Generalization Bounds for Model-based Algorithm Configuration

Zhiyang Chen[1] **Hailong Yao**[2,3]* **Xia Yin**[1]
[1]Tsinghua University [2]University of Science and Technology Beijing
[3]Key Laboratory of Advanced Materials and Devices
for Post-Moore Chips, Ministry of Education of China

Abstract

Algorithm configuration, which involves selecting algorithm parameters based on sampled problem instances, is a crucial step in applying modern algorithms such as SAT solvers. Although prior work has attempted to understand the theoretical foundations of algorithm configuration, we still lack a comprehensive understanding of why practical algorithm configurators exhibit strong generalization performances in real-world scenarios. In this paper, through the lens of machine learning theory, we provide an algorithm-dependent generalization bound for the widely used model-based algorithm configurators under mild assumptions. Our approach is based on the algorithmic stability framework for generalization bounds. To the best of our knowledge, this is the first generalization bound that applies to a model closely approximating practical model-based algorithm configurators.

1 Introduction

Many algorithms used in practice contain a large number of parameters that need to be tuned by users. For example, modern SAT and mixed-integer programming solvers generally have dozens of parameters that define the search strategy. A set of carefully tuned parameters may provide over $1000\times$ performance improvement compared with default settings [1]. Therefore, selecting algorithm parameters, a.k.a., *algorithm configuration* (AC), is a crucial step in applying parameterized algorithms.

Tuning parameters manually is usually time-consuming and highly dependent on the user's experience. Much prior work has been devoted to designing automatic AC methods. Due to the high intrinsic complexities of modern algorithms, these methods generally treat AC as a black-box optimization problem. The AC problem can be formally stated as follows: Given a parameterized[1] algorithm $\mathcal{A}$ with parameter space Θ, a set of problem instances $I_1, \ldots, I_m$ independently sampled from some probabilistic distribution $\mathcal{D}$, and a metric function $u(\theta, I)$ that measures the performance of parameter $\theta \in \Theta$ on instance I (e.g., running time), find a parameter configuration $\tilde{\theta} \in \Theta$ that optimizes $\frac{1}{m}\sum_{i=1}^{m} u(\tilde{\theta}, I_i)$. Here, $\mathcal{D}$ can be regarded as an application-specific model of problem instances, e.g., a uniform distribution over mixed-integer programs formulating facility location instances. We hope the found configuration $\tilde{\theta}$ performs well on distribution $\mathcal{D}$, i.e., with optimized $\mathbb{E}_{I\sim\mathcal{D}}[u(\tilde{\theta}, I)]$.

Various AC methods have been proposed to tackle this problem [2, 3, 4]. One of the most popular methods is the so-called sequential model-based optimization (SMBO), which exploits a surrogate model to fit the landscape of the performance metric function, and iteratively samples configurations with promising predicted performances. See Schede et al. [5] for a comprehensive survey.

*Corresponding author. E-mail: `hailongyao@ustb.edu.cn`

[1]By *parameterized algorithms*, we mean the algorithms contain tunable parameters (a.k.a. hyper-parameters in machine learning), instead of algorithms in the context of parameterized complexity theory.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Although state-of-the-art model-based methods yield good performances on practical algorithmic problems, an interesting theoretical question remains unsolved:

The parameters found by algorithm configurators are only evaluated on sampled problem instances from distribution $\mathcal{D}$. Why do they perform well on other unseen instances from $\mathcal{D}$?

In this work, we answer this question by presenting generalization bounds for model-based algorithm configurators through the lens of statistical learning theory. A generalization bound provides a guarantee that upper bounds the gap between training and testing performances of the learned parameter, i.e., $|\mathbb{E}_{I\sim\mathcal{D}}[u(\tilde{\theta}, I)] - \frac{1}{m}\sum_{i=1}^{m} u(\tilde{\theta}, I_i)|$. From the practical view, a tight generalization bound for AC is important since it implies the number of data samples such that the learned configuration does not overfit training data and exhibits strong generalization abilities.

1.1 Limitations of previous work

Several studies have been devoted to generalization guarantees of AC. Previous work can be generally categorized into two classes: Problem-dependent and problem-independent guarantees.

Problem-dependent guarantees, a.k.a., *data-driven algorithm design* [6], provide generalization bounds for specific families of parameterized algorithms. This line of research exploits the structures of the problem instances and the parameterized algorithms, instead of studying the algorithm configurators. Gupta and Roughgarden [7] initially propose the framework of data-driven algorithm design, and present generalization bounds for tuning greedy heuristics, tuning local search, and learning the step size for gradient descent, etc. Subsequent work provides generalization bounds for learning the branch strategy for branch-and-bound [8, 9], learning cutting plane parameters for branch-and-cut [10], tuning cooling parameters for simulated annealing [11], tuning dynamic programming for string alignment [12], tuning multi-objective optimization [13], and VLSI design [14, 15], etc. Although these studies are highly non-trivial, they suffer from two limitations:

- These generalization guarantees are problem-dependent. We need to analyze parameterized algorithms in practice one by one. It is hard to establish a general result for AC following this line of research.
- These results are generally limited to a **single** family of parameters, which are incompatible with practice. For example, the work of Balcan et al. [8] and Balcan et al. [10] studies generalization bounds of branch policies and cut policies for integer program (IP) solvers, respectively. However, there is no known method to combine these two results to obtain a guarantee for the joint space of branch and cut parameters. In contrast, a modern solver has dozens of parameters (e.g., 135 parameters for CPLEX). It seems intractable to prove a problem-dependent generalization bound for the entire parameter space of IP solvers.

Problem-independent guarantees provide generalization bounds that are independent of the parameterized algorithm. Thus, this kind of guarantee treats the algorithm to be configured as a black-box and provides results that are more general. To the best of our knowledge, the only problem-independent generalization bound is due to Liu et al. [16]. However, their result only applies to discrete parameters (which is technically easy to obtain by union bounds), while in practice, an algorithm may contain continuous parameters. Although their result can be extended to the continuous case by assuming that the performance metric is Lipschitz on the parameters, the Lipschitzness assumption is too strong to be compatible with practice: As discussed in previous work [8, 9], the landscape of the performance metric in AC is usually piecewise structured, with many non-smooth discontinuities.

To summarize, it remains an open question whether we can provide generalization guarantees for general AC problems (with multiple and continuous parameters). Note that all previously discussed work derives generalization bounds using *uniform convergence*. Uniform convergence bounds imply that the generalization upper bound holds for **any** learned parameter in the parameter space. Thus, they are independent from the algorithm configurator. Due to the black-box nature of AC, it is impossible to derive a general problem-independent uniform convergence bound. To overcome this barrier, we instead consider *algorithm-dependent*[2] generalization bounds, which utilize the properties of the algorithm configurator.

[2]The *algorithm-dependent* term is borrowed from statistical learning theory. In generalization bounds of neural networks, an *algorithm-dependent* bound depends on the learning algorithm (e.g., gradient descent). In

Table 1: A comparison with previous generalization bounds for AC.

Reference	AC Problem	#Parameters	Parameter Space	Configurator	Proof Technique
Gupta and Roughgarden [7]	Greedy & Local search	1	Continuous	Any	Uniform convergence
Gupta and Roughgarden [7]	Gradient descent	1	Continuous	Any	Uniform convergence
Balcan et al. [8, 9]	Branch policy for MIP	Several	Continuous	Any	Uniform convergence
Balcan et al. [10]	Cutting plane for MIP	Several	Continuous	Any	Uniform convergence
Blum et al. [11]	Simulated annealing	Multiple	Continuous	Any	Uniform convergence
Balcan et al. [12]	String alignment	Several	Continuous	Any	Uniform convergence
Balcan et al. [12]	Clustering heuristics	1	Continuous	Any	Uniform convergence
Balcan et al. [17, 18]	Regularized regression	2	Continuous	Any	Uniform convergence
Balcan and Sharma [19]	Decision tree heuristics	≤ 3	Continuous	Any	Uniform convergence
Chen et al. [13]	Multi-obj. optimization	1	Continuous	Any	Uniform convergence
Liu et al. [16]	Problem-independent	Multiple	Discrete only	Any	Uniform convergence
Ours (Theorem 5)	**Problem-independent**	**Multiple**	**Continuous**	Model-based	Algorithmic stability

1.2 Our contributions

Main results. In this work, we present the first generalization bound for the general AC problem using model-based algorithm configurators. Model-based AC is a popular framework in practice, which applies a Bayesian optimization framework to configure the parameterized algorithm. In Bayesian optimization, random forests are often used as the surrogate model[3]. With a few assumptions on the algorithm configurator (these assumptions are generally realistic, as we will show in experiments), we prove a generalization bound for the learned parameter in Theorem 5. To the best of our knowledge, this is the first generalization bound for general AC with continuous and multiple parameters, and the first generalization bound that applies to a model closely approximating practical model-based algorithm configurators. See Table 1 for a comparison with previous work. An informal version of our bound is as follows.

Theorem 1 (Informally, Theorem 5)**.** *The generalization error of model-based algorithm configurators with random forest surrogate models is upper bounded by*

$$\tilde{O}\left(\sqrt{T\left(\frac{1}{m}+\sqrt{\frac{n+d}{Q}}\right)}\right)$$

in expectation, where T is the iteration number of the configurator, m is the number of sampled instances, Q is the number of decision trees in the random forest surrogate model, n is the number of parameters, and d is the number of instance features.

Note that our bound converges with a rate of $m^{-1/2}$ with respect to the number of training samples, but with an additive term of $((n+d)/Q)^{1/4}$. This term is related to the stability of the random forest model. Intuitively, the model becomes more stable as we increase the number of decision trees in the forest. In fact, this term comes from the Rademacher complexity of the *dual* decision tree class. We will numerically show that this term is very small for a mild number of trees Q in experiments. To compute this data-dependent empirical Rademacher complexity term, we propose a divide-and-conquer algorithm in Appendix C.

Our techniques. Different from previous uniform convergence results for AC, our generalization bound is algorithm-dependent, which exploits the properties of the algorithm configurator. Our technical approach relies on the algorithmic stability framework [20]. We show that the model-based algorithm configurator with random forest surrogate models is stable, i.e., a small change in the training data to the configurator does not change the learned configuration much. Note that the algorithm configurator is a randomized algorithm. Thus, we bound the difference between the distributions of the output configuration on two training datasets with only one different instance. To achieve this, two building blocks are required to bound the stability of the random forest model

our work, by *algorithm-dependent*, we mean the bound depends on the algorithm configurator, instead of the parameterized algorithm to be configured.

[3]Although Gaussian processes are popular in Bayesian optimization, model-based algorithm configurators usually use random forests since (1) random forests are faster compared with Gaussian processes; (2) random forests can integrate the features of problem instances to achieve accurate predictions. See, e.g., Hutter et al. [2].

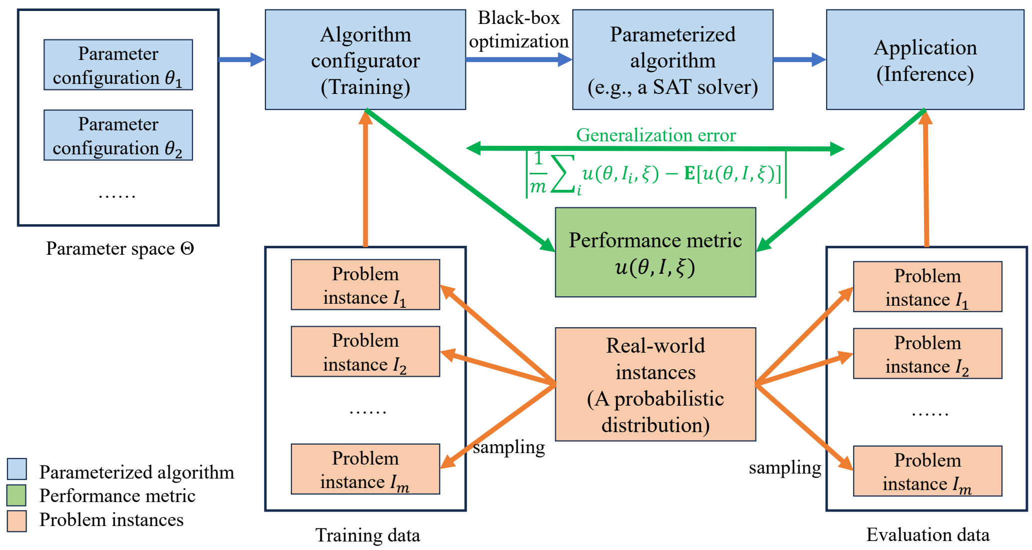

Figure 1: The framework of algorithm configuration and generalization guarantees.

and the local search sampling method in the configurator, respectively, which may be of independent interest.

The stability of random forests has been studied in previous work, such as Soloff et al. [21] and Wang et al. [22]. However, in our analysis, we need (1) a uniform upper bound on the stability, while previous work focuses on the stability of a single point; (2) a stability bound for both the empirical mean and the variance estimations, while previous work only considers the mean. Therefore, we derive upper bounds on *Rademacher complexity* of the mean and the variance of the *dual* decision tree class to achieve these targets. For the local search sampling method, we assume the configurator applies a simulated-annealing-style rule, and study its convergence using the theory of Monte-Carlo Markov Chains (MCMC). Different from classic mixing-time bounds for MCMC that consider additive errors [23], we consider the convergence using the multiplicative error.

1.3 Organization

The rest of this paper is organized as follows: Section 2 introduces the problem formulation and technical preliminaries. Section 3 introduces the framework of model-based algorithm configurators and presents our generalization bound. Section 4 concludes the paper and discusses future work. Proofs are omitted to the appendix. Appendix A presents some technical lemmas. Appendix B gives the proof of our main theorem. Appendix C provides the data-dependent bound and numerical experiments.

2 Preliminaries

2.1 Problem formulation

Notations. Throughout this paper, we use $[n]$ to denote the set $\{1, \ldots, n\}$. For asymptotic symbols, we use $\tilde{O}(f(n)) = f(n) \cdot \text{poly}\log n$ to hide logarithmic factors and use $f(n) \lesssim g(n)$ to denote $f(n) \leq O(1) \cdot g(n)$.

Formulation of algorithm configuration. Figure 1 illustrates an overview of AC and generalization bounds. Let $\mathcal{A}$ be an algorithm with parameters θ (e.g., a SAT solver). In this paper, we consider algorithms with n bounded continuous parameters. Without loss of generality, we assume $\theta \in [0, 1]^n$.

Let $u(\theta, I, \xi)$ denote the performance metric (e.g., running time) of algorithm $\mathcal{A}$ using parameters θ on instance I. Notice that algorithm $\mathcal{A}$ may be stochastic. Thus, $u(\theta, I, \cdot)$ may be a random variable.

We use ξ to denote the random seed to characterize the randomness of $\mathcal{A}$. Given a pair (θ, I), the algorithm configurator can only access $u(\theta, I, \cdot)$ by randomly sampling a seed $\xi \in \Xi$ and observing $u(\theta, I, \xi)$. Without loss of generality, we assume $u(\cdot,\cdot,\cdot) \in [0,1]$, since we can always normalize the performance metric (e.g., if u represents the running time, we can set a time limit t, and divide the running time by t).

Let $\mathcal{D}$ be a probabilistic distribution of problem instances. Given m independently sampled instances $I_1, \ldots, I_m \sim \mathcal{D}$, an algorithm configurator finds $\tilde{\theta}$ to minimize the loss function $l(\theta) = \mathbb{E}_{I\sim\mathcal{D},\xi\sim\Xi}[u(\theta, I, \xi)]$. Since the configurator only observes the sampled instances $I_1, \ldots, I_m$, it instead minimizes the empirical loss function $\hat{l}(\theta) = \frac{1}{m}\sum_{i=1}^{m} u(\theta, I_i, \xi_i)$, where ξ_i is the randomness of the evaluation on instance I_i.

We are interested in providing generalization guarantees for $l(\tilde{\theta})$ of the learned $\tilde{\theta}$. A generalization guarantee is in the form of

$$|l(\tilde{\theta}) - \hat{l}(\tilde{\theta})| \le \epsilon_{\mathsf{err}},$$

where ϵ_{err} is called the *generalization error* or *generalization gap* of the learned configuration.

2.2 Algorithmic stability

Algorithmic stability is an algorithm-dependent approach to generalization bounds. If a learning algorithm (i.e., the algorithm configurator in our setting) produces similar models (i.e., algorithm parameters) under slight perturbations to training data (i.e., sampled instances), then, the learned model has strong generalization guarantees. The following result can be found in most learning theory textbooks, e.g., Shalev-Shwartz and Ben-David [20].

Definition 2 (Stability)**.** Let $x_1, \ldots, x_m, x'_m \in \mathbb{X}$ be $m+1$ samples from some distribution $\mathcal{D}$. Let $S = \{x_1, \ldots, x_m\}$ and $S' = \{x_1, \ldots, x_{m-1}, x'_m\}$ be two datasets with only one different sample. Suppose L is a learning algorithm that takes a dataset S as inputs, and outputs a parameter $L(S) \in \Theta$. Let $u(\theta, x)$ denote the loss function on data $x \in \mathbb{X}$ using parameter $\theta \in \Theta$. We say L is *ε-stable*, if

$$|\mathbb{E}_L[u(L(S), x)] - \mathbb{E}_L[u(L(S'), x)]| \le \varepsilon,\ \forall\, x \in \mathbb{X},$$

where the expectation $E_L[\cdot]$ is taken over the randomness of learning algorithm L (since the learned parameter $L(S)$ may be stochastic).

Theorem 3. *Let L be an ε-stable learning algorithm. Suppose $S = \{x_1, \ldots, x_m\} \sim \mathcal{D}^m$ are m independent samples from some distribution $\mathcal{D}$, and $u(\theta, x)$ be the loss function. Then, the expected generalization error is upper bounded by*

$$\mathbb{E}_{L,S}\left|\mathbb{E}_{x\sim\mathcal{D}}[u(L(S), x)] - \frac{1}{m}\sum_{i=1}^{m} u(L(S), x_i)\right| \le \varepsilon.$$

Therefore, to prove generalization guarantees for AC, it suffices to show that the algorithm configurator is stable.

Remark 4. Theorem 3 provides a generalization bound that holds in expectation. Feldman and Vondrák [24] show that such results can also be turned into high-probability bounds: For ε-stable learning algorithms, the generalization error is $O\left(\varepsilon \log(m)\log(m/\delta) + \sqrt{\log(1/\delta)/m}\right)$ with probability at least $1-\delta$. Thus, in the following, we only consider generalization bounds in expectation.

3 Generalization Guarantees

In this section, we formally present our generalization guarantee. We first introduce the model-based AC framework studied in our setting. Then, we present the main generalization bound and discuss the implications of our result.

3.1 Model-based algorithm configuration

The framework of model-based algorithm configuration is illustrated in Algorithm 1. The configurator uses the outcomes of sampled runs of algorithm $\mathcal{A}$ to train a surrogate model that predicts the

Algorithm 1 The algorithmic framework of model-based algorithm configurators.

Input: The sampled instances $I_1, \dots, I_m \sim \mathcal{D}$.

Output: A parameter configuration $\tilde{\theta} \in [0,1]^n$ for algorithm $\mathcal{A}$.

Sample T_{init} initial parameters θ_t for $t \in [T_{\mathsf{init}}]$.

Evaluate parameters $\theta_{[T_{\mathsf{init}}]}$ on m instances to obtain $\hat{u}_{ij}$ that estimates $\mathbb{E}_\xi[u(\theta_i, I_j, \xi)]$.

for $t = T_{\mathsf{init}} + 1, \dots, T_{\mathsf{init}} + T_{\mathsf{iter}}$ **do**

  Build a dataset $D = \{\langle(\theta_i, \phi_j), \hat{u}_{ij}\rangle\}_{i \in [t-1], j \in [m]}$, where ϕ_j is the feature vector of problem instance I_j.

  Use D to train a surrogate model that predicts the performance metric function $\mathbb{E}_\xi[u(\theta, I, \xi)]$.

  Use a local search heuristic to find θ_t that maximizes the acquisition function.

  Evaluate parameter θ_t to obtain $\hat{u}_{tj}$ for $j \in [m]$ and update the incumbent parameter.

end for

return $\theta_{T_{\mathsf{total}}} = \theta_{T_{\mathsf{init}} + T_{\mathsf{iter}}}$.

performance metric function $\mathbb{E}_\xi[u(\theta, I, \xi)]$ for any parameter θ and instance I. Based on this model, the configurator searches for a promising parameter, and evaluates this parameter to obtain new data. The surrogate model and the parameters are iteratively updated to incorporate new training data.

In the following, we introduce the algorithmic details of the configurator in our analysis. We emphasize that our setting is generally consistent with algorithm configurators in practice [2], but with some slight modifications to make the generalization error analysis possible. In experiments, we will show that our modifications negligibly affect the performance of the configurator.

Sampling initial parameters. The configurator initially samples T_{init} parameters to "warm-start" the surrogate model. Our analysis works for arbitrary sampling methods: It could be a uniform distribution over the entire parameter space $[0,1]^n$, or any other distributions based on prior knowledge (e.g., applying large language models to enhance the initial samplings [25]), as long as the distribution is fixed conditioning on the instances $I_1, \dots, I_m$.

Building datasets. In the t-th iteration, the configurator has collected the outcomes of $\mathcal{A}$ on parameters $\theta_1, \dots, \theta_{t-1}$. Suppose for each instance I_j and parameter θ_i, algorithm $\mathcal{A}$ is evaluated and the outcomes are $u(\theta_i, I_j, \xi_{ij})$. The configurator may use $\hat{u}_{ij} = u(\theta_i, I_j, \xi_{ij})$ to estimate $\mathbb{E}_\xi[u(\theta_i, I_j, \xi)]$.

Moreover, configurators in practice may also predict the algorithmic performance based on the features of problem instances. We use ϕ_i to denote the feature vector of instance I_i. Let d denote the number of features (i.e., $\phi_i \in \mathbb{R}^d$). The configurator trains a surrogate model to predict $\mathbb{E}_\xi[u(\theta, I, \xi)]$ based on training data $\{\langle(\theta_i, \phi_j), \hat{u}_{ij}\rangle\}_{i \in [t-1], j \in [m]}$.

Surrogate models. In Bayesian optimization, many machine learning models are utilized as surrogate models, such as Gaussian processes and random forests. Algorithm configurators in practice usually use the random forest as the surrogate model, due to its low training time complexity. Random forests can be built in almost linear time, while standard Gaussian processes require $O(N^3)$ time to fit N data points.

In our analysis, we assume the configurator adopts random forests as the surrogate model. The random forest is a classic learning model, which applies the *bagging* heuristic to decision trees. A random forest consists of Q independently built decision trees. Each tree is trained with a randomly sampled subset of the entire dataset. The most common bagging strategy is *sub-bagging*, which uniformly samples m_{bag} out of m instances without replacement and learns the decision tree with data related to these instances in D. In practice, the number of samples for each decision tree is not very large to prevent overfitting. We assume m_{bag} is a constant as we increase the total number of instances m.

Note that in model-based AC, the surrogate model is required to predict both the expectation and the variance. The random forest predicts the expectation by averaging the outputs, and predicts the variance by computing the empirical variance of all trees. Suppose Q is even and Tree_i is the output of the i-th tree. We predict the empirical variance by

$$\mathsf{RF}_{\mathbb{V}} = \frac{1}{Q} \sum_{i=1}^{Q/2} (\mathsf{Tree}_{2i-1} - \mathsf{Tree}_{2i})^2.$$

We assume the predicted empirical variance is lower bounded by a small constant $\sigma_{\min}$, since in practice, $\mathsf{RF}_{\mathbb{V}}$ cannot be arbitrarily small due to the randomness of random forests. The variance ensures that the configurator will explore new parameters instead of purely exploiting.

Acquisition functions. After training a surrogate model for the metric function, the algorithm configurator finds a promising parameter by maximizing an acquisition function. In our analysis, we consider the widely used *expected improvement* (EI) function: Suppose the surrogate model predicts the mean and the variances of $u(\theta, I, \cdot)$ to be $\hat{\mu}$ and $\hat{\sigma}^2$, and the empirical estimation of the incumbent parameter (with currently the best mean) is $\hat{u}'$. The expected improvement of $\hat{u}$ is defined as

$$\mathsf{EI}(\theta) = \mathbb{E}_{\hat{u} \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)}[\max\{\hat{u}' - \hat{u}, 0\}].$$

Sampling promising parameters with local search. In each iteration, the configurator finds a parameter configuration with maximized EI acquisition. Model-based algorithm configurators, such as SMAC [2], apply a local search heuristic to solve this optimization problem. Concretely, suppose the incumbent parameter is θ. The configurator iteratively samples $\theta' \sim \mathcal{N}(\theta, \Delta \cdot I_n)$ for some $\Delta > 0$, and updates $\theta \leftarrow \theta'$ if θ' has a better EI.

However, due to its greedy nature, it is hard to analyze the convergence properties of this local search algorithm. In our analysis, we instead assume applying an MCMC-based algorithm. Concretely, starting from a uniformly sampled initial parameter θ, the algorithm iteratively samples $\theta' \sim \mathcal{N}(\theta, \Delta \cdot I_n)$. The difference is that we accept $\theta \leftarrow \theta'$ with probability $\min\left\{1, \exp\left(\frac{\mathsf{EI}(\theta') - \mathsf{EI}(\theta)}{\tau}\right)\right\}$.

This adaptation is reasonable since it is equivalent to applying the Metropolis-Hastings algorithm to sample θ with probability proportional to $\exp(\mathsf{EI}(\theta)/\tau)$. We can also regard it as a simulated-annealing version of local search with a fixed temperature τ. This assumption allows convergence rate analysis of the sampling algorithm, and negligibly affects the empirical performance as we will show in experiments.

Selecting the best configuration. The algorithm configurator performs T_{iter} iterations in total. In modern algorithm configurators, the configurator maintains an incumbent parameter in each iteration using a so-called *intensification* method. Intensification adaptively allocates algorithm runs to different parameters to boost the accuracy of parameter evaluations. This makes our analysis hard since different configurations may be evaluated with different numbers of runs. In this paper, we simplify the configurator for ease of analysis. In initial samplings, we select a parameter $\theta_i \in \{\theta_1, \ldots, \theta_{T_{\mathsf{init}}}\}$ with minimized $\sum_{j=1}^{m} \hat{u}_{ij}$ as the initial incumbent. In subsequent iterations, we update the incumbent if the latest parameter θ_t has a better empirical metric value.

Let $T_{\mathsf{total}} = T_{\mathsf{init}} + T_{\mathsf{iter}}$ be the total number of sampled configurations. A modification to the algorithm configurator in our analysis is that we let the configurator output $\theta_{T_{\mathsf{total}}}$ instead of the incumbent parameter. We make this modification to ensure that if the trajectory of the configurator (i.e., the sequence $\theta_1, \theta_2, \ldots$) is fixed, the output parameter is also fixed. This negligibly affects the performance as long as the configurator finally converges to a promising region of the parameter space.

3.2 The main theorem

Now, we are ready to present our main generalization bound.

Theorem 5. *Let $u(\theta, I, \xi)$ be the performance metric function on instance I with parameter θ and algorithmic randomness ξ. Let $S = \{I_1, \ldots, I_m\} \sim \mathcal{D}^m$ be m independent problem instances sampled from some distribution $\mathcal{D}$, and $\xi_1, \ldots, \xi_m$ denote the independent random seeds of m calls. Let $\tilde{\theta}$ denote the parameter configuration learned by the model-based algorithm configurator. The expected generalization error of $\tilde{\theta}$ is upper bounded by*

$$\mathop{\mathbb{E}}_{S, \tilde{\theta}} \left| \mathop{\mathbb{E}}_{I \sim \mathcal{D}, \xi \sim \Xi}[u(\tilde{\theta}, I, \xi)] - \frac{1}{m} \sum_{i=1}^{m} u(\tilde{\theta}, I_i, \xi_i) \right| \lesssim \sqrt{T_{\mathsf{iter}}\,(\varepsilon_{\mathsf{Stab}} + \varepsilon_{\mathsf{MH}})},$$

where

$$\varepsilon_{\mathsf{Stab}} = \frac{1}{\tau \cdot \sigma_{\min}} \left(\frac{m_{\mathsf{bag}}}{m} + \sqrt{\frac{(n+d) \log(Q \cdot T_{\mathsf{total}} \cdot m_{\mathsf{bag}})}{Q}} \right),$$

and

$$\varepsilon_{\mathsf{MH}} = \exp\left(-\frac{T_{\mathsf{MH}}}{\sqrt{(2\pi\Delta)^n}}\exp\left(-\frac{n}{2\Delta}-\frac{1}{\tau}\right)+\frac{1}{\tau}\right).$$

Remark 6. We make some comments on this bound:

- The notations $\Delta, \sigma_{\min}, \tau, m_{\mathsf{bag}}$ can be regarded as constants. We focus on the asymptotic behavior with respect to the sample number m, the surrogate model size Q, the number of iterations T_{iter} and T_{MH}, and the dimension n and d of the AC problem considered.
- The generalization bound grows linearly with respect to the root of the total iteration number $\sqrt{T_{\mathsf{iter}}}$. The generalization gap is larger as the training time grows longer. This is similar to stability bounds of (stochastic) gradient descent in the classic machine learning literature [26].
- The $\tilde{O}\left(\sqrt{(n+d)/Q}\right)$ term in $\varepsilon_{\mathsf{Stab}}$ comes from the Rademacher complexity of the *dual* decision tree class. We can numerically compute the empirical Rademacher complexity (see Appendix C) to obtain a tighter data-dependent bound.
- The $\varepsilon_{\mathsf{MH}}$ term comes from the convergence of the local search process for sampling promising parameters in Algorithm 1. We study its convergence by treating it as a Metropolis-Hastings sampling algorithm, and derive a geometric convergence rate. This term can be almost neglected numerically by selecting a large enough T_{MH} (e.g., SMAC [2] performs $T_{\mathsf{MH}} = 10000$ steps) since it converges to zero exponentially.

We sketch the proof of Theorem 5. The complete proof is in Appendix B due to the page limit. Required technical lemmas are listed in Appendix A.

Proof sketch of Theorem 5. Our analysis is based on the algorithmic stability framework of generalization bounds as introduced in Section 2. Thus, throughout our analysis, we define $S = \{I_1, \ldots, I_m\}$ and $S' = \{I_1, \ldots, I_{m-1}, I'_m\}$ as two sets of problem instances, with only one different instance. Let θ_i and θ'_i denote the sampled parameters for datasets S and S'. Recall that $\hat{u}_{ij} = u(\theta_i, I_j, \xi_{ij})$ is the empirical estimation of $\mathbb{E}_\xi[u(\theta_i, I_j, \xi)]$ for parameter θ_i and instance I_j. We also have $\hat{l}(\theta_i) = \frac{1}{m}\sum_{j=1}^m \hat{u}(\theta_i, I_j, \xi_{ij})$ be the empirical estimation of the performance metric. Similarly, we define $\hat{u}', \xi', \hat{l}'$ for dataset S'. To analyze the stability, we fix the randomness of evaluations such that $\xi_{ij} = \xi'_{ij}$ for any i, j. Thus, $\hat{u}_{ij} = \hat{u}'_{ij}$ for any $j \neq m$.

Our proof consists of four steps: Firstly, the configurator samples T_{init} parameters before the main loop. In this step, the distribution of initial parameters is fixed. Secondly, we analyze the stability of the random forest surrogate model. We present a uniform stability bound for both the mean and the variance estimation of the random forest. Thirdly, we turn the stability bound of the surrogate model into the stability of the acquisition function, and study the convergence of the local search sampling process that maximizes the acquisition function. Finally, we put the building blocks together to obtain an algorithmic stability bound for the complete algorithm configuration method by bounding the KL divergence between two search paths on S and S'. This leads to a generalization bound via Theorem 3.

Step 1: Sampling initial parameters. In Lines 1–2 of Algorithm 1, T_{init} initial configurations are randomly sampled to form the initial dataset. Since the initial samplings are only based on prior knowledge, the distributions of $(\theta_1, \ldots, \theta_{T_{\mathsf{init}}})$ and $(\theta'_1, \ldots, \theta'_{T_{\mathsf{init}}})$ are the same. Moreover, since S differs S' by only one instance, it is easy to show that the difference between the empirical estimations (i.e., $\hat{l}(\theta_{\mathsf{inc}}) - \hat{l}'(\theta_{\mathsf{inc}})$) of the incumbent parameter for S and S' is at most $1/m$.

Step 2: Uniform stability of surrogate models. In this step, we aim to show that the prediction of the surrogate model is perturbation-resilient when one instance is modified in the dataset, conditioned on the randomness of the algorithm and the model.

Suppose the configurator is working in the t-th iteration ($T_{\mathsf{init}} < t \leq T_{\mathsf{init}} + T_{\mathsf{iter}}$). The configuration trains the random forest model using $D = \{\langle(\theta_i, \phi_j), \hat{u}_{ij}\rangle\}_{i\in[t], j\in[m]}$ for S, and similarly D' for S'.

Let $\mathsf{Tree}_i(\theta, \phi)$ denote the prediction of the i-th decision tree in the random forest for S. The random forest model predicts the mean and the variance by $\mathsf{RF}_{\mathbb{E}}(\theta, \phi) = \frac{1}{Q}\sum_{i=1}^{Q} \mathsf{Tree}_i(\theta, \phi)$ and $\mathsf{RF}_{\mathbb{V}}(\theta, \phi) = \frac{1}{Q}\sum_{i=1}^{Q/2}(\mathsf{Tree}_{2i-1}(\theta, \phi) - \mathsf{Tree}_{2i}(\theta, \phi))^2$. We similarly define $\mathsf{Tree}', \mathsf{RF}'$ for S'.

We consider the stability of the random forest prediction, i.e., $\mathsf{Gap}_{\mathbb{E}}(\theta, \phi) = \mathsf{RF}_{\mathbb{E}}(\theta, \phi) - \mathsf{RF}'_{\mathbb{E}}(\theta, \phi)$ and $\mathsf{Gap}_{\mathbb{V}}(\theta, \phi) = \mathsf{RF}_{\mathbb{V}}(\theta, \phi) - \mathsf{RF}'_{\mathbb{V}}(\theta, \phi)$. It is easy to show that for a fixed pair of (θ, ϕ), we have $\mathbb{E}[|\mathsf{Gap}_{\mathbb{E}}(\theta, \phi)|] \le m_{\mathsf{bag}}/m$ and $\mathbb{E}[|\mathsf{Gap}_{\mathbb{V}}(\theta, \phi)|] \le 2m_{\mathsf{bag}}/m$, where the expectation is taken over the randomness of the bagging process in random forests.

However, this bound only holds for a single parameter. To derive a stability bound for the complete configurator, we require uniform bounds, i.e., upper bounds for $\sup_{\theta,\phi} \mathsf{Gap}_{\mathbb{E}}(\theta, \phi)$ and $\sup_{\theta,\phi} \mathsf{Gap}_{\mathbb{V}}(\theta, \phi)$. For $\mathsf{Gap}_{\mathbb{E}}(\cdot, \cdot)$, we can directly apply the Rademacher complexity tool to achieve this. Note that the supremum is taken over the input of the random forest. Thus, different from results in classic learning theory, we bound the Rademacher complexity of the dual decision tree class. For $\mathsf{Gap}_{\mathbb{V}}(\cdot, \cdot)$, we apply a similar bound to Massart's lemma to obtain an upper bound in Appendix A.3.

Step 3: An analysis of the Metropolis-Hastings sampling. In model-based algorithm configurators, a local search heuristic is applied to sample promising parameters with a high acquisition function value.

As discussed in Section 3.1, we apply a Metropolis-Hastings-style method to sample θ_t in Line 7 of Algorithm 1. This method samples a parameter such that the target distribution has a density proportional to $\exp(\mathsf{EI}(\theta)/\tau)$. To analyze the stability of Metropolis-Hastings, two building blocks are required. We first analyze the stability of the target distribution, i.e., the difference between $\mathsf{EI}(\theta)$ for S and S'. Then, we analyze the convergence rate of the sampling method.

Let μ and σ^2 denote the mean and the variance predictions of the surrogate model for S. Let $\hat{l}_{\mathsf{inc}}$ denote the incumbent performance for S. We similarly define μ', σ' and $\hat{l}'_{\mathsf{inc}}$ for S'. By showing that $\mathsf{EI}(\mu, \sigma)$ is $\frac{1}{2\pi}$-Lipschitz w.r.t. σ, we have $|\mathsf{EI}(\mu, \sigma) - \mathsf{EI}'(\mu', \sigma')| \le |\hat{l}_{\mathsf{inc}} - \hat{l}'_{\mathsf{inc}}| + |\mu - \mu'| + |\sigma - \sigma'|/(2\pi)$. In Step 2, we have already obtained stability bounds on $|\mu - \mu'|$ and $|\sigma - \sigma'|$. Moreover, we directly have $|\hat{l}_{\mathsf{inc}} - \hat{l}'_{\mathsf{inc}}| \le 1/m$. Thus, we have a stability bound on the expected improvement acquisition function.

Next, by applying a convergence bound in Appendix A.4, we have a convergence rate of the Metropolis-Hastings method. Note that, unlike the classic mixing time analysis in Markov chains, our analysis requires a convergence bound on the ratio of the sampling distribution and the target distribution. Combining the above two bounds yields an upper bound on the ratio of the probability densities of θ_t for S and S'.

Step 4: Putting things together. Finally, we consider the complete algorithm configurator. Let $\tilde{\theta}$ and $\tilde{\theta}'$ denote the output of the configurator on S and S', respectively. Let $\Theta = (\theta_t)_t$ and $\Theta' = (\theta'_t)_t$ denote the search paths of the configurator. Let $p(\cdot)$ denote the probability density. We show that

$$
\begin{aligned}
\mathop{\mathbb{E}}_{\tilde{\theta},\tilde{\theta}'}\left[|l(\tilde{\theta}) - l(\tilde{\theta}')|\right] &= \int_x l(x) \cdot \left|p(\tilde{\theta} = x) - p(\tilde{\theta}' = x)\right| \mathrm{d}x \\
&\le 2\mathsf{TV}(\Theta, \Theta') \\
&\le \sqrt{2\mathsf{KL}(\Theta \| \Theta')} = \sqrt{2 \mathop{\mathbb{E}}_{x \sim \Theta}\left[\log \frac{p(\Theta = x)}{p(\Theta' = x)}\right]}
\end{aligned}
$$

by Pinsker's inequality. To upper-bound the KL divergence, it suffices to bound the density ratio between two paths. Fixing a path $x = (x_t)_t$, we have

$$
\frac{p(\Theta = x)}{p(\Theta' = x)} = \prod_{t=T_{\mathsf{init}}+1}^{T_{\mathsf{total}}} \frac{p(\theta_t = x_t | \theta_{[t-1]} = x_{[t-1]})}{p(\theta'_t = x_t | \theta'_{[t-1]} = x_{[t-1]})}.
$$

Plugging in the bounds in Steps 2 and 3 yields the desired bound for the stability of the algorithm configurator. □

4 Conclusion and Future Research

In this paper, we present an algorithm-dependent generalization bound for model-based algorithm configurators, based on the algorithmic stability framework. To the best of our knowledge, this is the first generalization bound that applies to a model closely approximating practical model-based algorithm configurators.

We briefly discuss some directions for future research. A limitation of our work is that the generalization bound is still loose compared with practice. In practice, a small number of instances is sufficient to learn a good parameter configuration. Sometimes, even a single instance works for algorithm configuration [2]. It would be interesting to establish connections between algorithm configuration and few-shot learning. To tighten the generalization bound, we may need to find novel characterizations of the performance metric landscape to derive more sophisticated data-dependent bounds. It is also interesting to extend our results to other advanced surrogate models, e.g., tree-structured Parzen estimators [27].

Another significant line of research involves providing theoretical guarantees for the empirical risk minimization problem of algorithm configuration, i.e., explaining why model-based algorithm configurators can find nearly optimal parameters in real-world scenarios. Due to the black-box nature of algorithm configuration, this question is extremely difficult, and may require stronger assumptions and sophisticated beyond-worst-case analysis to answer it.

Acknowledgements

We thank anonymous NeurIPS reviewers for the helpful comments on this paper. This work is supported by the Key Program of National Natural Science Foundation of China (No. 62034005).

References

[1] Frank Hutter, Marius Lindauer, Adrian Balint, Sam Bayless, Holger Hoos, and Kevin Leyton-Brown. The configurable SAT solver challenge (CSSC). *arXiv preprint arXiv 1505.01221*, 2015.

[2] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the International Conference on Learning and Intelligent Optimization*, page 507–523, 2005.

[3] Belarmino Adenso-Díaz and Manuel Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1):99–114, 2006.

[4] Frank Hutter, Holger H. Hoos, and Thomas Stützle. Automatic algorithm configuration based on local search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, page 1152–1157, 2007.

[5] Elias Schede, Jasmin Brandt, Alexander Tornede, Marcel Wever, Viktor Bengs, Eyke Hullermeier, and Kevin Tierney. A survey of methods for automated algorithm configuration. *Journal of Artificial Intelligence Research*, 75:425–487, 2022.

[6] Maria-Florina Balcan. Data-driven algorithm design. *arXiv preprint arXiv 2011.07177*, 2020.

[7] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

[8] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 344–353. PMLR, 2018.

[9] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. Refined bounds for algorithm configuration: The knife-edge of dual class approximability. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 580–590, 2020.

[10] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. Structural analysis of branch-and-cut and the learnability of Gomory mixed integer cuts. In *Advances in Neural Information Processing Systems*, volume 35, pages 33890–33903, 2022.

[11] Avrim Blum, Chen Dan, and Saeed Seddighin. Learning complexity of simulated annealing. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pages 1540–1548, 2021.

[12] Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. How much data is sufficient to learn high-performing algorithms? Generalization guarantees for data-driven algorithm design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, page 919–932, 2021.

[13] Zhiyang Chen, Hailong Yao, and Xia Yin. Learning configurations for data-driven multi-objective optimization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 9642–9663. PMLR, 2025.

[14] Zhiyang Chen, Hailong Yao, and Xia Yin. Learning a provably good wirelength model for analytical placement. In *International Symposium of Electronics Design Automation (ISEDA)*, pages 358–363, 2023.

[15] Zhiyang Chen, Tsung-Yi Ho, Ulf Schlichtmann, Datao Chen, Mingyu Liu, Hailong Yao, and Xia Yin. NeuroEscape: Ordered escape routing via Monte-Carlo tree search and neural network. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 01–09, 2023.

[16] Shengcai Liu, Ke Tang, Yunwen Lei, and Xin Yao. On performance estimation in automatic algorithm configuration. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 2384–2391. AAAI Press, 2020.

[17] Maria-Florina Balcan, Mikhail Khodak, Dravyansh Sharma, and Ameet Talwalkar. Provably tuning the ElasticNet across instances. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

[18] Maria-Florina Balcan, Anh Tuan Nguyen, and Dravyansh Sharma. New bounds for hyperparameter tuning of regression problems across instances. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

[19] Maria-Florina Balcan and Dravyansh Sharma. Learning accurate and interpretable decision trees. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

[20] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[21] Jake A. Soloff, Rina Foygel Barber, and Rebecca Willett. Bagging provides assumption-free stability. *Journal of Machine Learning Research*, 25(131):1–35, 2024.

[22] Yan Wang, Huaiqing Wu, and Dan Nettleton. Stability of random forests and coverage of random-forest prediction intervals. In *Advances in Neural Information Processing Systems*, volume 36, pages 31558–31569, 2023.

[23] Austin Brown and Galin L. Jones. Convergence rates of Metropolis–Hastings algorithms. *WIREs Computational Statistics*, 16(5):e70002, 2024.

[24] Vitaly Feldman and Jan Vondrák. High probability generalization bounds for uniformly stable algorithms with nearly optimal rate. In *Proceedings of Conference on Learning Theory*, pages 1270–1279, 2019.

[25] Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *The Twelfth International Conference on Learning Representations*, 2024.

[26] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2nd edition, 2018.

[27] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, page 2546–2554, 2011.

[28] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv 1012.2599*, 2010.

[29] Tai-Hsuan Wu, Azadeh Davoodi, and Jeffrey T. Linderoth. GRIP: Global routing via integer programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(1):72–84, 2011.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim all the contributions of this work in the abstract and the introduction section.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of this work in Section 4.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All theoretical assumptions are provided in Section 3.1. All proofs are provided in the Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experimental details are presented in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will release the codes upon the acceptance of the paper.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental details are presented in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Error bars are given in Table 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We discuss the experimental environment in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This work generally focuses on the theoretical foundations of algorithm configuration. There is no societal impact that we feel obliged to discuss here.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: There is no data or model that have a risk for misuse in this work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We state all assets as well as their authors in this paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We release no new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: This work is not relevant to LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Technical Lemmas

In this section, we review and prove some technical lemmas that are useful in the proof of the main theorem. Results in Appendices A.1 and A.2 can be found in most learning theory textbooks, e.g., Mohri et al. [26].

A.1 Information theory

Definition 7 (Total variation distance and Kullback-Leibler divergence)**.** Suppose P and Q are two distributions over the same domain. Let $p(\cdot)$ and $q(\cdot)$ denote their probability density functions, respectively. The *total variation distance* between P and Q is defined as

$$\mathsf{TV}(P, Q) = \frac{1}{2} \int_x |p(x) - q(x)| \mathrm{d}x.$$

The *KL divergence* of P over Q is defined as

$$\mathsf{KL}(P \| Q) = \mathop{\mathbb{E}}_{x \sim P} \left[\log \frac{p(x)}{q(x)} \right].$$

Theorem 8 (Pinsker inequality)**.** *Suppose P and Q are two distributions over the same domain. We have $\mathsf{TV}(P, Q) \le \sqrt{\frac{1}{2} \mathsf{KL}(P \| Q)}$.*

A.2 Rademacher complexity

Definition 9 (Rademacher complexity)**.** Let $\mathcal{H}$ denote a set of functions from some domain $\mathbb{X}$ to $[0, 1]$. The *empirical Rademacher complexity* of $\mathcal{H}$ for a finite set $S = \{x_1, \ldots, x_m\} \subseteq \mathbb{X}$ is

$$\hat{\mathcal{R}}_S(\mathcal{H}) = \mathop{\mathbb{E}}_{\sigma \sim \{-1,1\}^m} \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \sigma_i h(x_i) \right],$$

where each σ_i is independently and uniformly sampled from $\{-1, 1\}$. The *Rademacher complexity* of $\mathcal{H}$ is $\mathcal{R}_{\mathcal{D}}(\mathcal{H}) = \mathbb{E}_{S \sim \mathcal{D}^m}[\hat{\mathcal{R}}_S(\mathcal{H})]$ where each $x_i \in S$ is independently sampled from some distribution $\mathcal{D}$.

Theorem 10 (Uniform convergence)**.** *Given m i.i.d. samples $S = \{x_1, \ldots, x_m\}$ from a distribution $\mathcal{D}$, for a function class $\mathcal{H}$ from $\mathbb{X}$ to $[0, 1]$, we have*

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m} \left[\sup_{h \in \mathcal{H}} \left(\frac{1}{m} \sum_{i=1}^m h(x_i) - \mathop{\mathbb{E}}_{x \sim \mathcal{D}}[h(x)] \right) \right] \le 2 \mathcal{R}_{\mathcal{D}}(\mathcal{H}).$$

Thus, Rademacher complexity can be applied to obtain uniform convergence concentration for empirical means. To obtain an upper bound of Rademacher complexity, the following Massart's lemma is popular.

Lemma 11 (Massart)**.** *Let $A \subseteq \mathbb{R}^m$ be a finite set, and $\sigma_1, \ldots, \sigma_m$ be the Rademacher variables. We have*

$$\mathop{\mathbb{E}}_{\sigma \sim \{-1,1\}^m} \left[\frac{1}{m} \sup_{x \in A} \sum_{i=1}^m \sigma_i x_i \right] \le \max_{x \in A} \|x\|_2 \cdot \frac{\sqrt{2 \log |A|}}{m}.$$

Corollary 12. *For a function class $\mathcal{H}$ from $\mathbb{X}$ to $[0, 1]$, and $S = \{x_1, \ldots, x_m\} \subseteq \mathbb{X}$, we have*

$$\hat{\mathcal{R}}_S(\mathcal{H}) \le \sqrt{\frac{2 \log |A|}{m}},$$

where $A = \{(h(x_1), \ldots, h(x_m)) \mid h \in \mathcal{H}\}$ is the set of all possible values of $h \in \mathcal{H}$ on S.

A.3 Rademacher complexity for empirical variances

The classic Rademacher complexity is proposed to study the expectation of an empirical process. In this work, we also need to analyze the empirical variance estimation. To address this problem, we make a small modification to the definition of Rademacher complexity.

Definition 13 (Rademacher complexity)**.** Let $\mathcal{H}$ denote a set of functions from some domain $\mathbb{X}^2$ to $[0, 1]$ and $m > 0$ be an even number. The *empirical Rademacher complexity* of $\mathcal{H}$ for a finite set $S = \{x_1, \ldots, x_m\} \subseteq \mathbb{X}$ is

$$\hat{\mathcal{R}}_S(\mathcal{H}) = \mathop{\mathbb{E}}_{\sigma \sim \{-1,1\}^{m/2}} \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m/2} \sigma_i h(x_{2i-1}, x_{2i}) \right],$$

where each σ_i is independently and uniformly sampled from $\{-1, 1\}$. The *Rademacher complexity* of $\mathcal{H}$ is $\mathcal{R}_{\mathcal{D}}(\mathcal{H}) = \mathbb{E}_{S \sim \mathcal{D}^m}[\hat{\mathcal{R}}(\mathcal{H})]$ where each x_i is independently sampled from distribution $\mathcal{D}$.

Similar to Theorem 10, we also have a uniform convergence theorem. The proof is similar to Theorem 10, which applies the classic symmetrization technique. We provide the proof for the sake of completeness.

Theorem 14. *Given m i.i.d. samples $S = \{x_1, \ldots, x_m\}$ from a distribution $\mathcal{D}$, for a symmetric function class $\mathcal{H}$ from $\mathbb{X}^2$ to $[0, 1]$, we have*

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m} \left[\sup_{h \in \mathcal{H}} \left(\frac{1}{m} \sum_{i=1}^{m/2} h(x_{2i-1}, x_{2i}) - \mathop{\mathbb{E}}_{x, x' \sim \mathcal{D}} [h(x, x')] \right) \right] \le 2\mathcal{R}_{\mathcal{D}}(\mathcal{H}).$$

Proof. For simplicity of notations, we define

$$\hat{\mathbb{E}}_S[h] = \frac{1}{m} \sum_{i=1}^{m/2} h(x_{2i-1}, x_{2i}),$$

and use $\mathbb{E}_{x,x'}[h]$ to denote $\mathbb{E}_{x,x'}[h(x, x')]$ for brevity. Let $S' = \{x'_1, \ldots, x'_m\}$ be an independent copy of S and $\sigma \sim \{-1, 1\}^{m/2}$ be the Rademacher variables. We have

$$\begin{aligned}
\mathbb{E}_S \left[\sup_{h \in \mathcal{H}} \left(\hat{\mathbb{E}}_S(h) - \mathbb{E}_{x,x'}(h) \right) \right] &= \mathbb{E}_S \left[\sup_{h \in \mathcal{H}} \left(\hat{\mathbb{E}}_S(h) - \mathbb{E}_{S' \sim \mathcal{D}^m} \hat{\mathbb{E}}'_S(h) \right) \right] \\
&\le \mathbb{E}_{S,S'} \left[\sup_h \left(\hat{\mathbb{E}}_S(h) - \hat{\mathbb{E}}_{S'}(h) \right) \right] \\
&= \mathbb{E}_{S,S'} \left[\sup_h \left(\frac{1}{m} \sum_{i=1}^{m/2} (h(x_{2i-1}, x_{2i}) - h(x'_{2i-1}, x'_{2i})) \right) \right] \\
&= \mathbb{E}_{S,S',\sigma} \left[\sup_h \left(\frac{1}{m} \sum_{i=1}^{m/2} \sigma_i (h(x_{2i-1}, x_{2i}) - h(x'_{2i-1}, x'_{2i})) \right) \right] \\
&\le 2\mathcal{R}_{\mathcal{D}}(\mathcal{H}),
\end{aligned}$$

where the second line is due to Jensen's inequality, and the fourth line is due to the fact that S' is identically distributed to S. □

We can upper bound the modified Rademacher complexity by the following theorem. The proof is simply by applying Massart's lemma. The only difference is the constant, since we only have $m/2$ Rademacher variables here.

Theorem 15. *For a symmetric function class $\mathcal{H}$ from $\mathbb{X}^2$ to $[0, 1]$, $S = \{x_1, \ldots, x_m\} \subseteq \mathbb{X}$ with m being even, we have*

$$\hat{\mathcal{R}}_S(\mathcal{H}) \le \sqrt{\frac{4 \log |A|}{m}},$$

where $A = \{(h(x_i, x_j))_{ij} \mid h \in \mathcal{H}\}$ is the set of all possible values of $h \in \mathcal{H}$ on S.

A.4 Metropolis-Hastings sampling

A Metropolis-Hastings algorithm generates a Markov chain $(x_t)_{t\geq 0}$ such that x_t converges to a given distribution Π. The algorithm chooses a *proposal distribution* $Q(\cdot)$, and an initial distribution Q_0. Starting with $x_0 \sim Q_0$, we independently generate $x'_{t+1}|x_t \sim Q(x_t)$ and a random variable $\delta \sim \text{Unif}[0,1]$ that is uniformly sampled over $[0,1]$. The Markov chain is defined by

$$x_{t+1} = \begin{cases} x'_{t+1}, \text{ if } \delta \leq \dfrac{\pi(x'_{t+1})q(x_t|x'_{t+1})}{\pi(x_t)q(x'_{t+1}|x_t)}, \\ x_t, \text{ otherwise,} \end{cases}$$

where $\pi(\cdot)$ is the probability density of the target distribution Π, and $q(\cdot)$ is the density of the proposal distribution Q. Note that the proposal distribution is usually symmetric (i.e., $q(a|b) = q(b|a)$, as is the case of Gaussian proposals in this work). Thus, the acceptance probability can be reduced to the ratio of the target density.

In this work, we are interested in the convergence rate of the Metropolis-Hastings algorithm, i.e., how fast the Markov chain x_t converges to Π. Previous work [23] has been devoted to studying the *mixing time* of Metropolis-Hastings, i.e., the number of iterations such that the total variation distance between x_t and Π is small enough. However, in the proof of our main theorem, a KL divergence bound is required. Instead, we need a uniform convergence bound on the probability density ratio of the sample and the target distribution.

Definition 16 (Minorization [23])**.** We say the proposal distribution satisfies the *global minorization condition*, if $q(x'_{t+1}|x_t) \geq \alpha\pi(x_t)$ for some $\alpha \in (0,1)$.

Theorem 17. *Suppose the proposal probability density $q(\cdot)$ is symmetric and satisfies the global minorization condition. Let $q_t(\cdot)$ denote the probability density of x_t and $q_0(\cdot)$ is the density of the initial distribution. If for any x,*

$$1 - \alpha_0 \leq \frac{q_0(x)}{\pi(x)} \leq 1 + \alpha_0,$$

then, for any x, we also have

$$1 - \alpha_0 \cdot (1-\alpha)^t \leq \frac{q_t(x)}{\pi(x)} \leq 1 + \alpha_0 \cdot (1-\alpha)^t.$$

Proof. Let $\varphi(x', x) = \min\left\{1, \frac{\pi(x')}{\pi(x)}\right\}$. It is easy to note that $\min\{\pi(x), \pi(x')\} = \varphi(x', x)\pi(x) = \varphi(x, x')\pi(x')$. Throughout this proof, we use $\int$ to denote the integral over the support of the distribution Π. By definition, we have

$$\begin{aligned} q_{t+1}(x') &= \left(\int q_t(x)q(x'|x)\varphi(x', x)\mathrm{d}x + \int q_t(x')q(x'|x)(1 - \varphi(x, x'))\mathrm{d}x\right) \\ &= q_t(x') + \int \left(q_t(x)\varphi(x', x) - q_t(x')\varphi(x, x')\right) q(x'|x)\mathrm{d}x. \end{aligned}$$

If $\pi(x') \geq \pi(x)$, we have

$$\begin{aligned} \int \left(q_t(x)\varphi(x', x) - q_t(x')\varphi(x, x')\right) q(x'|x)\mathrm{d}x &= \int \left(q_t(x) - q_t(x')\frac{\pi(x)}{\pi(x')}\right) q(x'|x)\mathrm{d}x \\ &= \int \left(\frac{q_t(x)}{\pi(x)} - \frac{q_t(x')}{\pi(x')}\right) \varphi(x, x')\pi(x')q(x'|x)\mathrm{d}x. \end{aligned}$$

If $\pi(x') < \pi(x)$, we have

$$\begin{aligned} \int \left(q_t(x)\varphi(x', x) - q_t(x')\varphi(x, x')\right) q(x'|x)\mathrm{d}x &= \int \left(q_t(x)\frac{\pi(x')}{\pi(x)} - q_t(x')\right) q(x'|x)\mathrm{d}x \\ &= \int \left(\frac{q_t(x)}{\pi(x)} - \frac{q_t(x')}{\pi(x')}\right) \varphi(x', x)\pi(x')q(x'|x)\mathrm{d}x. \end{aligned}$$

Combining the above two cases, we have

$$q_{t+1}(x') = q_t(x') + \int \left(\frac{q_t(x)}{\pi(x)} - \frac{q_t(x')}{\pi(x')} \right) q(x'|x) \min\{\pi(x), \pi(x')\} \mathrm{d}x.$$

Let $E_t(x) = \frac{q_t(x)}{\pi(x)} - 1$. We have

$$\begin{aligned} E_{t+1}(x') &= E_t(x') + \int (E_t(x) - E_t(x')) \frac{q(x'|x) \min\{\pi(x), \pi(x')\}}{\pi(x')} \mathrm{d}x \\ &= E_t(x') \left(1 - \int \varphi(x, x') q(x|x') \mathrm{d}x \right) + \int E_t(x) \varphi(x, x') q(x|x') \mathrm{d}x. \end{aligned}$$

Since $\varphi(\cdot, \cdot) \leq 1$, we have $1 - \int \varphi(x, x') q(x|x') \mathrm{d}x \geq 0$. Therefore, we have

$$\begin{aligned} E_{t+1}(x') &\leq \sup_{x_0} E_t(x_0) \left(1 - \int \varphi(x, x') q(x|x') \mathrm{d}x \right) + \int E_t(x) \varphi(x, x') q(x|x') \mathrm{d}x \\ &= \sup_{x_0} E_t(x_0) - \int \left(\sup_{x_0} E_t(x_0) - E_t(x) \right) \min \left\{ q(x|x'), q(x|x') \frac{\pi(x)}{\pi(x')} \right\} \mathrm{d}x \\ &\leq \sup_{x_0} E_t(x_0) - \alpha \int \left(\sup_{x_0} E_t(x_0) - E_t(x) \right) \pi(x) \mathrm{d}x \\ &= (1 - \alpha) \sup_{x_0} E_t(x_0) + \alpha \int \left(\frac{q_t(x)}{\pi(x)} - 1 \right) \pi(x) \mathrm{d}x \\ &= (1 - \alpha) \sup_{x_0} E_t(x_0). \end{aligned}$$

Since this bound holds for any x', we have

$$\sup_{x_0} E_{t+1}(x_0) \leq (1 - \alpha) \sup_{x_0} E_t(x_0),$$

which yields the desired bound by induction. □

B Proof of Theorem 5

In this section, we prove the main theorem. We first introduce the necessary notations for our proof. Then, we prove some building blocks in Sections B.2 to B.4. Finally, we put things together to obtain the desired bound.

B.1 Step 0: Notation preparations

Our analysis is based on the algorithmic stability framework of generalization bounds. Thus, throughout our analysis, we define $S = \{I_1, \ldots, I_m\}$ and $S' = \{I_1, \ldots, I_{m-1}, I'_m\}$ are two sets of problem instances, with only one different instance. We study the difference between the behaviors of the algorithm configurator on S and S'.

Let $\theta_1, \ldots, \theta_{T_{\mathsf{init}}}$ and $\theta'_1, \ldots, \theta'_{T_{\mathsf{init}}}$ denote the sampled parameters using datasets S and S'. Let $\hat{u}_{ij} = u(\theta_i, I_j, \xi_{ij})$ denote the empirical estimation of $\mathbb{E}_\xi[u(\theta_i, I_j, \xi)]$ for S and $\hat{l}(\theta_i) = \frac{1}{m} \sum_{j=1}^m \hat{u}(\theta_i, I_j, \xi_{ij})$ be the empirical metric function of S. Similarly, we define $\hat{u}'_{ij}$, ξ'_{ij}, and $\hat{l}'$ for S'.

We fix the randomness of evaluations such that $\xi_{ij} = \xi'_{ij}$ for any i, j. Thus, if $j \neq m$, we have $\hat{u}_{ij} = \hat{u}'_{ij}$. This helps us analyze the stability of the configurator under stochasticity.

B.2 Step 1: Sampling initial parameters

Initially, the algorithm configurator randomly samples T_{init} parameters and selects the best one as the incumbent. Since the initial samplings are only based on prior knowledge, we directly have the following claim.

Claim 18. *The parameters* $(\theta_1, \ldots, \theta_{T_{\mathsf{init}}})$ *and* $(\theta'_1, \ldots, \theta'_{T_{\mathsf{init}}})$ *follow an identical distribution.*

Moreover, since S differs S' by only one instance, the empirical estimations of incumbents are close to each other, conditioned on the evaluation randomness. Let θ_{inc} and θ'_{inc} denote the best parameter among $(\theta_1, \ldots, \theta_{T_{\mathsf{init}}})$ and $(\theta'_1, \ldots, \theta'_{T_{\mathsf{init}}})$, respectively. We have the following lemma.

Lemma 19. *If* $\theta_i = \theta'_i$ *for any* $i \in [T_{\mathsf{init}}]$, *we have* $\left|\hat{l}(\theta_{\mathsf{inc}}) - \hat{l}'(\theta'_{\mathsf{inc}})\right| \leq \frac{1}{m}$.

Proof. Since $\hat{u}_{ij} = \hat{u}'_{ij}$ for any $i \in [T_{\mathsf{init}}], j \in [m-1]$, we have $\left|\hat{l}(\theta_i) - \hat{l}'(\theta'_i)\right| = \frac{1}{m}|\hat{u}_{im} - \hat{u}'_{im}| \leq \frac{1}{m}$ for any $i \in [T_{\mathsf{init}}]$. Thus, we have

$$\hat{l}(\theta_{\mathsf{inc}}) \leq \hat{l}'(\theta_{\mathsf{inc}}) + \frac{1}{m} \leq \max_i \hat{l}'(\theta_i) + \frac{1}{m} = \hat{l}'(\theta'_{\mathsf{inc}}) + \frac{1}{m}.$$

Similarly, we also have $\hat{l}'(\theta'_{\mathsf{inc}}) \leq \hat{l}(\theta_{\mathsf{inc}}) + \frac{1}{m}$. Combining these two inequalities yields the desired bound. □

B.3 Step 2: Uniform stability of surrogate models

Now, we consider the stability of the surrogate model. We aim to show that the prediction of the surrogate model is perturbation-resilient when one instance is modified in the dataset, conditioned on the randomness of the algorithm and the model.

Suppose the configurator is working on the t-th iteration for some $t \in [T_{\mathsf{init}} + 1, T_{\mathsf{init}} + T_{\mathsf{iter}}]$. We study the learned surrogate model using datasets $D = \{\langle(\theta_i, \phi_j), \hat{u}_{ij}\rangle\}_{i\in[t],j\in[m]}$ and $D' = \{\langle(\theta'_i, \phi'_j), \hat{u}'_{ij}\rangle\}_{i\in[t],j\in[m]}$.

Since we fix the randomness of the algorithm configurator, we assume the sampled parameters $(\theta_1, \ldots, \theta_t) = (\theta'_1, \ldots, \theta'_t)$. Since S and S' differs by only one instance, we have $\phi_j = \phi'_j$, $\hat{u}_{ij} = \hat{u}'_{ij}$ for any $i \in [t]$ and $j \in [m-1]$. Suppose the random forest surrogate model consists of Q decision trees. We use $\mathsf{Tree}_i(\theta, \phi)$ to denote the output of the i-th decision tree on parameter θ and instance feature ϕ using training data D. We use

$$\mathsf{RF}_{\mathbb{E}}(\theta, \phi) = \frac{1}{Q}\sum_{i=1}^{Q} \mathsf{Tree}_i(\theta, \phi)$$

and

$$\mathsf{RF}_{\mathbb{V}}(\theta, \phi) = \frac{1}{Q}\sum_{i=1}^{Q/2} (\mathsf{Tree}_{2i-1}(\theta, \phi) - \mathsf{Tree}_{2i}(\theta, \phi))^2$$

to denote the empirical expectation and the variance prediction, respectively. Similarly, we define Tree'_i, $\mathsf{RF}'_{\mathbb{E}}$ and $\mathsf{RF}'_{\mathbb{V}}$ for D'.

Let $\mathsf{Gap}_{\mathbb{E}}(\theta, \phi) = \mathsf{RF}_{\mathbb{E}}(\theta, \phi) - \mathsf{RF}'_{\mathbb{E}}(\theta, \phi)$ and $\mathsf{Gap}_{\mathbb{V}}(\theta, \phi) = \mathsf{RF}_{\mathbb{V}}(\theta, \phi) - \mathsf{RF}'_{\mathbb{V}}(\theta, \phi)$. We can bound the stability of the prediction as follows.

Lemma 20. *For any parameter* θ *and instance feature* ϕ, *we have*

$$\mathbb{E}[|\mathsf{Gap}_{\mathbb{E}}(\theta, \phi)|] \leq \frac{m_{\mathsf{bag}}}{m},\ \mathbb{E}[|\mathsf{Gap}_{\mathbb{V}}(\theta, \phi)|] \leq \frac{2m_{\mathsf{bag}}}{m},$$

where the expectation is taken over the randomness of the bagging.

Proof. Note that $|\mathsf{Gap}_{\mathbb{E}}(\theta, \phi)| \leq \frac{1}{Q}\sum_{i=1}^{Q} |\mathsf{Tree}_i(\theta, \phi) - \mathsf{Tree}'_i(\theta, \phi)|$. Since S and S' differs by only one instance I_m and I'_m, we have $\mathsf{Tree}_i(\theta, \phi) \neq \mathsf{Tree}'_i(\theta, \phi)$ with probability m_{bag}/m. Thus, $\mathbb{E}[|\mathsf{Gap}_{\mathbb{E}}(\theta, \phi)|] \leq \frac{1}{Q}\sum_{i=1}^{Q} m_{\mathsf{bag}}/m = m_{\mathsf{bag}}/m$.

Similarly, $|\mathsf{Gap}_{\mathbb{V}}(\theta, \phi)| \leq \frac{1}{Q}\sum_{i=1}^{Q/2} |(\mathsf{Tree}_{2i-1}(\theta, \phi) - \mathsf{Tree}_{2i}(\theta, \phi))^2 - (\mathsf{Tree}'_{2i-1}(\theta, \phi) - \mathsf{Tree}'_{2i}(\theta, \phi))^2|$. With probability $1 - (1 - m_{\mathsf{bag}}/m)^2 \leq 2m_{\mathsf{bag}}/m$, we have that $\mathsf{Tree}_i(\theta, \phi) \neq \mathsf{Tree}'_i(\theta, \phi) \vee \mathsf{Tree}_j(\theta, \phi) \neq \mathsf{Tree}'_j(\theta, \phi)$ holds. Thus, $\mathbb{E}[|\mathsf{Gap}_{\mathbb{V}}(\theta, \phi)|] \leq 2m_{\mathsf{bag}}/m$. □

However, this bound only holds for a single parameter in expectation. The subsequent analysis requires a uniform bound that holds for any θ. We can use the classic tool of Rademacher complexity to achieve this.

Let $\mathsf{Gap}_{\mathbb{E}}^{(i)}(\theta, \phi) = \mathsf{Tree}_i(\theta, \phi) - \mathsf{Tree}'_i(\theta, \phi)$. We can regard $\mathsf{Gap}_{\mathbb{E}}^{(i)}(\cdot, \cdot)$ as i.i.d. samples of $\mathbb{E}[\mathsf{Gap}_{\mathbb{E}}(\cdot, \cdot)]$. Note that $\mathsf{Gap}^{(i)}$ is a function that maps a pair of θ and ϕ to the stability gap on a fixed decision tree i. We define the dual function of Gap as $\mathsf{DualGap}_{\mathbb{E}}^{(\theta,\phi)} : T \mapsto \mathsf{Tree}_T(\theta, \phi) - \mathsf{Tree}'_T(\theta, \phi)$, where T is a bagging sequence of the random forest that induces a decision tree. For the i-th decision tree in the random forest, we use $\mathsf{DualGap}_{\mathbb{E}}^{(\theta,\phi)}(i)$ to denote $\mathsf{Gap}_{\mathbb{E}}^{(i)}(\theta, \phi)$.

Let $\mathcal{G}_{\mathbb{E}} = \{\mathsf{DualGap}_{\mathbb{E}}^{(\theta,\phi)} : T \mapsto \mathsf{Tree}_T(\theta, \phi) - \mathsf{Tree}'_T(\theta, \phi) \mid (\theta, \phi)\}$ (T is an arbitrary valid decision tree model, including $\mathsf{Tree}_1, \ldots, \mathsf{Tree}_Q$) be the set of dual functions of Gap that maps decision trees to prediction values. The empirical Rademacher complexity of $\mathcal{G}_{\mathbb{E}}$ is

$$\hat{\mathcal{R}}(\mathcal{G}_{\mathbb{E}}) = \mathop{\mathbb{E}}_{\sigma \sim \{1,-1\}^Q} \left[\sup_{(\theta,\phi)} \frac{1}{Q} \sum_{i=1}^{Q} \sigma_i \mathsf{DualGap}_{\mathbb{E}}^{(\theta,\phi)}(i) \right] = \mathop{\mathbb{E}}_{\sigma \sim \{1,-1\}^Q} \left[\sup_{(\theta,\phi)} \frac{1}{Q} \sum_{i=1}^{Q} \sigma_i \mathsf{Gap}_{\mathbb{E}}^{(i)}(\theta, \phi) \right].$$

Lemma 21. *We have*

$$\hat{\mathcal{R}}(\mathcal{G}_{\mathbb{E}}) \le 2\sqrt{\frac{(n+d)(1 + \log(2Q \cdot t \cdot m_{\mathsf{bag}}))}{Q}},$$

where t is the current iteration of the configurator.

Proof. Massart's lemma shows that for a function class $\mathcal{H}$ for a finite set $\{x_1, \ldots, x_m\}$, we have $\mathcal{R}(\mathcal{H}) \le \sqrt{\frac{2\log|A|}{m}}$, where $A = \{(h(x_1), \ldots, h(x_m)) \mid h \in \mathcal{H}\}$. To apply Massart's lemma, we need to bound the number of possible values of $\left(\mathsf{Gap}_{\mathbb{E}}^{(1)}(\theta, \phi), \ldots, \mathsf{Gap}_{\mathbb{E}}^{(Q)}(\theta, \phi)\right)$ for all (θ, ϕ). Note that each decision tree Tree_i is built upon $t \cdot m_{\mathsf{bag}}$ data points. Each decision tree has at most $t \cdot m_{\mathsf{bag}}$ leaves. Each leaf of a decision tree is a hyper-rectangle in $(n+d)$-dimension space, where n is the number of parameters and d is the number of features (i.e., $(\theta, \phi) \in \mathbb{R}^{n+d}$). Altogether, there are at most $2Q \cdot t \cdot m_{\mathsf{bag}}$ hyper-rectangles. Since the VC-dimension of d-dimensional hyper-rectangles is $2d$, these hyper-rectangles divide the $\mathbb{R}^{n+d}$ space into at most $(2eQtm_{\mathsf{bag}})^{2n+2d}$ regions by Sauer's lemma. Plugging this bound into Massart's lemma yields the desired result. ☐

Corollary 22. *We have*

$$\mathbb{E}\left[\sup_{\theta,\phi} |\mathsf{Gap}_{\mathbb{E}}(\theta, \phi)|\right] \lesssim \frac{m_{\mathsf{bag}}}{m} + \sqrt{\frac{(n+d)\log(Q \cdot t \cdot m_{\mathsf{bag}})}{Q}}.$$

Proof. Note that by symmetry,

$$\begin{aligned}\mathbb{E}\left[\sup_{\theta,\phi} |\mathsf{Gap}_{\mathbb{E}}(\theta, \phi)|\right] &\le \mathbb{E}\left[\sup_{\theta,\phi} \mathsf{Gap}_{\mathbb{E}}(\theta, \phi)\right] + \mathbb{E}\left[\sup_{\theta,\phi}(-\mathsf{Gap}_{\mathbb{E}}(\theta, \phi))\right] \\ &= 2\mathbb{E}\left[\sup_{\theta,\phi} \mathsf{Gap}_{\mathbb{E}}(\theta, \phi)\right].\end{aligned}$$

It suffices to consider $\mathbb{E}\left[\sup_{\theta,\phi} \mathsf{Gap}_{\mathbb{E}}(\theta, \phi)\right]$. The result holds by applying Lemma 21 and Theorem 10. ☐

Next, we consider the upper bound of $\mathsf{Gap}_{\mathbb{V}}$. Similarly, we define $\mathsf{Gap}_{\mathbb{V}}^{(i)}(\theta, \phi) = (\mathsf{Tree}_{2i-1}(\theta, \phi) - \mathsf{Tree}_{2i}(\theta, \phi))^2 - (\mathsf{Tree}'_{2i-1}(\theta, \phi) - \mathsf{Tree}'_{2i}(\theta, \phi))^2$ and the dual function $\mathsf{DualGap}_{\mathbb{V}}^{(\theta,\phi)}(i) = \mathsf{Gap}_{\mathbb{V}}^{(i)}(\theta, \phi)$.

Let $\mathcal{G}_{\mathbb{V}} = \left\{\mathsf{DualGap}_{\mathbb{V}}^{(\theta,\phi)} : (i) \mapsto \mathsf{Gap}_{\mathbb{V}}^{(i)}(\theta, \phi)\right\}$ be the dual class. By Definition 13, the empirical Rademacher complexity of $\mathcal{G}_{\mathbb{V}}$ is

$$\hat{\mathcal{R}}(\mathcal{G}_{\mathbb{V}}) = \mathop{\mathbb{E}}_{\sigma \sim \{1,-1\}^Q} \left[\sup_{(\theta,\phi)} \frac{1}{Q} \sum_{i=1}^{Q/2} \sigma_i \mathsf{Gap}_{\mathbb{V}}^{(i)}(\theta, \phi) \right].$$

Lemma 23. *We have*

$$\hat{\mathcal{R}}(\mathcal{G}_{\mathbb{V}}) \lesssim \sqrt{\frac{(n+d)\log(Q\cdot t\cdot m_{\mathsf{bag}})}{Q}}.$$

The proof of this lemma is identical to that of Lemma 21, since the number of possible values of $\left(\mathsf{Gap}_{\mathbb{V}}^{(i)}(\theta,\phi)\right)_i$ has the same upper bound as the number of $(\mathsf{Gap}_{\mathbb{E}}^{(i)}(\theta,\phi))_i$. The only difference is that we apply Theorem 15 instead of Massart's lemma. By Theorem 14, we have the following corollary.

Corollary 24. *We have*

$$\mathbb{E}\left[\sup_{\theta,\phi}|\mathsf{Gap}_{\mathbb{V}}(\theta,\phi)|\right] \lesssim \frac{m_{\mathsf{bag}}}{m} + \sqrt{\frac{(n+d)\log(Q\cdot t\cdot m_{\mathsf{bag}})}{Q}}.$$

B.4 Step 3: An analysis of the Metropolis-Hastings sampling

In this step, we analyze the stability of the local search sampling. In our framework, we apply the Metropolis-Hastings sampling method to sample a configuration where the distribution is relevant to the expected improvement acquisition function.

Let θ_{inc} and θ'_{inc} denote the incumbent parameter for S and S' for some iteration t, respectively. With a little abuse of notation, we use $\hat{l}_{\mathsf{inc}} = \hat{l}(\theta_{\mathsf{inc}})$ and $\hat{l}'_{\mathsf{inc}} = \hat{l}'(\theta_{\mathsf{inc}})$ to denote their empirical losses. Let μ,σ^2 and $\mu',{\sigma'}^2$ denote the mean and the variance for S and S' on parameter θ. The expected improvement for (μ,σ^2) is thus $\mathsf{EI}(\mu,\sigma) = \mathbb{E}_{u\sim\mathcal{N}(\mu,\sigma^2)}\left[\max\{\hat{l}_{\mathsf{inc}} - u, 0\}\right]$, and we similarly define $\mathsf{EI}'(\mu',\sigma')$ for $(\mu',{\sigma'}^2)$.

Lemma 25. *We have*

$$|\mathsf{EI}(\mu,\sigma) - \mathsf{EI}'(\mu',\sigma')| \le |\hat{l}_{\mathsf{inc}} - \hat{l}'_{\mathsf{inc}}| + |\mu-\mu'| + \frac{1}{2\pi}|\sigma-\sigma'|.$$

Proof. Note that $|\max\{l-u,0\} - \max\{l'-u,0\}| \le |l-l'|$, and $|\max\{l-u,0\} - \max\{l-u',0\}| \le |u-u'|$. Thus, it suffices to consider the case such that $\hat{l}_{\mathsf{inc}} = \hat{l}'_{\mathsf{inc}}$ and $\mu=\mu'$.

Recall that [28] the expected improvement can be computed by

$$\mathsf{EI}(\mu,\sigma) = (\hat{l}_{\mathsf{inc}} - \mu)\cdot\Phi(Z) + \sigma\cdot\varphi(Z), \quad \left(Z = \frac{\hat{l}_{\mathsf{inc}}-\mu}{\sigma}\right),$$

where $\Phi(\cdot)$ and $\varphi(\cdot)$ are the c.d.f. and the p.d.f. of Gaussian distribution $\mathcal{N}(0,1)$. Note that for $\mathcal{N}(0,1)$, we have $0 \le \varphi(\cdot) \le \frac{1}{2\pi}$. Now, fix μ and $\hat{l}_{\mathsf{inc}}$. The derivative of EI w.r.t. σ is

$$\frac{\mathrm{d}\mathsf{EI}}{\mathrm{d}\sigma} = (\hat{l}_{\mathsf{inc}} - \mu)\frac{\mathrm{d}\Phi}{\mathrm{d}\sigma} + \varphi(Z) + \sigma\frac{\mathrm{d}\varphi}{\mathrm{d}\sigma}.$$

Note that

$$\frac{\mathrm{d}Z}{\mathrm{d}\sigma} = -\frac{\hat{l}_{\mathsf{inc}}-\mu}{\sigma^2},\ \frac{\mathrm{d}\varphi}{\mathrm{d}\sigma} = -Z\cdot\varphi(Z)\cdot\frac{\mathrm{d}Z}{\mathrm{d}\sigma},\ \frac{\mathrm{d}\Phi}{\mathrm{d}\sigma} = \varphi(Z)\cdot\frac{\mathrm{d}Z}{\mathrm{d}\sigma}.$$

Thus,

$$\begin{aligned}\left|\frac{\mathrm{d}\mathsf{EI}}{\mathrm{d}\sigma}\right| &= \left|Z\sigma\cdot\varphi(Z)\cdot\frac{-Z}{\sigma} + \varphi(Z) + \sigma\cdot(-Z\varphi(Z))\cdot\frac{-Z}{\sigma}\right| \\ &= |\varphi(Z)| \le \frac{1}{2\pi},\end{aligned}$$

which indicates that EI is $\frac{1}{2\pi}$-Lipschitz with respect to σ. Therefore, we have $|\mathsf{EI}(\mu,\sigma) - \mathsf{EI}(\mu,\sigma')| \le \frac{|\sigma-\sigma'|}{2\pi}$. □

Now, we are ready to prove a stability bound for EI. The mean is estimated by

$$\mu = \frac{1}{m}\sum_{i=1}^{m} \mathsf{RF}_{\mathbb{E}}(\theta, \phi_i),\ \mu' = \frac{1}{m}\sum_{i=1}^{m} \mathsf{RF}'_{\mathbb{E}}(\theta, \phi'_i).$$

Since S and S' differs by only one instance, we have

$$\begin{aligned}\mathbb{E}\left[\sup_{\theta} |\mu - \mu'|\right] &\leq \frac{1}{m} + \left|\frac{1}{m}\sum_{i=1}^{m-1} \mathsf{Gap}_{\mathbb{E}}(\theta, \phi_i)\right| \\ &\lesssim \frac{m_{\mathsf{bag}}}{m} + \hat{\mathcal{R}}(\mathcal{G}_{\mathbb{E}}) \\ &\lesssim \frac{m_{\mathsf{bag}}}{m} + \sqrt{\frac{(n+d)\log(Q \cdot t \cdot m_{\mathsf{bag}})}{Q}}.\end{aligned}$$

In practice, the variance is usually estimated by a sum of the mean of the variance and the variance of the mean, i.e.,

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} \mathsf{RF}_{\mathbb{V}}(\theta, \phi_i) + \frac{1}{m(m-1)} \sum_{1 \leq i < j \leq m} (\mathsf{RF}_{\mathbb{E}}(\theta, \phi_i) - \mathsf{RF}_{\mathbb{E}}(\theta, \phi_j))^2,$$

and similarly for ${\sigma'}^2$. We have

$$\begin{aligned}\mathbb{E}\left[\sup_{\theta} |\sigma^2 - {\sigma'}^2|\right] &\lesssim \frac{1}{m} + \left|\frac{1}{m}\sum_{i=1}^{m-1} \mathsf{Gap}_{\mathbb{V}}(\theta, \phi_i)\right| \\ &\quad + \left|\frac{1}{m(m-1)} \sum_{1 \leq i < j \leq m-1} (\mathsf{Gap}_{\mathbb{E}}(\theta, \phi_i) + \mathsf{Gap}_{\mathbb{E}}(\theta, \phi_j))\right| \\ &\lesssim \frac{m_{\mathsf{bag}}}{m} + \hat{\mathcal{R}}(\mathcal{G}_{\mathbb{V}}) + \hat{\mathcal{R}}(\mathcal{G}_{\mathbb{E}}) \\ &\lesssim \frac{m_{\mathsf{bag}}}{m} + \sqrt{\frac{(n+d)\log(Q \cdot t \cdot m_{\mathsf{bag}})}{Q}}.\end{aligned}$$

Moreover, by an identical argument to Lemma 19, we have $|\hat{l}_{\mathsf{inc}} - \hat{l}'_{\mathsf{inc}}| \leq \frac{1}{m}$. Applying Lemma 25 yields

$$\begin{aligned}|\mathsf{EI}(\mu, \sigma) - \mathsf{EI}'(\mu', \sigma')| &\leq |\hat{l}_{\mathsf{inc}} - \hat{l}'_{\mathsf{inc}}| + |\mu - \mu'| + \frac{1}{2\pi}|\sigma - \sigma'| \\ &\leq |\hat{l}_{\mathsf{inc}} - \hat{l}'_{\mathsf{inc}}| + |\mu - \mu'| + \frac{|\sigma^2 - {\sigma'}^2|}{4\pi \cdot \sigma_{\min}}.\end{aligned}$$

After obtaining a stability bound for EI, we analyze the convergence of the local search heuristic. The local search method can be regarded as a Metropolis-Hastings sampling process to approximate a distribution Π where the density $\pi(\cdot) \propto \exp(\mathsf{EI}(\mu(\cdot), \sigma(\cdot))/\tau)$, and the proposal distribution Q with density $q(\cdot|\theta)$ is the Gaussian distribution $\mathcal{N}(\theta, \Delta \cdot I_n)$. Let $q_i(\cdot)$ denote the density of the sampling variable after i iterations.

Lemma 26. *Let T_{MH} denote the number of iterations in the sampling. We have*

$$\left|\frac{q_{T_{\mathsf{MH}}}(x)}{\pi(x)} - 1\right| \leq (\exp(1/\tau) - 1)\left(1 - \frac{1}{\sqrt{(2\pi\Delta)^n}}\exp\left(-\frac{n}{2\Delta} - \frac{1}{\tau}\right)\right)^{T_{\mathsf{MH}}},$$

for any $x \in [0,1]^n$.

Proof. Note that the expected improvement satisfies $0 \leq \mathsf{EI}(\cdot,\cdot) \leq 1$. Thus, $\mathsf{EI}(\cdot,\cdot)/\tau \in [0, 1/\tau]$. This indicates that the target density $\pi(\cdot) \in [\exp(-1/\tau), \exp(1/\tau)]$. The initial sample is uniformly sampled over the parameter space, which means $q_0(\cdot) = 1$.

The proposal distribution is $Q = \mathcal{N}(\theta, \Delta \cdot I_n)$. The density is

$$
\begin{aligned}
q(x) &= \frac{1}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{1}{2\Delta}\|x-\theta\|_2^2)\right) \\
&\geq \frac{1}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{n}{2\Delta}\right) \\
&\geq \frac{1}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{n}{2\Delta} - \frac{1}{\tau}\right) \cdot \pi(x).
\end{aligned}
$$

Therefore, we can apply Theorem 17 to obtain the desired convergence rate. □

B.5 Step 4: Putting things together

Finally, we apply the algorithmic stability framework to prove the main theorem.

Proof of Theorem 5. Recall that $l(\theta) = \mathbb{E}_{I\sim\mathcal{D},\xi\sim\Xi}[u(\theta, I, \xi)]$ denote the loss of parameter θ, and $\Theta = (\theta_1, \ldots, \theta_{T_{\mathsf{total}}})$ and $\Theta' = (\theta'_1, \ldots, \theta'_{T_{\mathsf{total}}})$ denote the iteration paths of the algorithm configurator on dataset S and S'. For simplicity, we use $\tilde{\theta} = \theta_{T_{\mathsf{total}}}$ and $\tilde{\theta}' = \theta'_{T_{\mathsf{total}}}$ to denote the final configuration. Note that $\tilde{\theta}$ and $\tilde{\theta}'$ are stochastic depending on the randomness of the configurator. We use $p(\tilde{\theta} = x)$ and $p(\tilde{\theta}' = x)$ to denote the probability densities of $\tilde{\theta}$ and $\tilde{\theta}'$.

Note that the randomness of the configurator consists of two parts: The bagging randomness of the random forest model, and the randomized sampling of the parameters θ_t. Let RF denote the randomness of the bagging (i.e., we assume we determine the bagging instances of each iteration before running the configurator). To apply the algorithmic stability framework, we need to bound

$$
\mathop{\mathbb{E}}_{\tilde{\theta},\tilde{\theta}'}\left[|l(\tilde{\theta}) - l(\tilde{\theta}')|\right] = \mathop{\mathbb{E}}_{\mathsf{RF}}\left[\mathop{\mathbb{E}}_{\tilde{\theta},\tilde{\theta}'}\left[\left|l(\tilde{\theta}) - l(\tilde{\theta}')\right| \middle| \mathsf{RF}\right]\right].
$$

Now, we fix RF and consider $\mathbb{E}_{\tilde{\theta},\tilde{\theta}'}\left[\left|l(\tilde{\theta}) - l(\tilde{\theta}')\right| \middle| \mathsf{RF}\right]$. For brevity, we use $\mathrm{d}x_{[k]}$ to denote $\mathrm{d}x_1\mathrm{d}x_2\cdots\mathrm{d}x_k$ in the integral. Fixing RF, we have

$$
\begin{aligned}
\mathop{\mathbb{E}}_{\tilde{\theta},\tilde{\theta}'}\left[|l(\tilde{\theta}) - l(\tilde{\theta}')|\right] &= \int_x l(x) \cdot \left|p(\tilde{\theta} = x) - p(\tilde{\theta}' = x)\right| \mathrm{d}x \\
&\leq \int_x \left|p(\tilde{\theta} = x) - p(\tilde{\theta}' = x)\right| \mathrm{d}x \\
&= \int_{x_{T_{\mathsf{total}}}} \left|\int_{x_{[T_{\mathsf{total}}-1]}} p(\Theta = x)\mathrm{d}x_{[T_{\mathsf{total}}-1]} - \int_{x_{[T_{\mathsf{total}}-1]}} p(\Theta' = x)\mathrm{d}x_{[T_{\mathsf{total}}-1]}\right| \mathrm{d}x_{T_{\mathsf{total}}} \\
&\leq \int_{x_{[T_{\mathsf{total}}]}} |p(\Theta = x) - p(\Theta' = x)| \, \mathrm{d}x_{[T_{\mathsf{total}}]} \\
&= 2\mathsf{TV}(\Theta, \Theta') \leq \sqrt{2\mathsf{KL}(\Theta\|\Theta')}.
\end{aligned}
$$

The first inequality is due to $l(\theta) \in [0, 1]$, the second inequality is due to $|\int_x f(x)\mathrm{d}x| \leq \int_x |f(x)|\mathrm{d}x$, and the last inequality is due to Pinsker's inequality (Theorem 8).

Therefore, it suffices to upper-bound the KL divergence between two iteration paths. By definition,

$$
\mathsf{KL}(\Theta\|\Theta') = \mathbb{E}_{x\sim\Theta}\left[\log \frac{p(\Theta = x)}{p(\Theta' = x)}\right].
$$

We aim to bound the ratio between the densities of Θ and Θ'. In the following, for simplicity of notations, we use $p(x), p'(x)$ to denote $p(\Theta = x), p(\Theta' = x)$, and use $p(x_i), p'(x_i)$ to denote

$p(\theta_i = x_i), p(\theta_i' = x_i)$. Note that

$$\begin{aligned}
\frac{p(x)}{p'(x)} &= \frac{p(x_1, \dots, x_{T_{\mathsf{init}}}) \prod_{t=T_{\mathsf{init}}+1}^{T_{\mathsf{total}}} p(x_t | x_1, \dots, x_{t-1})}{p'(x_1, \dots, x_{T_{\mathsf{init}}}) \prod_{t=T_{\mathsf{init}}+1}^{T_{\mathsf{total}}} p'(x_t | x_1, \dots, x_{t-1})} \\
&= \frac{\prod_{t=T_{\mathsf{init}}+1}^{T_{\mathsf{total}}} p(x_t | x_1, \dots, x_{t-1})}{\prod_{t=T_{\mathsf{init}}+1}^{T_{\mathsf{total}}} p'(x_t | x_1, \dots, x_{t-1})} \\
&= \prod_{t=T_{\mathsf{init}}+1}^{T_{\mathsf{total}}} \frac{p(x_t | x_1, \dots, x_{t-1})}{p'(x_t | x_1, \dots, x_{t-1})},
\end{aligned}$$

where the second equality is due to Claim 18. Now, fix an iteration number t, we bound the ratio between $p(x_t | x_1, \dots, x_{t-1})$ and $p'(x_t | x_1, \dots, x_{t-1})$.

Since we condition on the same $x_1, \dots, x_{t-1}$, the previously sampled parameters for S and S' are the same. Let $q_{T_{\mathsf{MH}}}(x_t)$ denote the sampled parameter for x_t using Metropolis-Hastings. Thus, $q_{T_{\mathsf{MH}}}(x_t) = p(x_t | x_1, \dots, x_{t-1})$. Let $\pi(x_t)$ denote the target distribution of Metropolis-Hastings. We similarly define $q'_{T_{\mathsf{MH}}}$ and π' for S'. By definition,

$$\frac{p(x_t | x_1, \dots, x_{t-1})}{p'(x_t | x_1, \dots, x_{t-1})} = \frac{q_{T_{\mathsf{MH}}}(x_t)}{q'_{T_{\mathsf{MH}}}(x_t)} = \frac{\pi(x_t)}{\pi'(x_t)} \cdot \frac{q_{T_{\mathsf{MH}}}(x_t)}{\pi(x_t)} \cdot \frac{\pi'(x_t)}{q_{T_{\mathsf{MH}}}(x_t)}.$$

We bound each term respectively. We have

$$\begin{aligned}
\frac{\pi(x_t)}{\pi'(x_t)} &= \left(\frac{\exp(\mathsf{EI}(\mu(x_t), \sigma(x_t))/\tau)}{\int_x \exp(\mathsf{EI}(\mu(x), \sigma(x))/\tau) \mathrm{d}x} \right) \cdot \left(\frac{\exp(\mathsf{EI}'(\mu'(x_t), \sigma'(x_t))/\tau)}{\int_x \exp(\mathsf{EI}'(\mu'(x), \sigma'(x))/\tau) \mathrm{d}x} \right)^{-1} \\
&= \exp\left(\frac{\mathsf{EI}(\mu(x_t), \sigma(x_t)) - \mathsf{EI}'(\mu'(x_t), \sigma'(x_t))}{\tau} \right) \cdot \frac{\int_x \exp(\mathsf{EI}'(\mu'(x), \sigma'(x))/\tau) \mathrm{d}x}{\int_x \exp(\mathsf{EI}(\mu(x), \sigma(x))/\tau) \mathrm{d}x} \\
&\le \exp\left(\frac{2}{\tau} \cdot \sup_x |\mathsf{EI}(\mu(x), \sigma(x)) - \mathsf{EI}'(\mu'(x), \sigma'(x))| \right).
\end{aligned}$$

Note that $\pi(x_t)$ and $\pi'(x_t)$ are actually random variables depending on the bagging randomness of the random forest. Now, we consider the randomness of RF. We have

$$\begin{aligned}
\mathbb{E}_{\mathsf{RF}}\left[\log \frac{\pi(x_t)}{\pi'(x_t)} \right] &\le \mathbb{E}_{\mathsf{RF}}\left[\frac{2}{\tau} \cdot \sup_x |\mathsf{EI}(\mu(x), \sigma(x)) - \mathsf{EI}'(\mu'(x), \sigma'(x))| \right] \\
&\le \mathbb{E}_{\mathsf{RF}}\left[\frac{2}{\tau} \cdot \sup_x \left(|\hat{l}_{\mathsf{inc}} - \hat{l}'_{\mathsf{inc}}| + |\mu - \mu'| + \frac{1}{4\pi\sigma_{\min}} |\sigma^2 - {\sigma'}^2| \right) \right] \\
&\lesssim \frac{1}{\tau\sigma_{\min}} \left(\frac{m_{\mathsf{bag}}}{m} + \sqrt{\frac{(n+d)\log(Qtm_{\mathsf{bag}})}{Q}} \right).
\end{aligned}$$

Moreover, by Lemma 26, we have

$$\begin{aligned}
\log \frac{q_{T_{\mathsf{MH}}}(x_t)}{\pi(x_t)} &\le \log \left(1 + \left(\exp\left(\frac{1}{\tau}\right) - 1 \right) \left(1 - \frac{1}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{n}{2\Delta} - \frac{1}{\tau} \right) \right)^{T_{\mathsf{MH}}} \right) \\
&\le \left(\exp\left(\frac{1}{\tau}\right) - 1 \right) \left(1 - \frac{1}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{n}{2\Delta} - \frac{1}{\tau} \right) \right)^{T_{\mathsf{MH}}} \\
&\lesssim \left(\exp\left(\frac{1}{\tau}\right) - 1 \right) \exp\left(-\frac{T_{\mathsf{MH}}}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{n}{2\Delta} - \frac{1}{\tau} \right) \right),
\end{aligned}$$

where the second inequality is due to $\log(1+x) \leq x$, and the last one is due to $(1-x)^{1/x} \approx 1/e$ for small enough $x > 0$. We also have

$$\log \frac{\pi'(x_t)}{q'_{T_{\mathsf{MH}}}(x_t)} \leq \log \left(1 + 2\left(\exp\left(\frac{1}{\tau}\right) - 1\right)\left(1 - \frac{1}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{n}{2\Delta} - \frac{1}{\tau}\right)\right)^{T_{\mathsf{MH}}}\right)$$

$$\leq 2\left(\exp\left(\frac{1}{\tau}\right) - 1\right)\left(1 - \frac{1}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{n}{2\Delta} - \frac{1}{\tau}\right)\right)^{T_{\mathsf{MH}}}$$

$$\lesssim \left(\exp\left(\frac{1}{\tau}\right) - 1\right) \exp\left(-\frac{T_{\mathsf{MH}}}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{n}{2\Delta} - \frac{1}{\tau}\right)\right)$$

for sufficiently large T_{MH}, since $\frac{1}{1-x} \leq 1 + 2x$ for $x \in [0, 0.5]$.

Therefore,

$$\mathsf{KL}(\Theta \| \Theta') = \mathbb{E}_x \left[\sum_{t=T_{\mathsf{init}}+1}^{T_{\mathsf{total}}} \log \frac{p(x_t | x_1, \ldots, x_{t-1})}{p'(x_t | x_1, \ldots, x_{t-1})}\right]$$

$$\leq \sum_{t=T_{\mathsf{init}}+1}^{T_{\mathsf{total}}} \sup_{x_t} \left(\log \frac{\pi(x_t)}{\pi'(x_t)} + \log \frac{q_{T_{\mathsf{MH}}}(x_t)}{\pi(x_t)} + \log \frac{\pi'(x_t)}{q_{T_{\mathsf{MH}}}(x_t)}\right).$$

Plugging this bound into $\mathbb{E}[|l(\tilde{\theta}) - l(\tilde{\theta}')|] \leq \sqrt{2\mathsf{KL}(\Theta\|\Theta')}$ yields

$$\mathop{\mathbb{E}}_{\tilde{\theta}, \tilde{\theta}'} \left[|l(\tilde{\theta}) - l(\tilde{\theta}')|\right] \lesssim \sqrt{T_{\mathsf{iter}} \left(\varepsilon_{\mathsf{Stab}} + \varepsilon_{\mathsf{MH}}\right)},$$

where

$$\varepsilon_{\mathsf{Stab}} = \frac{1}{\tau \sigma_{\min}} \left(\frac{m_{\mathsf{bag}}}{m} + \sqrt{\frac{(n+d)\log(Q T_{\mathsf{total}} m_{\mathsf{bag}})}{Q}}\right),$$

and

$$\varepsilon_{\mathsf{MH}} = \left(\exp\frac{1}{\tau} - 1\right) \exp\left(-\frac{T_{\mathsf{MH}}}{\sqrt{(2\pi\Delta)^n}} \exp\left(-\frac{n}{2\Delta} - \frac{1}{\tau}\right)\right).$$

Applying Theorem 3 yields the desired result. □

Algorithm 2 An algorithm to compute the empirical Rademacher complexity.

Input: The random forest predictor $\mathsf{Tree}_1, \ldots, \mathsf{Tree}_Q$.
Output: The empirical Rademacher complexity $\hat{\mathcal{G}}_{\mathsf{RF}}$ and $\hat{\mathcal{G}}^{\mathbb{V}}_{\mathsf{RF}}$.

Let $\hat{\mathcal{G}}_{\mathsf{RF}}, \hat{\mathcal{G}}^{\mathbb{V}}_{\mathsf{RF}} \leftarrow 0$.
for $k = 1, \ldots, M$ **do**
  Randomly sample the Rademacher variables $\sigma_1, \ldots, \sigma_Q \sim \{-1, 1\}^Q$.
  Let $\mathsf{MAX}_{\mathbb{E}}, \mathsf{MAX}_{\mathbb{V}} \leftarrow 0$.
  for (θ, ϕ) in each piece of $(\mathsf{Tree}_i(\cdot, \cdot))_i$ (by calling Algorithm 3) **do**
    Let $\mathsf{MAX}_{\mathbb{E}} \leftarrow \max\{\mathsf{MAX}_{\mathbb{E}}, \frac{1}{Q}\sum_{i=1}^{Q} \sigma_i \mathsf{Tree}_i(\theta, \phi)\}$.
    Let $\mathsf{MAX}_{\mathbb{V}} \leftarrow \max\{\mathsf{MAX}_{\mathbb{V}}, \frac{1}{Q}\sum_{i=1}^{Q/2} \sigma_i (\mathsf{Tree}_{2i-1}(\theta, \phi) - \mathsf{Tree}_{2i}(\theta, \phi))^2\}$.
  end for
  Let $\hat{\mathcal{G}}_{\mathsf{RF}} \leftarrow \hat{\mathcal{G}}_{\mathsf{RF}} + \mathsf{MAX}_{\mathbb{E}}$, and $\hat{\mathcal{G}}^{\mathbb{V}}_{\mathsf{RF}} \leftarrow \hat{\mathcal{G}}^{\mathbb{V}}_{\mathsf{RF}} + \mathsf{MAX}_{\mathbb{V}}$.
end for
Let $\hat{\mathcal{G}}_{\mathsf{RF}} \leftarrow \hat{\mathcal{G}}_{\mathsf{RF}}/M$ and $\hat{\mathcal{G}}^{\mathbb{V}}_{\mathsf{RF}} \leftarrow \hat{\mathcal{G}}^{\mathbb{V}}_{\mathsf{RF}}/M$.
return $\hat{\mathcal{G}}_{\mathsf{RF}}$ and $\hat{\mathcal{G}}^{\mathbb{V}}_{\mathsf{RF}}$.

C Numerical Experiments

C.1 Data-dependent empirical Rademacher complexity

In our generalization bound, the $\tilde{O}(\sqrt{(n+d)/Q})$ term usually dominates the generalization error, since it converges slower as the number of decision trees Q grows, compared with the number of samples m. As shown in Appendix B.3, this term comes from the Rademacher complexities $\hat{\mathcal{R}}(\mathcal{G}_{\mathbb{E}})$ and $\hat{\mathcal{R}}(\mathcal{G}_{\mathbb{V}})$. In fact, we can derive tighter data-dependent bounds by computing the empirical Rademacher complexity.

Notice that

$$\begin{aligned}\hat{\mathcal{R}}(\mathcal{G}_{\mathbb{E}}) &= \mathop{\mathbb{E}}_{\sigma\sim\{-1,1\}^Q}\left[\sup_{\theta,\phi}\frac{1}{Q}\sum_{i=1}^{Q}\sigma_i \mathsf{Gap}_{\mathbb{E}}^{(i)}(\theta,\phi)\right] \\ &= \mathop{\mathbb{E}}_{\sigma\sim\{-1,1\}^Q}\left[\sup_{\theta,\phi}\frac{1}{Q}\sum_{i=1}^{Q}\sigma_i\left(\mathsf{Tree}_i(\theta,\phi)-\mathsf{Tree}'_i(\theta,\phi)\right)\right] \\ &\le \mathop{\mathbb{E}}_{\sigma\sim\{-1,1\}^Q}\left[\sup_{\theta,\phi}\frac{1}{Q}\sum_{i=1}^{Q}\sigma_i \mathsf{Tree}_i(\theta,\phi)\right] + \mathop{\mathbb{E}}_{\sigma\sim\{-1,1\}^Q}\left[\sup_{\theta,\phi}\frac{1}{Q}\sum_{i=1}^{Q}\sigma_i \mathsf{Tree}'_i(\theta,\phi)\right].\end{aligned}$$

Note that the training samples for Tree and Tree' are identical (though not independent). Thus, it suffices to compute the Rademacher complexity of the decision tree class

$$\hat{\mathcal{R}}_{\mathsf{RF}} = \mathbb{E}_\sigma\left[\sup_{\theta,\phi}\frac{1}{Q}\sum_{i=1}^{Q}\sigma_i \mathsf{Tree}_i(\theta,\phi)\right],$$

given the problem instances for the algorithm configurator. Similarly, for $\hat{\mathcal{R}}(\mathcal{G}_{\mathbb{V}})$, it suffices to compute the empirical Rademacher complexity

$$\hat{\mathcal{R}}_{\mathsf{RF}}^{\mathbb{V}} = \mathbb{E}_\sigma\left[\sup_{\theta,\phi}\frac{1}{Q}\sum_{i=1}^{Q/2}\sigma_i(\mathsf{Tree}_{2i-1}(\theta,\phi)-\mathsf{Tree}_{2i}(\theta,\phi))^2\right].$$

Since the decision tree predictor is a piecewise-constant function, it is easy to compute the values of $\hat{\mathcal{G}}_{\mathsf{RF}}$ and $\hat{\mathcal{G}}_{\mathsf{RF}}^{\mathbb{V}}$. We can randomly sample the values of Rademacher variables σ and compute the supremum by enumerating all pieces (Algorithm 2). By Hoeffding inequality, Algorithm 2 returns $\hat{\mathcal{G}}_{\mathsf{RF}}$ and $\hat{\mathcal{G}}_{\mathsf{RF}}^{\mathbb{V}}$ with an additive error of at most $O(\sqrt{\log(1/\delta)/M})$ with probability at least $1-\delta$. The key problem is to compute the piecewise-constant structure of $(\mathsf{Tree}_i(\cdot,\cdot))_i$ for the random forest model in Line 5 of Algorithm 2.

To enumerate the piecewise structure of the random forest model, we propose a divide-and-conquer algorithm (Algorithm 3). Suppose the total number of pieces of $(\mathsf{Tree}_i(\cdot,\cdot))_i$ is P. It is easy to note that Algorithm 3 finds all pieces in $O(P\cdot(n+d))$ time.

C.2 Experimental results

In this section, we present numerical results for our generalization bound using the empirical Rademacher complexity by Algorithm 2. Experiments are conducted on a 2.0GHz Intel CPU with 4 cores.

AC settings. We perform experiments on the algorithm configuration of the SCIP integer programming (IP) solver. SCIP is a classic open-source IP solver with many tunable parameters[4]. For simplicity, we select four continuous parameters that most significantly influence the performance of the solver. These parameters directly control the key components of the solver, including branching, conflict analysis, cut generation, and presolving. All these four parameters lie in $[0,1]$. See Table 2 for concrete descriptions. We tune these parameters with other ones being the default values to minimize the total running time. In the surrogate model, besides the algorithm parameter, we use two

[4] See `https://www.scipopt.org/doc/html/PARAMETERS.php` for a complete list of parameters.

Algorithm 3 A divide-and-conquer algorithm to enumerate the pieces of $(\mathsf{Tree}_i(\cdot,\cdot))_i$.

Input: The random forest predictor $\mathsf{Tree}_1, \ldots, \mathsf{Tree}_Q$.
Output: Each piece (θ, ϕ) of $(\mathsf{Tree}_i(\cdot,\cdot))_i$.

function DIVIDE-AND-CONQUER(Space, $\mathsf{Node} = (\mathsf{Node}_i)_i$)
  if Node_i is a leaf for all $i = 1, \ldots, Q$ **then**
    return $\{\mathsf{Space}\}$.
  end if
  Find i such that Node_i that is not a leaf node.
  Let Left and Right be the left and right children of Node_i, respectively.
  Let Space_1 and Space_2 be the parameter and feature space of Left and Right, respectively.
  Let $\mathsf{Pieces} \leftarrow \emptyset$.
  if $\mathsf{Space} \cap \mathsf{Space}_1 \neq \emptyset$ **then**
    Let $\mathsf{LeftNodes} = (\mathsf{Node}_1, \ldots, \mathsf{Node}_{i-1}, \mathsf{Left}, \mathsf{Node}_{i+1}, \ldots, \mathsf{Node}_Q)$.
    Let $\mathsf{Pieces} \leftarrow \mathsf{Pieces} \cup$ DIVIDE-AND-CONQUER($\mathsf{Space} \cap \mathsf{Space}_1$, $\mathsf{LeftNodes}$).
  end if
  if $\mathsf{Space} \cap \mathsf{Space}_2 \neq \emptyset$ **then**
    Let $\mathsf{RightNodes} = (\mathsf{Node}_1, \ldots, \mathsf{Node}_{i-1}, \mathsf{Right}, \mathsf{Node}_{i+1}, \ldots, \mathsf{Node}_Q)$.
    Let $\mathsf{Pieces} \leftarrow \mathsf{Pieces} \cup$ DIVIDE-AND-CONQUER($\mathsf{Space} \cap \mathsf{Space}_2$, $\mathsf{RightNodes}$).
  end if
  return Pieces.
end function
Let $\mathsf{Space} \leftarrow [0,1]^{n+d}$ denote the complete parameter and feature space.
Let $\mathsf{Node} \leftarrow (\mathsf{Root}_1, \ldots, \mathsf{Root}_Q)$ where Root_i is the root node of the i-th decision tree.
return DIVIDE-AND-CONQUER(Space, Node).

Table 2: Selected parameters of SCIP in our experiments.

Name	Implication
`branching/scorefac`	The branching score factor to weigh downward and upward gain prediction in the score function.
`conflict/maxvarsfac`	The maximal fraction of variables involved in a conflict constraint.
`separating/maxbounddist`	The maximal relative distance compared to the best node's dual bound for applying separation.
`presolving/abortfac`	Abort presolving, if at most this fraction of the problem was changed in the last round.

values, the number of integer variables and the number of continuous variables in the IP formulation, as the instance features.

We implement the model-based algorithm configurator in Algorithm 1. We set the (hyper-)parameters of the configurator as follows: $T_{\mathsf{init}} = 20$, $T_{\mathsf{iter}} = 10$, $m_{\mathsf{bag}} = 10$, $Q = 1000$, $\sigma_{\min} = 0.1$, $\tau = 0.1$, $T_{\mathsf{MH}} = 10^5$, and $\Delta = 0.03$. We set the maximum time limit of the IP solver to be 60 seconds, and normalize the running time by the time limit so that the performance metric lies in $[0, 1]$. If the solver exceeds the limit, the performance is 1.0. With $Q = 1000$, the data-dependent Rademacher complexity computed by Algorithm 2 is about 10^{-3}, which is rather small and in the same order as m_{bag}/m.

We tune these parameters on two applications of integer programming: VLSI routing and facility location.

VLSI routing [29] is the problem of interconnecting multiple sets of points P_i (called *nets*) on a grid graph. For each net P_i, select a rectilinear Steiner tree topology to connect all points and minimize the total length of all trees. Each Steiner tree should not intersect with topologies of other nets. We randomly synthesize instances of VLSI routing on a 20×20 grid graph with 5 to 10 nets. Each net randomly consists of 2 to 5 points. The length of each grid edge is set to be 1.

Facility location is a classic operations research problem that selects the best locations for a set of facilities. There are n customers and m facilities that have not been built. Customer i can obtain some fraction y_{ij} of the good from facility j with a cost $d_{ij}y_{ij}$. Building a facility j has a cost f_j. We aim to select a subset of facilities to minimize the total cost. We randomly synthesize instances of facility location with $n \in [400, 500]$ customers and $m \in [200, 300]$ facilities. The cost is uniformly sampled from $[0, 10^4]$.

Table 3: Verification of the assumptions in Section 3.1. The performance results give 1-sigma error intervals, which characterize the performances on different instances in the test set.

Configurator	VLSI routing	Facility location
Ours	0.3657 ± 0.1041	0.1295 ± 0.0970
SMAC [2]	0.3629 ± 0.1125	0.1244 ± 0.0954

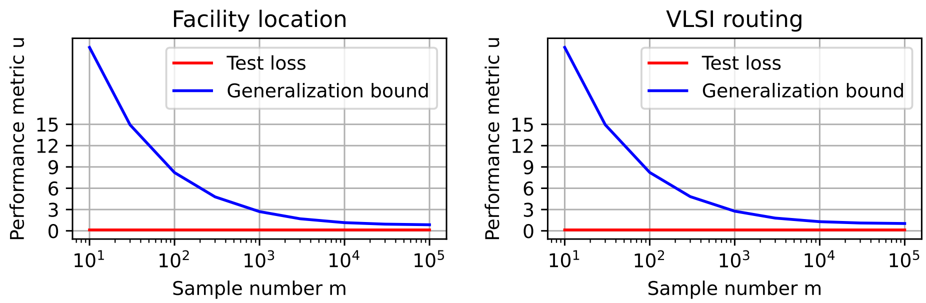

Figure 2: Our generalization bounds for facility location (left) and VLSI routing (right).

Generalization bounds. We plot the generalization bounds in Figure 2, as well as the empirical test loss on the dataset. The data-dependent Rademacher complexity bound is computed using Algorithm 2. We can notice that the generalization error converges to zero and our bound is non-vacuous as the sample number is larger than about 10^4.

Verification of our assumptions. We make some assumptions on the model-based algorithm configurator we considered in Section 3.1. Now, we empirically show that these assumptions are reasonable. They do not noticeably affect the performance of the configurator. We compare our configurator with the classic SMAC [2] configurator. The results are illustrated in Table 3. It can be seen that the effect of our modifications to the configurator is very small. This verifies the rationality of our assumptions.