c-TPE: Tree-Structured Parzen Estimator with Inequality Constraints for Expensive Hyperparameter Optimization

Shuhei Watanabe and **Frank Hutter**
Department of Computer Science, University of Freiburg, Germany
{watanabs, fh}@cs.uni-freiburg.de

Abstract

Hyperparameter optimization (HPO) is crucial for strong performance of deep learning algorithms and real-world applications often impose some constraints, such as memory usage, or latency on top of the performance requirement. In this work, we propose constrained TPE (c-TPE), an extension of the widely-used versatile Bayesian optimization method, tree-structured Parzen estimator (TPE), to handle these constraints. Our proposed extension goes beyond a simple combination of an existing acquisition function and the original TPE, and instead includes modifications that address issues that cause poor performance. We thoroughly analyze these modifications both empirically and theoretically, providing insights into how they effectively overcome these challenges. In the experiments, we demonstrate that c-TPE exhibits the best average rank performance among existing methods with statistical significance on 81 expensive HPO with inequality constraints. Due to the lack of baselines, we only discuss the applicability of our method to hard-constrained optimization in Appendix D. See https://arxiv.org/abs/2211.14411 for the latest version with Appendix.

1 Introduction

While deep learning (DL) has achieved various breakthrough successes, its performance highly depends on the proper settings of its hyperparameters [Chen *et al.*, 2018; Melis *et al.*, 2018]. Furthermore, practical applications often impose several constraints on memory usage or latency of inference, making it necessary to apply constrained hyperparameter optimization (HPO).

Recent developments in constrained HPO have led to the emergence of new acquisition functions (AFs) [Gardner *et al.*, 2014; Lobato *et al.*, 2015; Eriksson and Poloczek, 2021] in Bayesian optimization (BO) with Gaussian process (GP), which judge the promise of a configuration based on the surrogate model. While GP-based methods offer theoretical advantages, recent open source softwares (OSS) for HPO, such as Optuna [Akiba *et al.*, 2019], Hyperopt [Bergstra *et al.*, 2013], and Ray [Liaw *et al.*, 2018], instead employ the tree-structured Parzen estimator (TPE) [Bergstra *et al.*, 2011; Bergstra *et al.*, 2013; Watanabe, 2023], a variant of BO using the density ratio of kernel density estimators for good and bad observations, as the main algorithm, and Optuna played a pivotal role for HPO of DL models in winning Kaggle competitions [Alina *et al.*, 2019; Addison *et al.*, 2022]. Despite its versatility for expensive HPO problems, the existing AFs are not directly applicable to TPE and no study has been conducted on TPE's extension to constrained optimization.

In this paper, we propose c-TPE, a constrained optimization method that generalizes TPE. We first show that it is possible to integrate the original TPE into the existing AF proposed by Gelbart *et al.* [2014], which uses the product of AFs for the objective and each constraint, and thus TPE can be generalized with constrained settings. Then, a naïve extension, which calculates AF by the product of density ratios for the objective and each constraint with the same split algorithm, could be simply obtained; however, the naïve extension suffers from performance degradation under some circumstances. To circumvent these pitfalls, we propose (1) the split algorithm that includes a certain number of feasible solutions, and (2) AF by the product of relative density ratios, and analyze their effects empirically and theoretically.

In the experiments, we demonstrate (1) the strong performance of c-TPE with statistical significance on expensive HPO problems and (2) robustness to changes in the constraint level. Notice that we briefly discuss the applicability of our method to hard-constrained optimization in Appendix D, and we discuss the limitations of our work in Appendix E caused by our choices of search spaces that are limited to tabular benchmarks to enable the stability analysis of the performance variations depending on constraint levels.

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

1. prove that TPE can be extended to constrained settings using the AF proposed by Gelbart *et al.* [2014],
2. present two pitfalls in the naïve extension and describe how our modifications mitigate those issues,
3. provide the stability analysis of the performance variations depending on constraint levels, and
4. demonstrate that the proposed method outperforms existing methods with statistical significance on average on 9 tabular benchmarks with 27 different settings.

The implementation and the experiment scripts are available at https://github.com/nabenabe0928/constrained-tpe/.

2 Background

2.1 Bayesian Optimization (BO)

Suppose we would like to **minimize** a validation 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 HPO problem is defined as follows:

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

Note that $\boldsymbol{x} \in \mathcal{X}$ is a hyperparameter configuration, $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$ is the search space of the hyperparameter configurations, and $\mathcal{X}_d \subseteq \mathbb{R}$ (for $d = 1, \ldots, D$) is the domain of the d-th hyperparameter. 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 a set of observations $\mathcal{D} \coloneqq \{(\boldsymbol{x}_n, f_n)\}_{n=1}^N$. In each iteration of BO, we build a predictive model $p(f|\boldsymbol{x}, \mathcal{D})$ and optimize an AF to yield the next configuration. A common choice for AF is the following expected improvement (EI) [Jones *et al.*, 1998]:

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

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

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

2.2 Tree-Structured Parzen Estimator (TPE)

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

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

where $\mathcal{D}^{(l)}, \mathcal{D}^{(g)}$ are the observations with $f_n \leq f^\gamma$ and $f_n > f^\gamma$, respectively. Note that f^γ is the top-γ quantile objective value in $\mathcal{D}$ at each iteration and $p(\boldsymbol{x}|\mathcal{D}^{(l)}), p(\boldsymbol{x}|\mathcal{D}^{(g)})$ are built by the kernel density estimator [Bergstra *et al.*, 2011; Bergstra *et al.*, 2013; Falkner *et al.*, 2018]. Combining Eqs. (2), (4) and Bayes' theorem, the AF of TPE is computed as [Bergstra *et al.*, 2011]:

$$\mathrm{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}] \overset{\text{rank}}{\simeq} r(\boldsymbol{x}|\mathcal{D}) \coloneqq p(\boldsymbol{x}|\mathcal{D}^{(l)}) / p(\boldsymbol{x}|\mathcal{D}^{(g)}) \tag{5}$$

where $\phi(\boldsymbol{x}) \overset{\text{rank}}{\simeq} \psi(\boldsymbol{x})$ implies the order isomorphic and $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \phi(\boldsymbol{x}) \leq \phi(\boldsymbol{x}') \Leftrightarrow \psi(\boldsymbol{x}) \leq \psi(\boldsymbol{x}')$ holds and we use $f^\star = f^\gamma$ at each iteration. In each iteration, TPE samples configurations from $p(\boldsymbol{x}|\mathcal{D}^{(l)})$ and takes the configuration that achieves the maximum $r(\boldsymbol{x}|\mathcal{D})$.

2.3 Bayesian Optimization with Unknown Constraints

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

$$\begin{aligned} &\boldsymbol{x}_{\text{opt}} \in \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmin}} f(\boldsymbol{x}) \\ &\text{subject to } \forall i \in \{1, \ldots, C\}, c_i(\boldsymbol{x}) \leq c_i^\star \end{aligned} \tag{6}$$

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

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

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. However, the following simplified factorized form is the common choice:

$$\mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}] = \mathrm{EI}_{f^\star}[\boldsymbol{x}|\mathcal{D}] \prod_{i=1}^{C} \mathbb{P}(c_i \leq c_i^\star | \boldsymbol{x}, \mathcal{D}), \tag{8}$$

Since there are few methods available for hard-constrained optimization, we only discuss the applicability of our method to hard-constrained optimization in Appendix D.

3 Constrained TPE (c-TPE)

In this section, we first prove that TPE can be extended to constrained settings via the simple product of AFs. Then we describe an extension naïvely inspired by the original TPE and discuss two pitfalls hindering efficient search. Finally, we present modifications for those pitfalls and analyze the effects on toy problems.

Note that throughout this paper, we use the terms γ-*quantile value* f^γ as the top-γ quantile function value, $\gamma_{c^\star}$ as the quantile of $c^\star$, and Γ-*feasible domain* as the feasible domain in the search space $\mathcal{X}$ that covers $100 \times \Gamma\%$ of $\mathcal{X}$. For the formal definitions, see Appendix A.1. Furthermore, we consider two assumptions mentioned in Appendix A.2 and those assumptions allow the whole discussion to be extended to search spaces with categorical parameters.

3.1 Naïve Acquisition Function

Suppose we would like to solve constrained optimization problems formalized in Eq. (6) with ECI. To realize ECI in TPE, we first show the following proposition.

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

The proof is provided in Appendix A.3. Since PI and EI are equivalent under the TPE formulation, we obtain the following by combining Proposition 1 and Eq. (8):

$$\mathrm{ECI}_{f^\star}[\boldsymbol{x}|\boldsymbol{c}^\star, \mathcal{D}] \propto \underbrace{\mathbb{P}(f \leq f^\star | \boldsymbol{x}, \mathcal{D})}_{\overset{\text{rank}}{\simeq} r_0(\boldsymbol{x}|\mathcal{D})} \prod_{i=1}^{C} \underbrace{\mathbb{P}(c_i \leq c_i^\star | \boldsymbol{x}, \mathcal{D})}_{\overset{\text{rank}}{\simeq} r_i(\boldsymbol{x}|\mathcal{D})}. \tag{9}$$

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

3.2 Two Pitfalls in Naïve Extension

Naïve Extension and Modifications

From the discussion above, we could naïvely extend the original TPE to constrained settings using the split in Eq. (4) and the AF in Eq. (5). More specifically, the naïve extension computes the AF as follows:

Algorithm 1 c-TPE algorithm (With modifications)

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 $p(\cdot|\mathcal{D}_i^{(l)}), p(\cdot|\mathcal{D}_i^{(g)})$
11: $\{\boldsymbol{x}_j\}_{j=1}^{N_s} \sim p(\cdot|\mathcal{D}_i^{(l)}), \mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{x}_j\}_{j=1}^{N_s}$
12: ▷ See Appendix D for the hard-constrained version
13: Pick $\boldsymbol{x}_{\text{opt}} \in \text{argmax}_{\boldsymbol{x} \in \mathcal{S}} \prod_{i=0}^{C} r_i^{\text{rel}}(\boldsymbol{x}|\mathcal{D})$
14: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}_{\text{opt}}, f(\boldsymbol{x}_{\text{opt}}), c_1(\boldsymbol{x}_{\text{opt}}), \ldots, c_C(\boldsymbol{x}_{\text{opt}}))\}$

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

Note that as $c_i^\star$ is a user-defined threshold, $c_i^\star$ 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 top domains for the objective and feasible domains, or (2) vanished constraints.

For this reason, we change Operations 1 and 4 as follows:

- Pick the $\lceil \gamma|\mathcal{D}| \rceil$-th best **feasible** objective value $f^\star$ in $\mathcal{D}$ (Line 9), and
- Take the product of **relative** density ratios $\prod_{i=0}^{C} r_i^{\text{rel}}(\boldsymbol{x}|\mathcal{D}) \coloneqq \prod_{i=0}^{C} (\hat{\gamma}_i + (1 - \hat{\gamma}_i) r_i(\boldsymbol{x}|\mathcal{D})^{-1})^{-1}$ as the AF (Line 13).

Note that we color-coded the modifications in Algorithm 1 and we define $\hat{\gamma}_i \coloneqq |\mathcal{D}_i^{(l)}|/|\mathcal{D}|$. Intuitively, when all configurations satisfy the i-th constraint, i.e. $|\mathcal{D}_i^{(l)}| = |\mathcal{D}| \Rightarrow \hat{\gamma}_i = 1$, we trivially yield $r_i^{\text{rel}} = 1$; therefore, the i-th constraint will be ignored and it is equivalent to $\forall \boldsymbol{x} \in \mathcal{X}, \mathbb{P}[c_i \leq c_i^\star \boldsymbol{x}, \mathcal{D}] = 1$. Additionally, the following corollary guarantees the mathematical validity of our algorithm:

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

We provide the proof in Appendix A.4.

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** observations into $\mathcal{D}_0^{(l)}$ and the rest into $\mathcal{D}_0^{(g)}$, and this split algorithm matches the original algorithm when $\Gamma = 1$. For the split of constraints, we first check the upper bound of $\{c_{i,n}\}_{n=1}^{N}$ 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 $\min\{c_{i,n}\}_{n=1}^{N}$ so that the optimization of this constraint will be strengthened (see Theorem 1). 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. We describe more details in Appendix B and the applicability to hard-constrained optimization in Appendix D. We start the discussion of why these modifications mitigate the issues in the next section.

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 $\hat{\gamma}_i \coloneqq \hat{\gamma}_{c_i^\star} \simeq 1$. 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 $\boldsymbol{x}$. As discussed in Section 3.2, the relative density ratio $r_i^{\text{rel}}(\boldsymbol{x}|\mathcal{D})$ resolves this issue and it can be written more formally as follows:

Corollary 2 *Assuming the feasible domain quantile* $\Gamma = 1$, *then* $\prod_{i=0}^{C} r_i^{\text{rel}}(\boldsymbol{x}|\mathcal{D}) \overset{\text{rank}}{\simeq} r_0(\boldsymbol{x}|\mathcal{D})$ *holds.*

Recall that we previously defined $r_0(\boldsymbol{x}|\mathcal{D}) \coloneqq p(\boldsymbol{x}|\mathcal{D}_0^{(l)})/p(\boldsymbol{x}|\mathcal{D}_0^{(g)})$ for $\mathcal{D}_0^{(l)}, \mathcal{D}_0^{(g)}$ obtained by splitting $\mathcal{D}$ at $f^\star$. The proof is provided in Appendix A.6. Corollary 2 indicates that the AF of c-TPE is equivalent to that of the original TPE when $\Gamma = 1$ and it means that our formulation achieves $\mathbb{P}(c_i \leq c_i^\star|\boldsymbol{x}, \mathcal{D}) = 1$ if $\hat{\gamma}_i = 1$. Corollary 2 is a special case of the following theorem:

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*

$$\frac{\partial \prod_{k=0}^{C} r_k^{\text{rel}}(\boldsymbol{x}|\mathcal{D})}{\partial r_i} \geq \frac{\partial \prod_{k=0}^{C} r_k^{\text{rel}}(\boldsymbol{x}|\mathcal{D})}{\partial r_j} \geq 0 \tag{10}$$

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 A.5. Roughly speaking, Theorem 1 implies that our modified AF puts more priority on the variations of the density ratios with lower quantiles, i.e. r_i in the statement above, when $r_i = r_j$.

We empirically and intuitively present the effect of Theorem 1 in Figure 1. 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 G. As mentioned earlier, since the naïve extension (**Left column**) does not decay the contribution from the objective or the constraint with a large $\hat{\gamma}_i$, it has two peaks. For our algorithm, however, we only have one peak between the top-10% domain

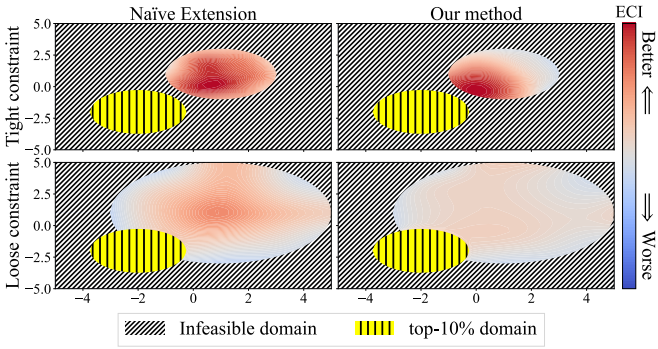

Figure 1: Heat maps of the AF in the naïve extension (**Left column**) and our c-TPE (**Right column**) with a tight (**Top row**, $c_1 = 4$) or loose (**Bottom row**, $c_1 = 16$) constraint. For fair comparisons, we use a fixed set of 200 randomly sampled configurations to compute the AF for all settings. In principle, red regions have higher AF values and the next configuration is likely to be picked from here.

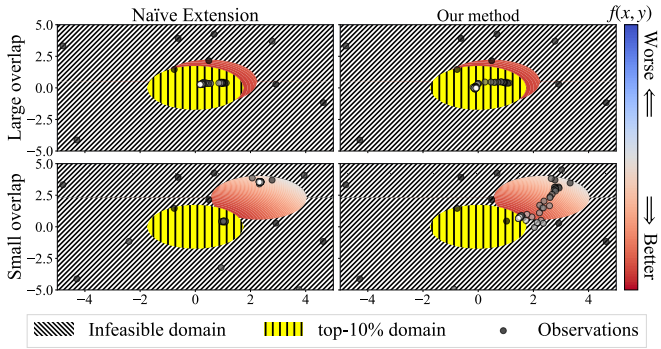

Figure 2: Scatter plots of observations obtained by the naïve extension (**Left column**) and our c-TPE (**Right column**) on a large (**Top row**, $z = 0.5$) or small (**Bottom row**, $z = 2.3$) overlap between the top-10% domain and feasible domain. Each figure shows the 2D search space for each task and the observations obtained during optimization are plotted. Earlier observations are colored black and later observations are colored white. Each figure has 50 observations.

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 tight constraint case (**Top right**), since the feasible domain quantile $\hat{\gamma}_1 \simeq 0.12$ is relatively small compared to the top-solution 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 the top-10% domain in the next section. For the loose constraint case (**Bottom right**), $\hat{\gamma}_1 \simeq 0.50$ 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 $r_1(\boldsymbol{x}|\mathcal{D})$. As mentioned in Corollary 2, $r_i^{\mathrm{rel}}(\boldsymbol{x}|\mathcal{D}) = 1$ holds for $i \in \{1, \dots, C\}$ when $\Gamma = 1$, and thus the AF coincides with that for the single objective optimization. Note that since we yield $\gamma_0 = 1.0$ in the case of all observations being infeasible, the objective function will be ignored and only constraints will be optimized.

Issue II: Small Overlaps in Top 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. We explain its effect using Figure 1. For the tight constraint case (**Top row**), an overlap between the feasible domain and the top-10% domain does not exist and it causes the two peaks in the AF for the original split algorithm (**Top left**); however, it is necessary for constrained optimization to sample intensively within feasible domains. In turn, we modify the split algorithm to include a certain number of feasible solutions. This modification leads to the large white circle that embraces the top-10% domain (**Top right**). As a result, our algorithm yields a peak at the overlap between the large white circle and the feasible domain.

In Figure 2, we visualize 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 settings used in Appendix G and both algorithms share the initial configurations. For the large overlap case (**Top row**), both algorithms search similarly. In contrast to this case, the small overlap case (**Bottom row**) obtained different sampling behaviors. While our algorithm (**Bottom right**) samples intensively at the boundary between the feasible domain and the top-10% domain, the naïve extension (**Bottom left**) does not. Furthermore, we can see a trajectory from the top right of the feasible domain to the boundary for our algorithm and it exists only in our algorithm although both methods have some observations, which are colored by strong gray, meaning that they were obtained at the early stage of the optimization, in the top right of the feasible domain. Based on Figure 1 (**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 toward the top-10% domain in our algorithm.

4 Experiments

4.1 Setup

The evaluations were performed on the following 10 tabular 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.

The reason behind this choice is that tabular benchmarks enable us to control the quantiles of each constraint γ_i^{true}, which significantly change the feasible domain size and the quality of solutions. For example, suppose a tabular dataset has N_{all} configurations $\{(\boldsymbol{x}_n, f_n, \boldsymbol{c}_n)\}_{n=1}^{N_{\mathrm{all}}}$ and the dataset is sorted so that it satisfies $c_{i,1} \leq c_{i,2} \leq \cdots \leq c_{i,N_{\mathrm{all}}}$ where $c_{i,n}$ is the i-th constraint value in the n-th configuration, then we fix the

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

Table 1: The table shows (Wins/Loses/Ties) of c-TPE against each method for optimizations with different constraint levels (9 benchmarks $\times$ 3 constraint choices $= 27$ settings). The number of wins was counted by comparing medians of performance over 50 random seeds in each setting between two methods. Non-bold numbers indicate $p < 0.01$ of the hypothesis "The other method is better than c-TPE" by the Wilcoxon signed-rank test.

threshold for the i-th constraint $c_i^\star$ as $c_{i,\lfloor N_{\text{all}}/10 \rfloor}$ in the setting of $\gamma_i^{\text{true}} = 1/10$. We evaluated each benchmark with 9 different quantiles γ_i^{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].

As the baseline methods, we chose:

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

We describe the details of each method and their control parameters in Appendix G. Note that all experiments were performed 50 times with different 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 H and the source code is available at https://github.com/nabenabe0928/constrained-tpe along with complete scripts to reproduce the experiments, tables, and figures. A query of c-TPE with $\{50, 100, 150, 200\}$ observations took $\{0.22, 0.24, 0.26, 0.28\}$ seconds for a 30D problem with 8 cores of core i7-10700.

4.2 Robustness to Feasible Domain Size

This experiment shows how c-TPE performance improves given various levels of constraints. We optimized each benchmark with the aforementioned three types of constraints and chose $\gamma_i^{\text{true}} \in \{0.1, 0.5, 0.9\}$ for each constraint. All results on other benchmarks are available in Appendix H. Table 1 presents the numbers of wins/loses/ties and statistical significance by the Wilcoxon signed-rank test and Figure 3 shows the performance curves for each benchmark.

As a whole, while the performance of c-TPE is stable across all constraint levels, that of NEI, HM2, and CNSGA-II variates 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.

For ImageNet of NAS-Bench-201 (**Bottom row**), the naïve c-TPE is completely defeated by the other methods while c-TPE achieves the best or indistinguishable performance from the best performance. This gap between c-TPE and the naïve c-TPE is caused by the small overlaps discussed in Section 3.2. For example, only 59% of the top-10% configurations belong to the feasible domain in NAS-Bench-201 of $\gamma_i^{\text{true}} = 0.9$ although we can usually expect that 90% of them belong to the feasible domain, and 84% and 77% of those in HPOlib and NAS-Bench-101 actually belong to the feasible domain for $\gamma_i^{\text{true}} = 0.9$, respectively. The small overlap leads to the performance gap between c-TPE and the vanilla TPE as well. As TPE is not a uniform sampler and tries to sample from top domains, $\hat{\gamma}_i$ will not necessarily approach γ_i^{true}. In our case, it is natural to consider $\hat{\gamma}_i$ to be closer to 59% rather than 90% as only 59% of top-10% configurations are feasible. As mentioned also in Theorem 1, c-TPE is advantageous to such settings compared to the vanilla TPE and the naïve c-TPE.

For CIFAR10A of NAS-Bench-101 (**Middle row**), the results show different patterns from the other settings due to the high-dimensional ($D = 26$) nature. For $\gamma_i^{\text{true}} = 0.1, 0.5$ (**Left, center**), most methods exhibit indistinguishable performance from random search especially in the beginning because little information on feasible domains is available in the early stage of optimizations due to the high dimensionality although c-TPE outperforms in the end. In $\gamma_i^{\text{true}} = 0.9$ (**Right**), 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 dimensions and overlaps in top domains and feasible domains are large, it might be better to greedily optimize only the objective rather than regularizing the optimization of the objective as in our

[1] Implementation: https://github.com/optuna/optuna

[2] Implementation: https://github.com/facebook/Ax

[3] Implementation: https://github.com/luinardi/hypermapper

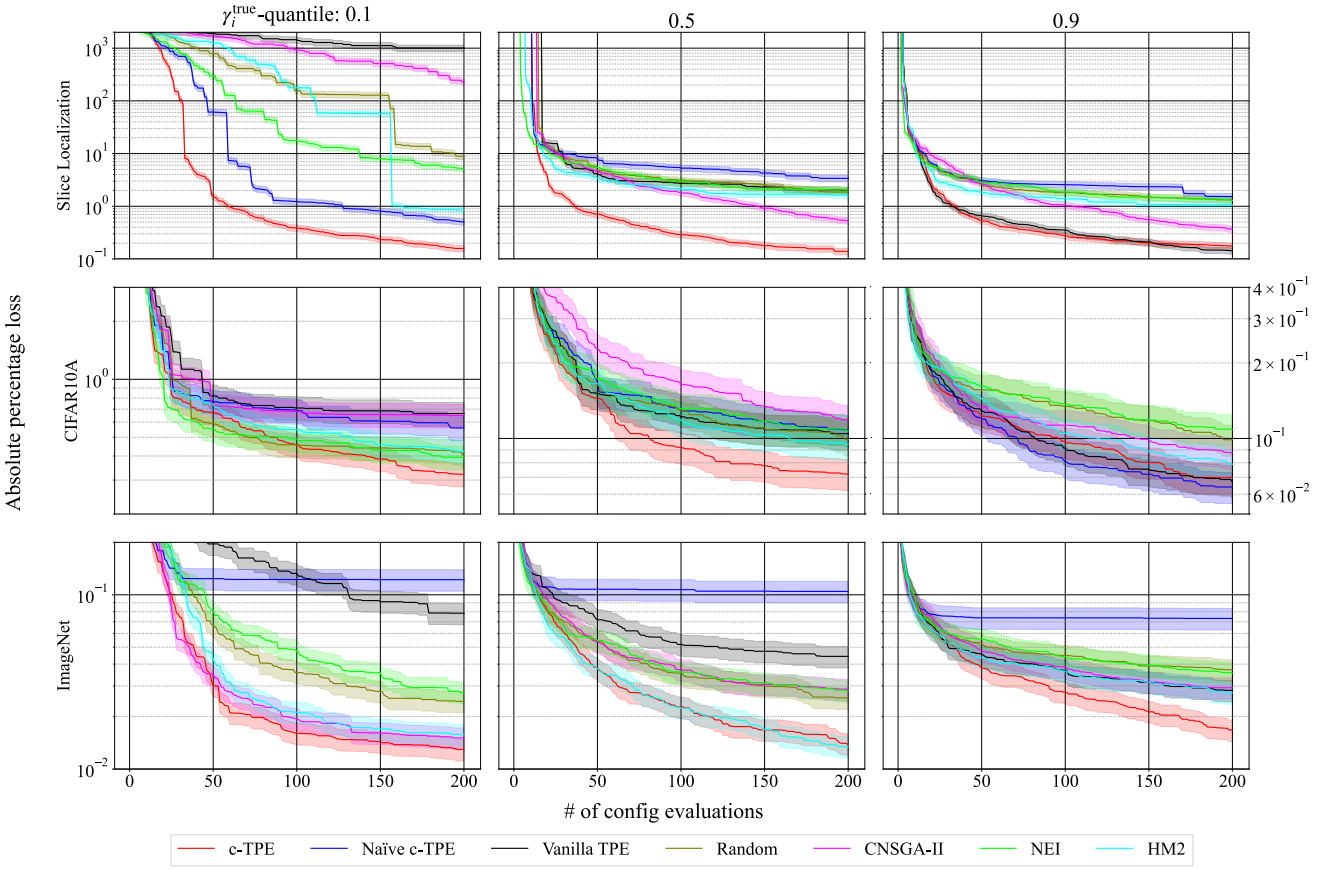

Figure 3: The performance curves on Slice Localization in HPOlib (**Top row**), CIFAR10A in NAS-Bench-101 (**Middle row**), and ImageNet16-120 in NAS-Bench-201 (**Bottom row**) with constraints of runtime and network size. We picked $\gamma_i^{\text{true}} = 0.1$ (**Left column**), 0.5 (**Center column**), 0.9 (**Right column**). 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. Further results are available in Appendix H.

modification.

For Slice Localization of HPOlib (**Top row**), c-TPE outperforms the other methods. Furthermore, its performance almost coincides with that of the vanilla TPE in $\gamma_i^{\text{true}} = 0.9$ and it implies that our method gradually decays the priority of each constraint as γ_i^{true} becomes larger. In fact, the naïve c-TPE does not exhibit stability when the constraint level changes as it does not consider the priority of each constraint and the objective. This result empirically validates Theorem 1.

4.3 Average Rank over Number of Evaluations

This experiment demonstrates how c-TPE performance improves compared to the other methods over the number of evaluations. Table 2 presents the numbers of wins/loses/ties and statistical significance by the Wilcoxon signed-rank test and Figure 4 shows the average rank over 81 settings (9 benchmarks $\times$ 9 quantiles).

According to Figure 4, c-TPE quickly takes the top and keeps the rank until the end. From the figures, we can see that CNSGA-II improves in rank as the number of evaluations grows. In fact, since such slow-starting is often the case for evolutionary algorithms such as CMA-ES [Loshchilov and Hutter, 2016], the quick convergence achieved by c-TPE is appealing. For the multiple-constraint setting (**Right**), while the naïve c-TPE is worse than random search due to the small overlap, c-TPE overcomes this problem as discussed in Section 3.2. Table 2 confirmed the anytime performance of c-TPE by the statistical test over all the settings. All results on individual settings and quantile-wise average rank are available in Appendices H and I.

5 Related Work & Discussion

ECI was introduced by Gardner *et al.* [2014] and Gelbart *et al.* [2014]. Furthermore, there are various extensions of these prior works. For example, NEI is more robust to the noise caused in experiments [Letham *et al.*, 2019] and SCBO is scalable to high dimensions [Eriksson and Poloczek, 2021]. Another technique for constrained BO is entropy search, such as predictive entropy search [Lobato *et al.*, 2015; Garrido-Merchán *et al.*, 2023] 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

Constraints	Runtime & Network size				Network size				Runtime			
Methods / # of configs	50	100	150	200	50	100	150	200	50	100	150	200
Naïve c-TPE	77/3/1	79/0/2	78/0/3	79/0/2	75/4/2	77/1/3	76/1/4	80/0/1	66/8/7	71/5/5	70/3/8	69/2/10
Vanilla TPE	73/7/1	75/5/1	72/3/6	73/4/4	69/10/2	74/6/1	74/3/4	72/6/3	62/12/7	67/10/4	62/6/13	60/9/12
Random	80/0/1	81/0/0	81/0/0	80/0/1	80/0/1	79/2/0	80/1/0	81/0/0	80/0/1	78/0/3	79/0/2	81/0/0
CNSGA-II	80/0/1	79/0/2	76/1/4	75/2/4	77/3/1	78/1/2	75/2/4	75/1/5	74/1/6	76/0/5	74/0/7	74/0/7
NEI	79/1/1	81/0/0	81/0/0	81/0/0	79/1/1	80/1/0	80/1/0	81/0/0	77/0/4	78/0/3	79/0/2	81/0/0
HM2	74/5/2	77/3/1	77/1/3	76/2/3	76/4/1	78/2/1	76/2/3	78/0/3	71/4/6	73/2/6	67/3/11	70/2/9

Table 2: The table shows (Wins/Loses/Ties) of c-TPE against each method for optimizations with different constraints (9 benchmarks $\times$ 9 quantiles $=$ 81 settings). The number of wins was counted by comparing medians of performance over 50 random seeds in each setting between two methods. In this table, All results indicate $p < 0.01$ of the hypothesis "The other method is better than c-TPE" by the Wilcoxon signed-rank test.

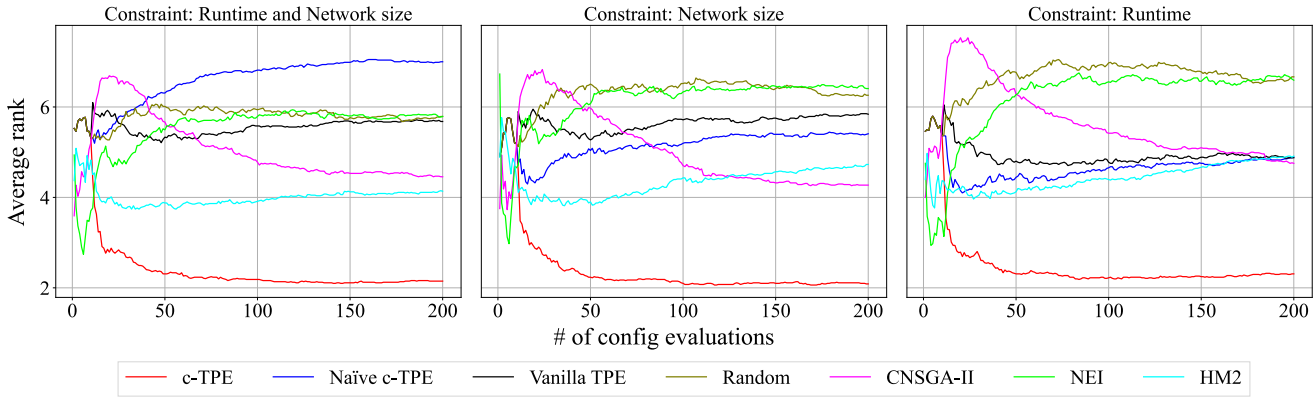

Figure 4: The average rank of each method over the number of evaluations. The horizontal axis shows the number of evaluated configurations in optimizations and the vertical axis shows the average rank over 81 settings. The title of each figure shows the constraint that the optimizations handled.

constrained minimizer. While entropy search could outperform c-TPE on multimodal functions by leveraging the global search nature, slow convergence due to the global search nature and the expensive query cost hinder practical usages. Note that as the implementations of these methods are not provided in the aforementioned papers except NEI, we used only NEI in the experiments. The major advantages of TPE over standard GP-based BOs, used by all of these papers, are more natural handling of categorical and conditional parameters (see Appendix F) and easier integration of cheap-to-evaluate partial observations due to the linear time complexity with respect to $|\mathcal{D}|$. The concept of the integration of partial observations and its results, which showed a further acceleration of c-TPE, are available in Appendix C.

Also in the evolutionary algorithm (EA) community, constrained 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, so we did not include it in our experiments. Furthermore, since EAs have many control parameters, such as mutation rate and population size, meta-tuning may be necessary. 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 C.

6 Conclusion

In this paper, we introduced c-TPE, a new constrained BO method. Although the AF of constrained BO and TPE could naturally come together using Corollary 1, such a naïve extension fails in some circumstances as discussed in Section 3. Based on the discussion, we modified c-TPE so that the formulation strictly generalizes TPE and falls back to it in settings of loose constraints. Furthermore, we empirically demonstrated that our modifications help to guide c-TPE to overlaps in the top and feasible domains. In our series of experiments on 9 tabular benchmarks and with 27 constraint settings, 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 E. Since TPE is very versatile and prominently used in several active OSS tools, such as Optuna and Ray, c-TPE will yield direct positive impact to practitioners in the future.

Acknowledgments

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

Funded by the European Union

References

[Addison *et al.*, 2022] H. Addison, KS. inversion, H. Ryan, and C. Ted. Happywhale - whale and dolphin identification. *Kaggle*, 2022.

[Akiba *et al.*, 2019] 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.

[Alina *et al.*, 2019] JE. Alina, C. Phil, B. Rodrigo, and G. Victor. Open images 2019 - object detection. *Kaggle*, 2019.

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

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

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

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

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

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

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

[Deb *et al.*, 2002] 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.

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

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

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

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

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

[Garrido-Merchán *et al.*, 2023] EC. Garrido-Merchán, D. Fernández-Sánchez, and D. Hernández-Lobato. Parallel predictive entropy search for multi-objective Bayesian optimization with constraints applied to the tuning of machine learning algorithms. *Expert Systems with Applications*, 215, 2023.

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

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

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

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

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

[Liaw *et al.*, 2018] 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.

[Lobato *et al.*, 2015] 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.

[Loshchilov and Hutter, 2016] I. Loshchilov and F. Hutter. CMA-ES for hyperparameter optimization of deep neural networks. *arXiv:1604.07269*, 2016.

[Loshchilov *et al.*, 2013] 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.

[Melis *et al.*, 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.

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

[Nardi *et al.*, 2019] 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.

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

[Shahriari *et al.*, 2016] B. Shahriari, K. Swersky, Z. Wang, R. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

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

[Ying *et al.*, 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.