} will not be identified unless we evaluate all the possible configurations. On the other hand, the quality of solutions heavily relies on γ_i^{true}. Since we can calculate the quantile γ_i^{true} by looking up all the results, we evaluate each method on tabular benchmarks with various γ_i^{true} to observe the performance variation of each method on different difficulties. For example, suppose a tabular dataset has N configurations $\{(\boldsymbol{x}_n, f_n, \boldsymbol{c}_n)\}_{n=1}^N$ and the dataset is sorted so that it satisfies $c_{i,1} \leq c_{i,2} \leq \cdots \leq c_{i,N}$ where $c_{i,n}$ is the i-th constraint value in the n-th configuration, then we fix the threshold for the i-th constraint $c_i^\star$ as $c_{i,\lfloor N/10 \rfloor}$ in the setting of $\gamma_i^{\text{true}} = 0.1$.

F.2 Tabular benchmarks

The evaluations were performed on the following 10 benchmarks:

1. HPOlib (slice localization, naval propulsion, parkinsons telemonitoring, protein structure) (Klein and Hutter (2019)): All with 6 numerical and 3 categorical parameters;
2. NAS-Bench-101 (CIFAR10A, CIFAR10B, CIFAR10C) (Ying *et al.* (2019)): Each with 26 categorical, 14 categorical, and 22 numerical and 5 categorical parameters, respectively; and
3. NAS-Bench-201 (ImageNet16-120, CIFAR10, CIFAR100) (Dong and Yang (2020)): All with 6 categorical parameters.

We evaluated each benchmark with 9 different quantiles $\gamma_{c_i^\star}^{\text{true}}$ for each constraint and 3 different constraint choices. Constraint choices are network size, runtime, or both. The search space for each benchmark followed Awad *et al.* (2021).

F.3 Baseline optimizers

As the baseline methods, we chose:

1. **Random search** (Bergstra and Bengio (2012))
2. **CNSGA-II** (Deb *et al.* (2002)) [2] (population size 8)
3. **NEI provided in Facebook Ax** (Letham *et al.* (2019)) [3]
4. **Hypermapper2.0 (HM2)** (Nardi *et al.* (2019)) [4]
5. **Vanilla TPE** (Optimize only loss as if we do not have constraints)
6. **Naïve c-TPE** (The naïve extension discussed in Section 2)

For all the methods using TPE, we used $N_{\text{s}} = 24$ and $N_{\text{init}} = 10$, which we obtain from the ratio (5%) of the initial sample size and the number of evaluations, as in the original paper (Bergstra *et al.* (2013)). Furthermore, we employed the multivariate kernel and its bandwidth selection used by Falkner *et al.* (2018). Due to this modification, our vanilla TPE implementation performs significantly better than Hyperopt (Bergstra *et al.* (2013)) [5] on our experiment settings and thus we would like to stress that our TPE may produce better results compared to what we can expect from prior

[2] Implementation: `https://github.com/optuna/optuna`

[3] Implementation: `https://github.com/facebook/Ax`

[4] Implementation: `https://github.com/luinardi/hypermapper`

[5] Implementation: `https://github.com/hyperopt/hyperopt`

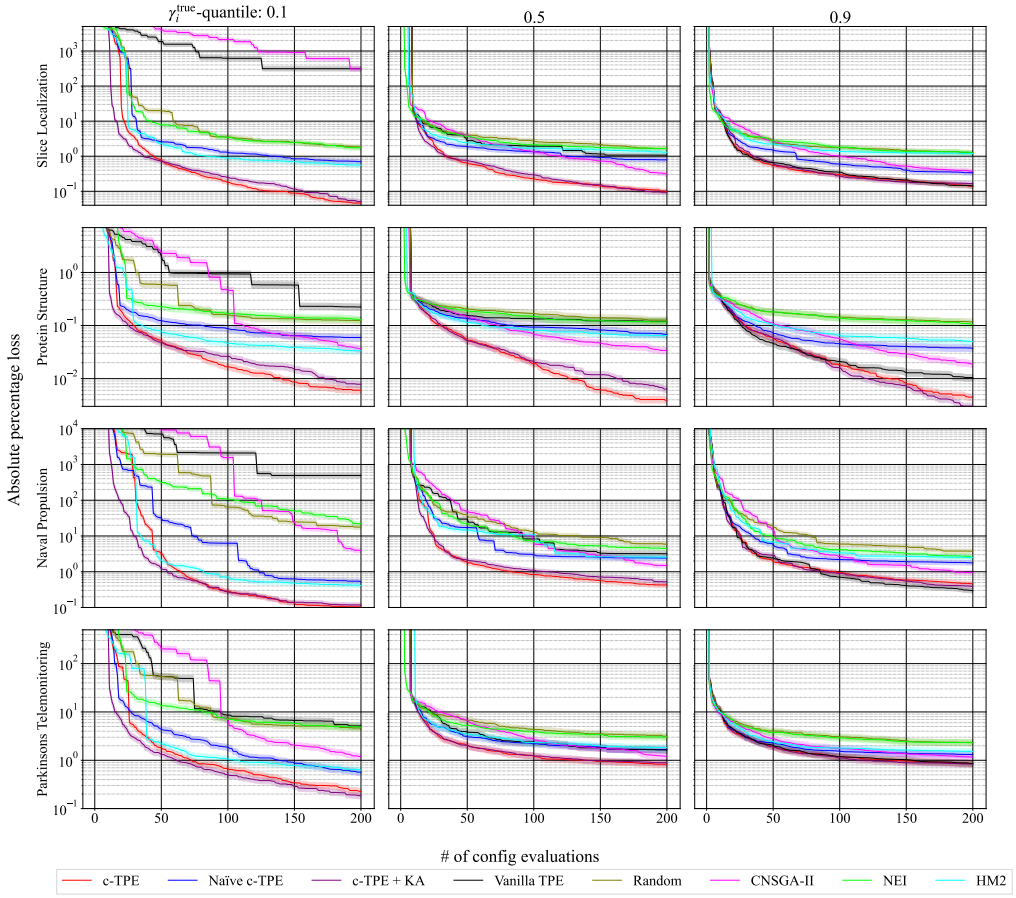

Figure 7: Figures show the performance curves on four benchmarks in HPOlib with a constraint of network size. We picked $\gamma_i^{\text{true}} = 0.1$ (left), 0.5 (center), 0.9 (right). The horizontal axis shows the number of evaluated configurations in optimizations and the vertical axis shows the absolute percentage error in each experiment.

works such as Daxberger *et al.* (2019); Deshwal *et al.* (2021); Eggensperger *et al.* (2013); Ru *et al.* (2020); Turner *et al.* (2021). For more details, see Appendix I. CNSGA-II is a genetic algorithm based constraint optimization method, NEI is a GP-based constraint BO method with EI for noisy observations, and HM2 is a random-forest-based constraint BO method with ECI, which implements major parts of SMAC (Lindauer *et al.* (2021)) to perform constraint optimization. The vanilla TPE is evaluated in order to demonstrate the improvement of c-TPE from TPE for non-constraint settings. CNSGA-II, NEI, and HM2 followed the default settings in each package. Note that all experiments were performed over 50 random seeds and we evaluated 200 configurations for each optimization. Additionally, since the optimizations by NEI and HM2 on CIFAR10C failed due to the high-dimensional (22 dimensions) continuous search space for NEI and an unknown internal issue for HM2, we used the results on 9 benchmarks (other than CIFAR10C) for the statistical test and the average rank computation. The results on CIFAR10C by the other methods are available in Appendix G and the source code is available at `https://anonymous.4open.science/r/constraint_tpe-342C` along with complete scripts to reproduce the experiments, tables and figures.

G Additional results for the performance over time

In this section, we show the additional results for Section 3.2. The main goal of those results is to show how robust c-TPE is over various levels of constraints. Note that we picked only network size as a cheap constraint and did not pick runtime as discussed in Appendix E and we used $N_p = 200$ throughout all the experiments.

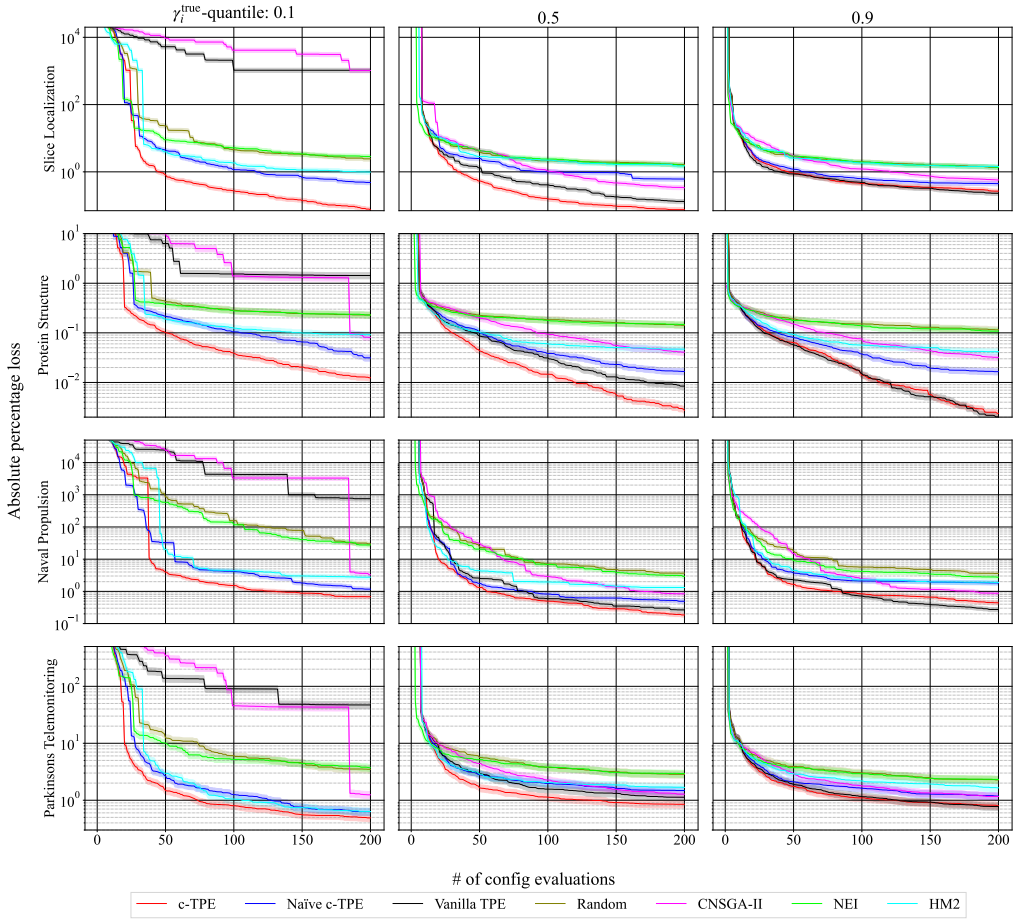

Figure 8: Figures show the performance curves on four benchmarks in HPOlib with a constraint of runtime.

G.1 Results on HPOlib

Figures 7, 8, and 9 show the time evolution of absolute percentage loss of each optimization method on HPOlib, NAS-Bench-101, and NAS-Bench-201 with the γ_i^{true}-quantile of 0.1, 0.5, and 0.9 for the network size constraint.

For tighter constraint settings, c-TPE outperformed other methods and KA accelerated c-TPE in the early stage. For looser constraint settings, CNSGA-II improves its performance in the early stage of optimizations although c-TPE still exhibits quicker convergence. On the other hand, the performance of NEI and HM2 was degraded. As mentioned in Corollary 2, c-TPE approaches the performance of the vanilla TPE in the settings of $\gamma_i^{\text{true}} = 0.9$ and thus such degradation does not happen to c-TPE. Furthermore, the contributions to the acquisition function from looser constraints decay and KA does not disrupt the performance of c-TPE thanks to this property.

For multiple constraints settings shown in Figure 9, both CNSGA-II and HM2 show slower convergence compared to single constraint settings. On the other hand, c-TPE shows quicker convergence in the settings as well.

G.2 Results on NAS-Bench-101

Figures 10, 11, and 12 show the time evolution of absolute percentage loss of each optimization method on HPOlib, NAS-Bench-101, and NAS-Bench-201 with the γ_i^{true}-quantile of 0.1, 0.5, and 0.9 for the runtime constraint. Note that since we could not run NEI and HM2 on CIFAR10C in our environment, the results for CIFAR10C do not have the performance curves of NEI and HM2.

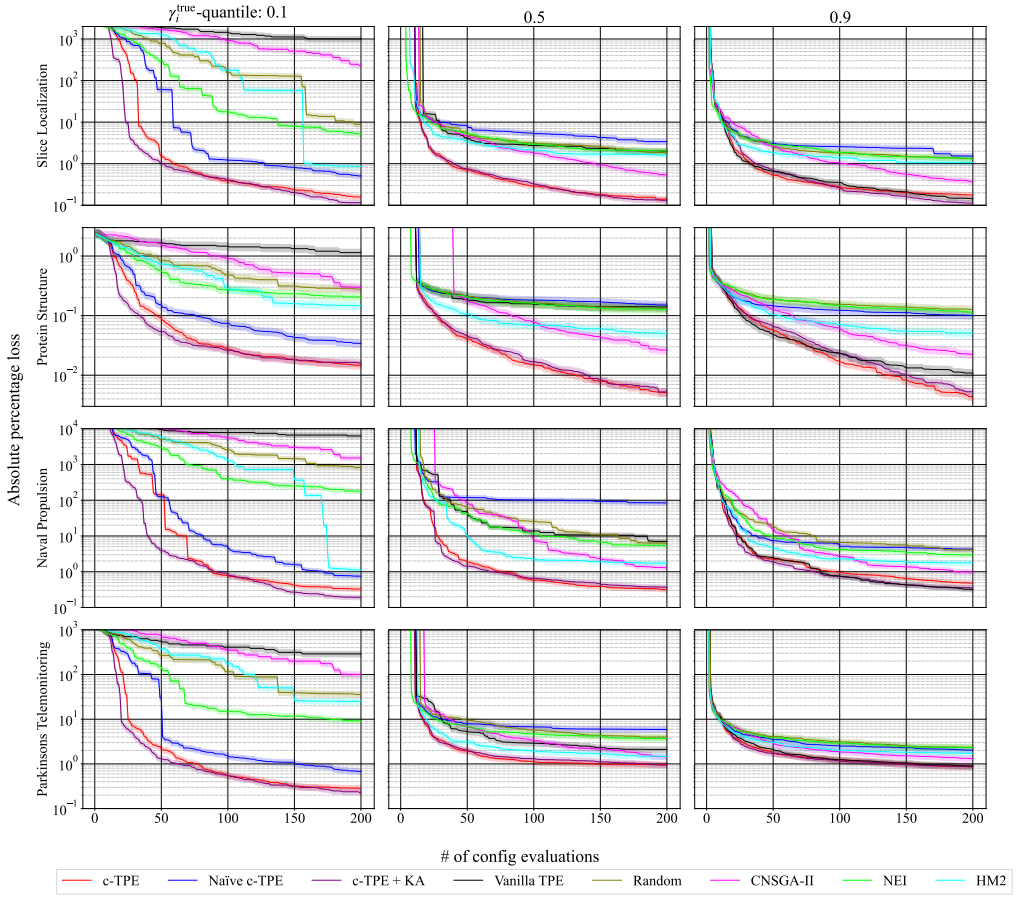

Figure 9: Figures show the performance curves on four benchmarks in HPOlib with constraints of runtime and network size.

The results on NAS-Bench-101 look different from those on HPOlib and NAS-Bench201. For example, random search outperforms other methods on the tighter constraint settings of CIFAR10C. This is because high-dimensional search space and tight constraints made the information collection harder and thus each method could not guide itself although c-TPE still outperformed other methods on average. If we add more strict constraints such that c-TPE will pick configurations from feasible domains, we could potentially achieve better results; however, it would lead to poor performance as the number of evaluations increases and thus this will be a trade-off. Additionally, KA still helps to yield better configurations quickly except CIFAR10C with runtime and network size constraints. As seen in the figures, the vanilla TPE exhibited better performance on loose constraint settings and thanks to the c-TPE's property discussed in Corollary 2, c-TPE improves its performance in loose constraint levels.

G.3 Results on NAS-Bench-201

Figures 13, 14, and 15 show the time evolution of absolute percentage loss of each optimization method on HPOlib, NAS-Bench-101, and NAS-Bench-201 with the γ_i^{true}-quantile of 0.1, 0.5, and 0.9 for the runtime and network size constraints. Note that the search space of NAS-Bench-201 is composed of six categorical parameters.

According to the figures, the discrepancy between c-TPE and the vanilla TPE is larger than HPOlib and NAS-Bench-101 settings. This means that there are many violated configurations that exhibit good performance. For this reason, the tighter constraint settings on NAS-Bench-201 are harder than the other benchmarks. However, c-TPE and HM2 showed better performance on tighter constraint settings although it may lead to cold-starting in c-TPE. Additionally, c-TPE maintained the

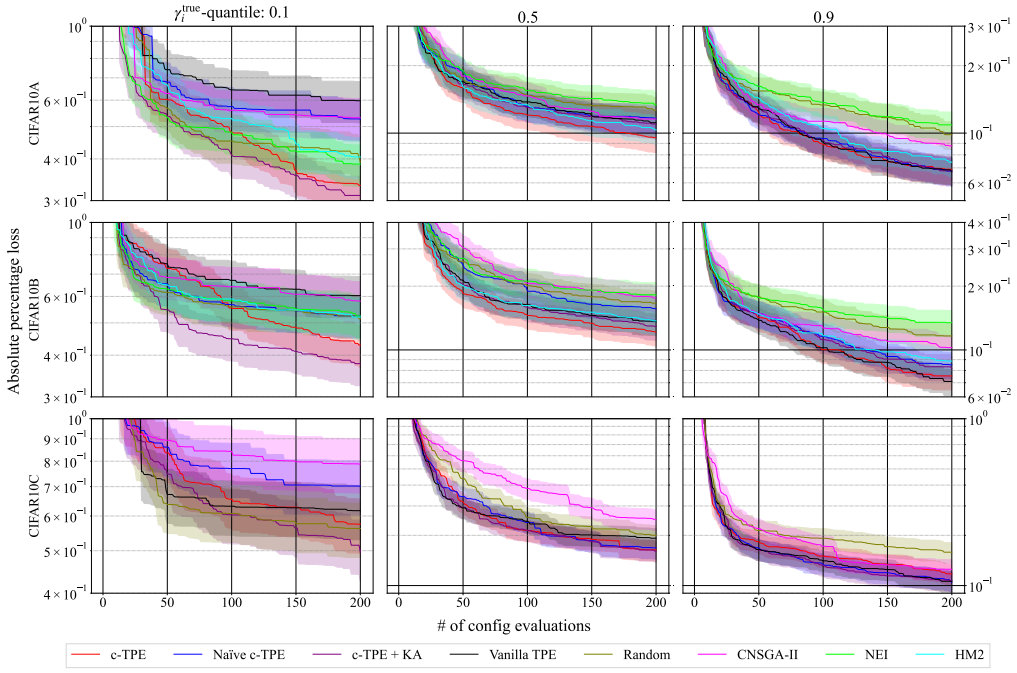

Figure 10: Figures show the performance curves on three benchmarks in NAS-Bench-101 with a constraint of network size. Note that since the scale of the results in γ_i^{true}-quantile of 0.1 is different from others, we separately scaled for the readability.

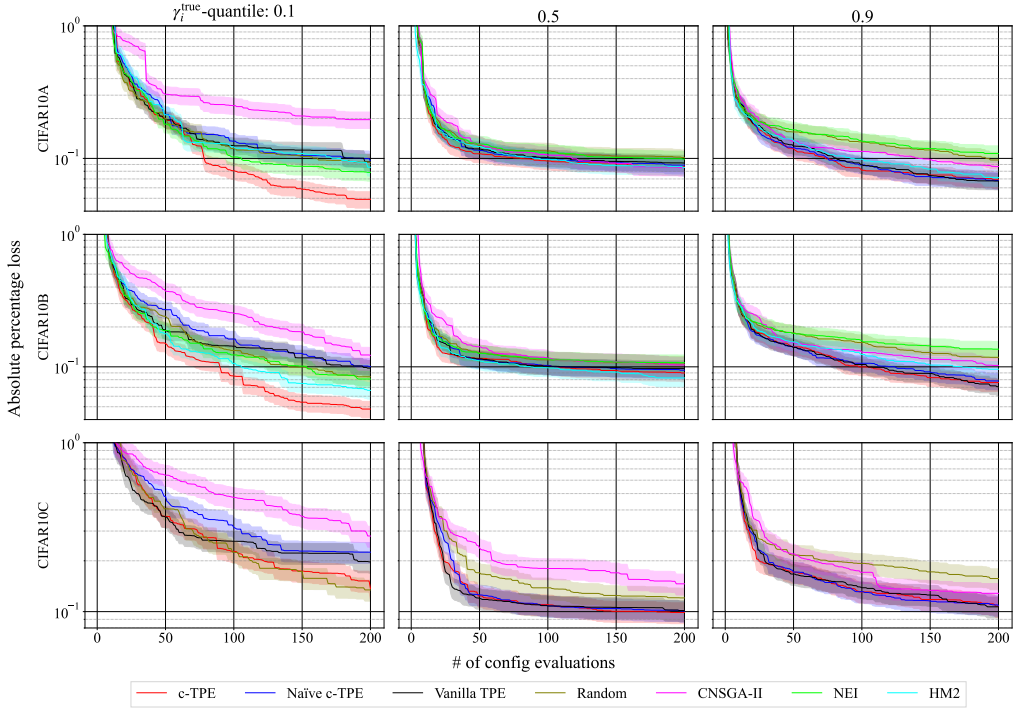

Figure 11: Figures show the performance curves on three benchmarks in NAS-Bench-101 with a constraint of runtime.

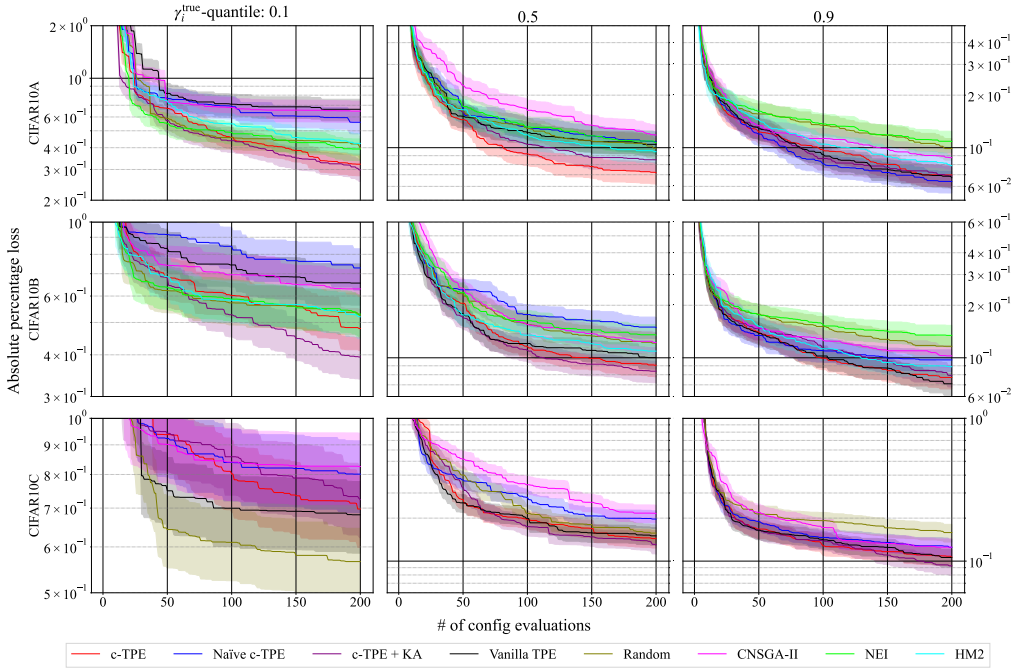

Figure 12: Figures show the performance curves on three benchmarks in NAS-Bench-101 with constraints of runtime and network size. Note that since the scale of the results in γ_i^{true}-quantile of 0.1 is different from others, we separately scaled for the readability.

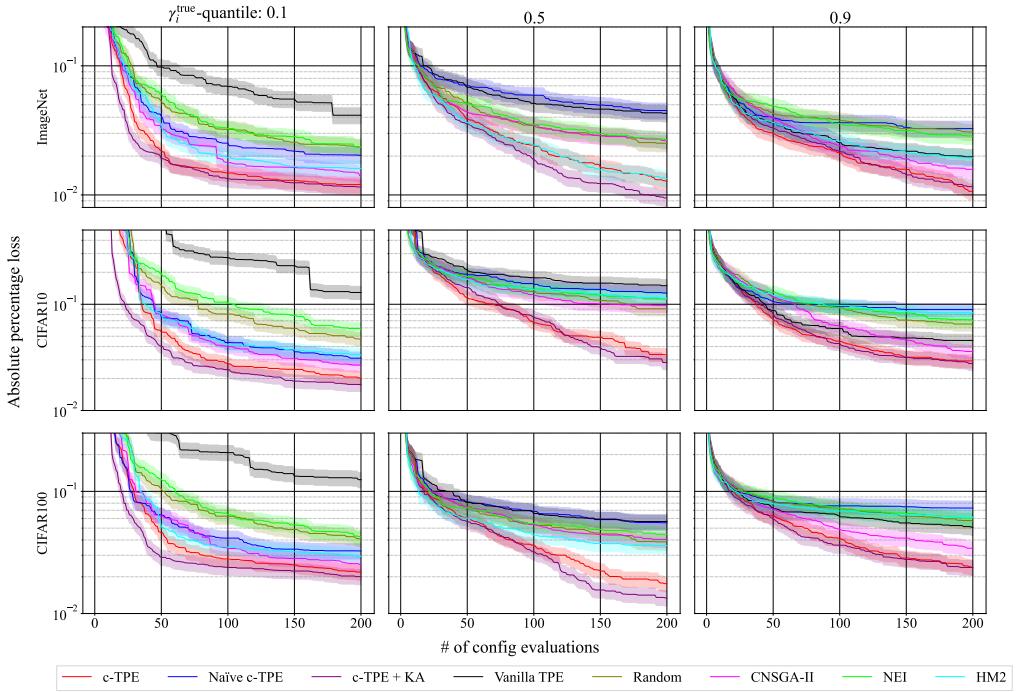

Figure 13: Figures show the performance curves on three benchmarks in NAS-Bench-201 with a constraint of network size.

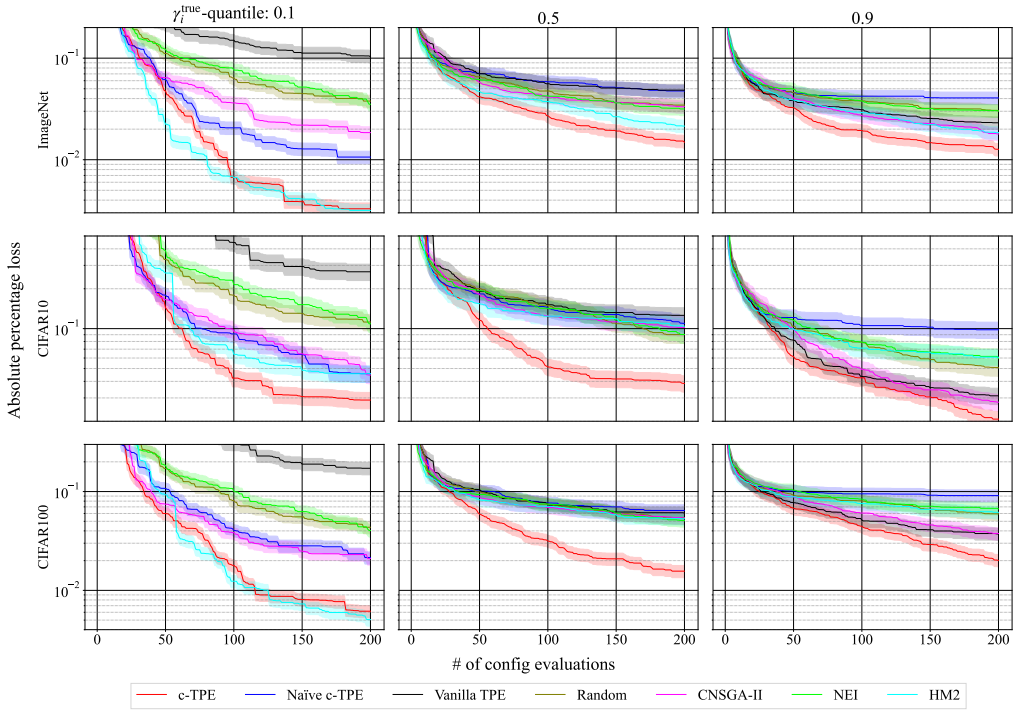

Figure 14: Figures show the performance curves on three benchmarks in NAS-Bench-201 with a constraint of runtime.

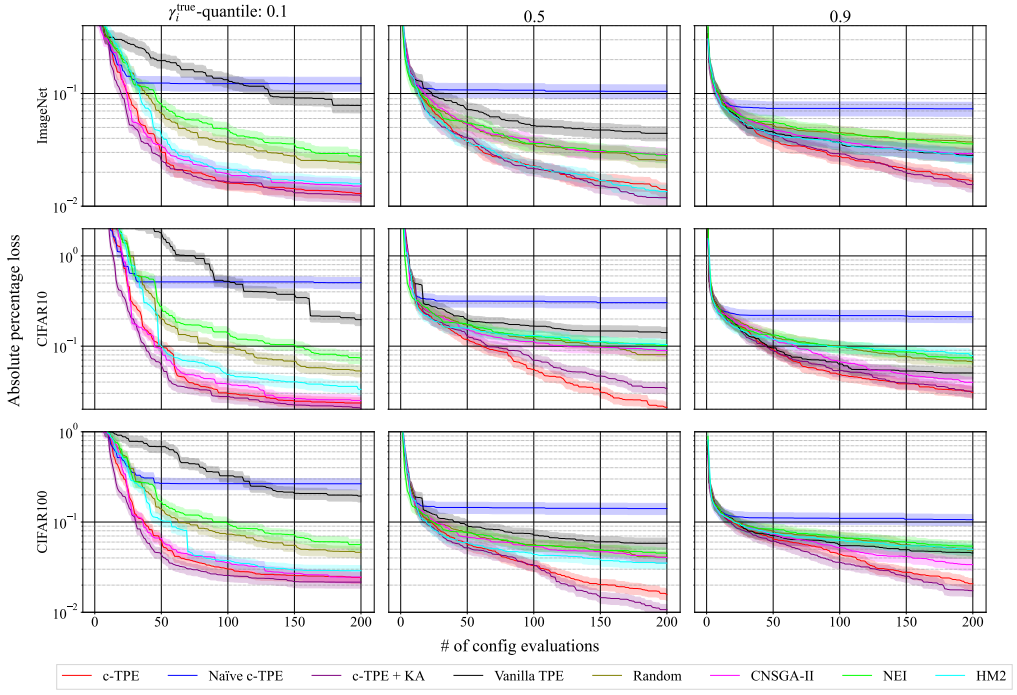

Figure 15: Figures show the performance curves on three benchmarks in NAS-Bench-201 with constraints of runtime and network size.

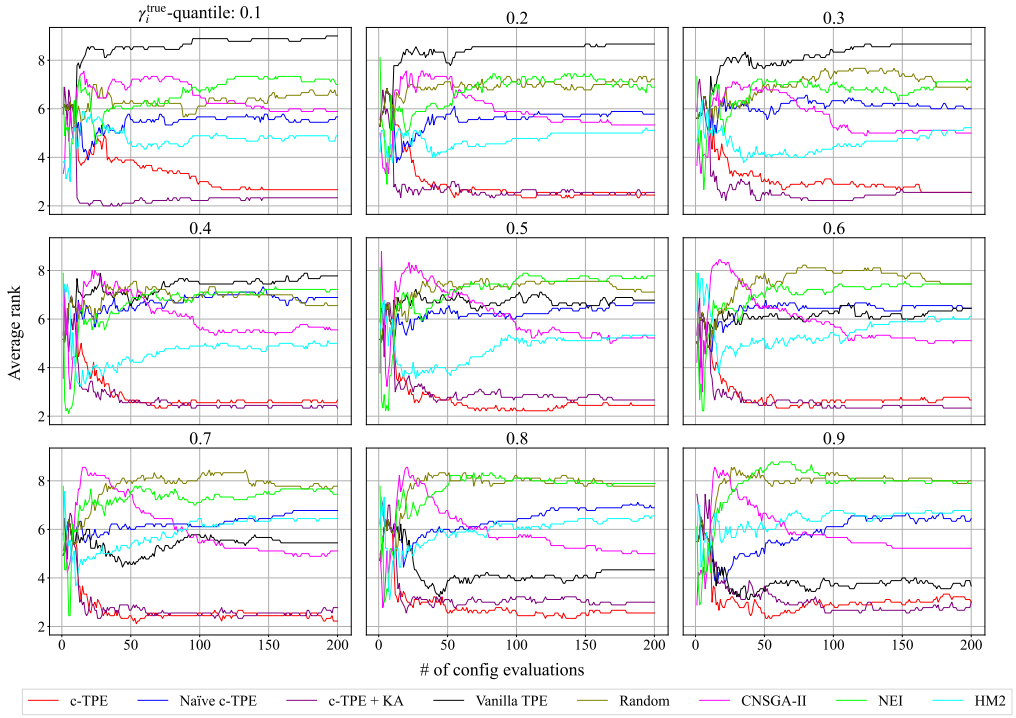

Figure 16: The average rank of each method over the number of evaluations. We evaluated each method on nine benchmarks with the network size constraint and each optimization was repeated over 50 random seeds. Each figure presents the results for γ_i^{true} of 0.1 to 0.9, respectively.

performance even over looser constraint settings while CNSGA-II and HM2 did not. This robustness is also from the property mentioned in Corollary 2.

H Additional results for the average rank over time

Figures 16, 17, and 18 show the average rank of each method over the number of evaluations. Each figure shows the performance of different constraint settings with 0.1 to 0.9 of γ_i^{true}.

As the constraint becomes tighter, c-TPE converges quicker in the early stage of the optimizations in all the settings due to KA. On the other hand, KA does not accelerate the optimizations as constraints become looser. This is because it is easy to obtain information about feasible domains even by random samplings. However, KA does not degrade the performance of c-TPE and thus it is recommended to add KA as much as possible.

Furthermore, it is worth noting that although the performance of HM2 and NEI outperformed the vanilla TPE in the tighter constraint settings, their performance is degraded as constraints become looser and they did not exhibit better performance than the vanilla TPE with $\gamma_i^{\text{true}} = 0.9$. On the other hand, c-TPE adapts the optimization based on the estimated $\gamma_{c^\star}$-quantile and thus it exhibited better performance than the vanilla TPE even in the settings of $\gamma_i^{\text{true}} = 0.9$.

I The performance of the vanilla TPE

As described in Appendix F, since our TPE implementation uses multivariate kernel density estimation, it is different from the Hyperopt implementation that is used in most prior works. For this reason, we demonstrate that our TPE implementation performs significantly better than Hyperopt and how our TPE compares with other BO methods in our experiment settings. In this experiment, we compare our TPE with Hyperopt and the following methods:

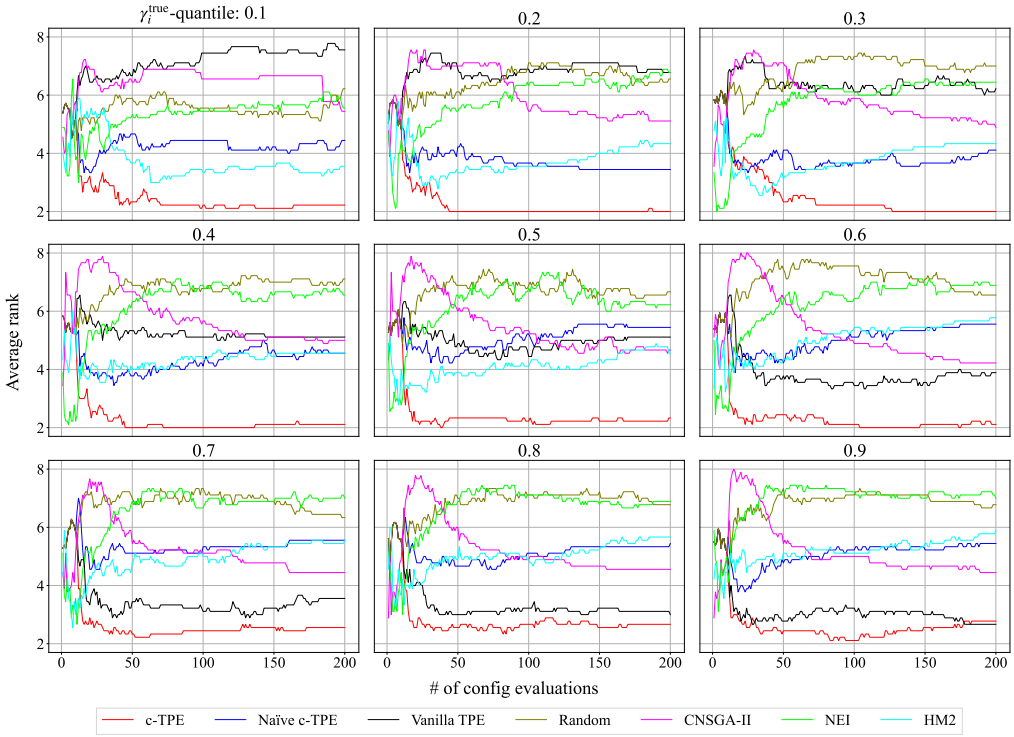

Figure 17: The average rank of each method over the number of evaluations. We evaluated each method on nine benchmarks with the runtime constraint and each optimization was repeated over 50 random seeds.

1. **TuRBO** (Eriksson *et al.* (2019)) [6], and
2. **CoCaBO** (Ru *et al.* (2020)) [7].

CoCaBO is a BO method that focuses on the handling of categorical parameters and TuRBO is one of the strongest BO methods developed recently. Both methods follow the default settings provided in the examples. Note that as both methods are either not extended to constraint optimization or not publicly available, we could not include those methods in Section 3.

Figure 19 shows the average rank over time for each method. As seen in the figure, our TPE is better than Hyperopt. Furthermore, while our TPE is significantly better than other methods in most settings, Hyperopt is better than only CoCaBO. On the other hand, TuRBO-1 performs better in the early stage of optimizations although our TPE outperforms TuRBO-1 with statistical significance, and this cold start in the vanilla TPE might be a trade-off. Notice that since most BO papers test performance on toy functions and we use the tabular benchmarks, the discussion here does not generalize and the results only validate why we should use our TPE in our paper.

J Limitations

In this paper, we focus on tabular benchmarks for search spaces with categorical parameters and with one or two constraints. We chose the tabular benchmarks to enable the stability analysis of the performance variations depending on constraint levels. Furthermore, such settings are common in HPO of deep learning. However, practitioners may use c-TPE for other settings, and thus we would like to discuss the following settings which we did not cover in the paper:

1. **Extremely small feasible domain size**
2. **Many constraints**

[6] Implementation: `https://github.com/uber-research/TuRBO`
[7] Implementation: `https://github.com/rubinxin/CoCaBO_code`

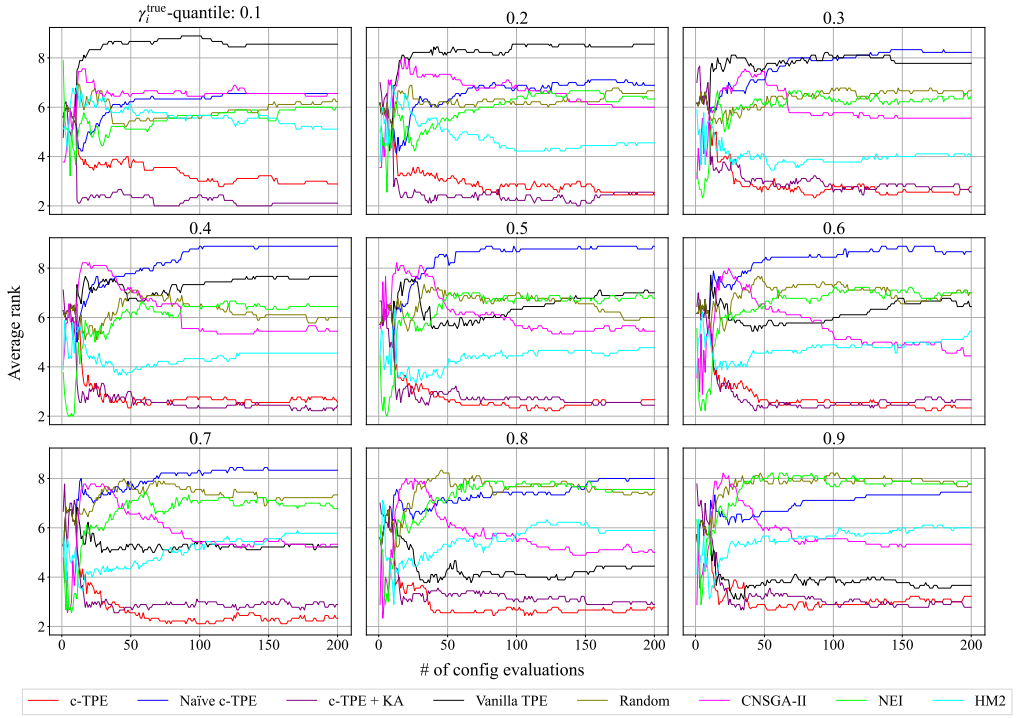

Figure 18: The average rank of each method over the number of evaluations. We evaluated each method on nine benchmarks with the runtime and the network size constraints and each optimization was repeated over 50 random seeds.

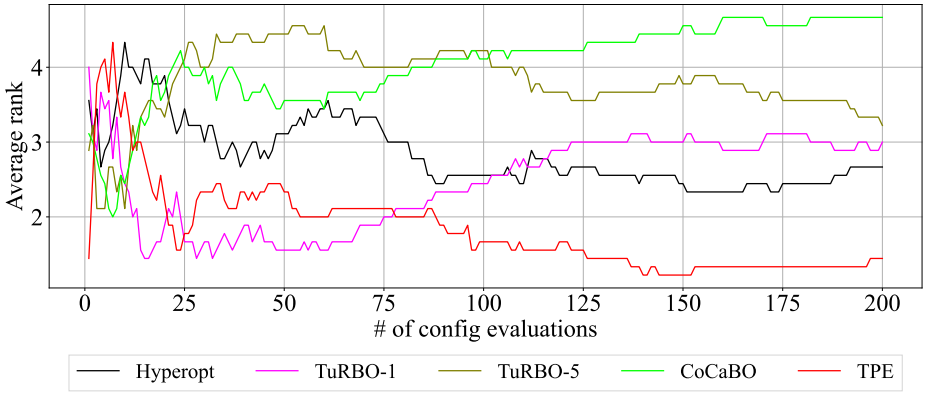

Figure 19: The performance comparison of our TPE implementation against prior works and Hyperopt implementation. The horizontal axis represents the number of configurations and the vertical axis represents the average rank of each method over 9 benchmarks that were used in Section 3.

3. **Parallel computation**
4. **Synthetic functions**

The first setting is an extremely small feasible domain size. For example, when we have $\Gamma = 10^{-4}$ for 200 evaluations and use random search, we will not get any feasible solutions with the probability of $(1 - 10^{-4})^{200} = 0.9802\cdots \simeq 98.0\%$. Such settings are generally hard for most optimizers to find even one feasible solution.

The second setting is tasks with many constraints. In our experiments, we have the constraints of runtime and network size. On the other hand, there might be more constraints in other purposes.

Table 3: In the table, we show the test results of The hypothesis "The other method is better than our TPE" for the "v.s. our TPE" column and the hypothesis "The other method is better than Hyperopt" for the "v.s. Hyperopt" column by the Wilcoxon signed-rank test. For example, the "TuRBO-1" row in the "v.s. our TPE" column says "N/N/W/W". It means while we cannot draw any conclusion about the performance difference with $50, 100$ evaluations, TuRBO-1 is significantly worse than our TPE with $150, 200$ evaluations in our settings. Note that we chose $p < 0.01$ as the threshold. Each method was run over 15 random seeds.

Methods # of configs	v.s. our TPE 50/100/150/200	v.s. Hyperopt 50/100/150/200
our TPE	–/–/–/–	N/B/B/B
Hyperopt	N/**W**/**W**/**W**	–/–/–/–
TuRBO-1	N/N/**W**/**W**	B/N/N/N
TuRBO-5	**W**/**W**/**W**/**W**	**W**/N/N/N
CoCaBO	**W**/**W**/**W**/**W**	N/**W**/**W**/**W**

Many constraints make the optimization harder because the feasible domain size becomes smaller as the number of constraints increases due to the curse of dimensionality. More formally, when we define the feasible domain for the i-th constraint as $\mathcal{X}'_i = \{\boldsymbol{x} \in \mathcal{X} | c_i(\boldsymbol{x}) \leq c_i^\star\}$, the feasible domain size shrinks exponentially unless some feasible domains are identical, i.e. $\mathcal{X}'_i = \mathcal{X}'_j$ for some pairs $(i, j) \in \{1, \ldots, C\} \times \{1, \ldots, C\}$ such that $i \neq j$. This setting is also generally hard due to the small feasible domain size.

The third setting is parallel computation. In HPO, since objective functions are usually expensive, it is often preferred to be able to optimize with less regret. For example, evolutionary algorithms evaluate a fixed number G of configurations in one generation and thus they optimize the objective function without any loss compared to the sequential setting up to G parallel processes. Although TPE (and c-TPE) are applicable to asynchronous settings, we cannot conclude c-TPE works nicely in parallel settings from our experiments.

The fourth setting is synthetic function. We did not handle synthetic function because we it is hard to prepare the exact γ_i^{true}. As mentioned earlier, one of the most important points of our method is the robustness with respect to various constraint levels. As synthetic functions are designed to be hard in certain constraint thresholds, it was hard to maintain the difficulties for different γ_i^{true} and to even analytically compute γ_i^{true}. Although we did not perform experiments on synthetic functions, c-TPE is likely to not perform well on multi-modal functions as c-TPE is a local search method due to the nature of PI.

We did not test c-TPE on those settings and thus practitioners are encouraged to compare c-TPE with other methods if their tasks of interest have the characteristics described above.

K Societal impacts

Since there are not many options for black-box constraint optimization and c-TPE is likely to be integrated into open source software, our method would enable, for example, efficient performance improvements along with a budget constraint and it potentially reduces the energy consumption during both HPO and actual operations. On the other hand, when practitioners run HPO, they are mostly required to design search spaces and to set constraints on their own. As discussed in the paper, the difficulties of constraint optimizations depend on the feasible domain size Γ, which heavily relies on the search space design and the choice of constraint thresholds. It might lead to yielding no feasible solutions in the case of too tight constraint values or no convergence in the case of too high search space dimensions. Then those runs will waste much energy; therefore, we still need to investigate ways to automate search space design to circumvent such difficulties.

References

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

D. Arnold and N. Hansen. A (1+1)-CMA-ES for constrained optimisation. In *Genetic and Evolutionary Computation Conference*, 2012.

N. Awad, N. Mallik, and F. Hutter. DEHB: Evolutionary hyberband for scalable, robust and efficient hyperparameter optimization. *arXiv:2105.09821*, 2021.

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.

E. Daxberger, A. Makarova, M. Turchetta, and A. Krause. Mixed-variable bayesian optimization. *arXiv:1907.01329*, 2019.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

A. Deshwal, S. Belakaria, and J. Doppa. Bayesian optimization over hybrid spaces. In *International Conference on Machine Learning*, 2021.

X. Dong and Y. Yang. NAS-bench-201: Extending the scope of reproducible neural architecture search. *arXiv:2001.00326*, 2020.

K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NeurIPS workshop on Bayesian Optimization in Theory and Practice*, 2013.

D. Eriksson and M. Poloczek. Scalable constrained bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, 2021.

D. Eriksson, M. Pearce, J. Gardner, RD. Turner, and M. Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems*, 2019.

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

J. Gardner, M. Kusner, ZE. Xu, K. Weinberger, and J. Cunningham. Bayesian optimization with inequality constraints. In *International Conference on Machine Learning*, 2014.

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

EC. Garrido-Merchán and D. Hernández-Lobato. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing*, 380, 2020.

M. Gelbart, J. Snoek, and R. Adams. Bayesian optimization with unknown constraints. *arXiv:1403.5607*, 2014.

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.

B. Letham, B. Karrer, G. Ottoni, and E. Bakshy. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, R. Sass, and F. Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *arXiv:2109.09831*, 2021.

JH. Lobato, M. Gelbart, M. Hoffman, R. Adams, and Z. Ghahramani. Predictive entropy search for bayesian optimization with unknown constraints. In *International Conference on Machine Learning*, 2015.

I. Loshchilov, M. Schoenauer, and M. Sebag. Bi-population CMA-ES agorithms with surrogate models and line searches. In *Genetic and Evolutionary Computation Conference*, 2013.

EM. Montes, J. Velázquez-Reyes, and CA. Coello. Modified differential evolution for constrained optimization. In *International Conference on Evolutionary Computation*, 2006.

L. Nardi, D. Koeplinger, and K. Olukotun. Practical design space exploration. In *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 347–358. IEEE, 2019.

V. Perrone, I. Shcherbatyi, R. Jenatton, C. Archambeau, and M. Seeger. Constrained bayesian optimization with max-value entropy search. *arXiv:1910.07003*, 2019.

B. Ru, A. Alvi, Vu V. Nguyen, M. Osborne, and S. Roberts. Bayesian optimisation over multiple continuous and categorical inputs. 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. *Proceedings of the IEEE*, 104(1):148–175, 2016.

R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track*, 2021.

C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, 2019.