On Performance Estimation in Automatic Algorithm Configuration

Shengcai Liu,[1] **Ke Tang,**[2*] **Yunwen Lei,**[2] **Xin Yao**[2]
[1]School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China
[2]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
liuscyyf@mail.ustc.edu.cn, {tangk3, leiyw, xiny}@sustech.edu.cn

Abstract

Over the last decade, research on automated parameter tuning, often referred to as automatic algorithm configuration (AAC), has made significant progress. Although the usefulness of such tools has been widely recognized in real world applications, the theoretical foundations of AAC are still very weak. This paper addresses this gap by studying the performance estimation problem in AAC. More specifically, this paper first proves the universal best performance estimator in a practical setting, and then establishes theoretical bounds on the estimation error, i.e., the difference between the training performance and the true performance for a parameter configuration, considering finite and infinite configuration spaces respectively. These findings were verified in extensive experiments conducted on four algorithm configuration scenarios involving different problem domains. Moreover, insights for enhancing existing AAC methods are also identified.

Introduction

Many high-performance algorithms for solving computationally hard problems, ranging from exact methods such as mixed integer programming solvers to heuristic methods such as local search, involve a large number of free parameters that need to be carefully tuned to achieve their best performance. In many cases, finding performance-optimizing parameter settings is performed manually in an ad-hoc way. However, the manually-tuning approach is often intensive in terms of human effort (López-Ibáñez et al. 2016) and thus there are a lot of attempts on automating this process (see (Hutter et al. 2009) for a comprehensive review), which is usually referred to as automatic algorithm configuration (AAC) (Hoos 2012). Many AAC methods such as ParamILS (Hutter et al. 2009), GGA/GGA+(Ansótegui, Sellmann, and Tierney 2009; Ansótegui et al. 2015), irace (López-Ibáñez et al. 2016) and SMAC (Hutter, Hoos, and Leyton-Brown 2011) have been proposed in the last few years. They have been used for boosting the algorithm's performance in a wide range of domains such as the boolean satisfiability problem (SAT) (Hutter et al. 2009), the traveling salesman problem (TSP) (López-Ibáñez et al. 2016; Liu, Tang, and Yao 2019), the answer set programming (ASP) (Hutter et al. 2014) and machine learning (Feurer et al. 2015; Kotthoff et al. 2017).

Despite the notable success achieved in application, the theoretical aspects of AAC have been rarely investigated. To our best knowledge, for AAC the first theoretical analysis was given by Birattari (2004), in which the author analyzed expectations and variances of different performance estimators that estimate the true performance of a given parameter configuration on the basis of N runs of the configuration. It is concluded in (Birattari 2004) that performing one single run on N different problem instances guarantees that the variance of the estimate is minimized, which has served as a guidance in the design of the performance estimation mechanisms in later AAC methods including irace, ParamILS and SMAC. It is noted that the analysis in (Birattari 2004) assumes that infinite problem instances could be sampled for configuration evaluation. However, in practice we are often only given a set of finite training instances (Hoos 2012).

Recently, Kleinberg, Leyton-Brown, and Lucier (2017) introduced a new algorithm configuration framework named Structured Procrastination (SP), which is guaranteed to find an approximately optimal parameter configuration within a logarithmic factor of the optimal runtime in a worst-case sense. Furthermore, the authors showed that the gap between worst-case runtimes of existing methods (ParamILS, GGA, irace, SMAC) and SP could be arbitrarily large. These results were later extended in (Weisz, György, and Szepesvári 2018; 2019), in which the authors proposed new methods, called LEAPSANDBOUNDS (LB) and CapsAndRuns (CR), with better runtime guarantees. However, there is a discrepancy between the algorithm configuration problem addressed by these methods (SP, LB and CR) and the problem that is most frequently encountered in practice. More specifically, these methods are designed to find parameter configurations with approximately optimal performances on the input (training) instances; while in practice it is more desirable to find parameter configurations that will perform well on new unseen instances rather than just the training instances (Hoos 2012). Indeed, one of the most critical issues that needs to be addressed in AAC is the over-tuning phenomenon (Birattari 2004), in which the found parame-

[*]Corresponding author

ter configuration is with excellent training performance, but performs badly on new instances [1].

Based on the above observation, this paper extends the results of (Birattari 2004) in several aspects. First, this paper introduces a new formulation of the algorithm configuration problem (Definition 1), which concerns the optimization of the expected performance of the configured algorithm on an instance distribution $\mathcal{D}$. Compared to the one considered by Birattari (2004) in which $\mathcal{D}$ is directly given (thus could be sampled infinitely), in the problem considered here $\mathcal{D}$ is unknown and inaccessible, and the assumption is that the input training instances (and the test instances) are sampled $i.i.d$ from $\mathcal{D}$. Therefore when solving this configuration problem, we can only use the given finite training instances. One key difficulty is that the true performance of a parameter configuration is unachievable. Subsequently, we could only run a configuration on the training instances to obtain an estimate of its true performance. Thus a natural and important question is that, given finite computational budgets, e.g., N runs of the configuration, how to allocate them over the training instances to obtain the most reliable estimate. Moreover, given that we could obtain an estimate of the true performance, is it possible to quantify the difference between the estimate and the true performance?

The second and the most important contribution of this paper is that it answers the above questions theoretically. More specifically, this paper first introduces a universal best performance estimator (Theorem 1) that always distributes the N runs of a configuration to all training instances as evenly as possible, such that the performance estimate is most reliable. Then this paper investigates the estimation error, i.e., the difference between the training performance (the estimate) and the true performance, and establishes a bound on the estimation error that holds for all configurations in the configuration space, considering the cardinality of the configuration space is finite (Theorem 2). It is shown that the bound deteriorates as the number of the considered configurations increases. Since in practice the cardinality of the configuration space considered could be considerably large or even infinite, by making two mild assumptions on the considered configuration scenarios, we remove the dependence on the cardinality of the configuration space and finally establish a new bound on the estimation error (Theorem 3).

The effectiveness of these results have been verified in extensive experiments conducted on four configuration scenarios involving problem domains including SAT, ASP and TSP. Some potential directions for improving current AAC methods from these results have also been identified.

Algorithm Configuration Problem

In a nutshell, the algorithm configuration problem concerns optimization of the free parameters of a given parameterized algorithm (called target algorithm) for which the performance is optimized.

Let $\mathcal{A}$ denote the target algorithm and let $p_1, ..., p_h$ be parameters of $\mathcal{A}$. Denote the set of possible values for each parameter p_i as Θ_i. A parameter configuration θ (or simply configuration) of $\mathcal{A}$ refers to a complete setting of $p_1, ..., p_h$, such that the behavior of $\mathcal{A}$ on a given problem instance is completely specified (up to randomization of $\mathcal{A}$ itself). The configuration space $\mathbf{\Theta} = \Theta_1 \times \Theta_2 ... \times \Theta_h$ contains all possible configurations of $\mathcal{A}$. For brevity, henceforth we will not distinguish between θ and the instantiation of $\mathcal{A}$ with θ. In real application $\mathcal{A}$ is often randomized and its output is determined by the used configuration θ, the input instance z and the random seed v. Let $\mathcal{D}$ denote a probability distribution over a space $\mathcal{Z}$ of problem instances from which z is sampled. Let $\mathcal{G}$ be a probability distribution over a space $\mathcal{V}$ of random seeds from which v is sampled. In practice $\mathcal{G}$ is often given implicitly through a random number generator.

Given an instance z and a seed v, the quality of θ at (z, v) is measured by a utility function $f_\theta : \mathcal{Z} \times \mathcal{V} \rightarrow [L, U]$, where L, U are bounded real numbers. In practice, it means running θ with v on z, and maps the result of this run to a scalar score. Note how the mapping is done depends on the considered performance metric. For examples, if we are interested in optimizing quality of the solutions found by $\mathcal{A}$, then we might take the (normalized) cost of the solution output by $\mathcal{A}$ as the utility; if we are interested in minimizing computational resources consumed by $\mathcal{A}$ (such as runtime, memory or communication bandwidth), then we might take the quantity of the consumed resource of the run as the utility. No matter which performance metric is considered, in practice the value of f_θ is bounded for all $\theta \in \mathbf{\Theta}$, i.e., for all $\theta \in \mathbf{\Theta}$ and all $(z, v) \in \mathcal{Z} \times \mathcal{V}$, $f_\theta(z, v) \in [L, U]$.

To measure the performance of θ, the expected value of the utility scores of θ across different (z, v), which is the most widely adopted criterion in AAC applications (Hoos 2012), is considered here. More specifically, as presented in Definition 1, the performance of θ, denoted as $u(\theta)$, is its expected utility score over instance distribution $\mathcal{D}$ and random seed distribution $\mathcal{G}$. Without loss of generality, we always assume a smaller value is better for $u(\theta)$. The goal of the algorithm configuration problem is to find a configuration from the configuration space $\mathbf{\Theta}$ with the best performance.

Definition 1 (Algorithm Configuration Problem)**.** *Given a target algorithm $\mathcal{A}$ with configuration space $\mathbf{\Theta}$, an instance distribution $\mathcal{D}$ defined over space $\mathcal{Z}$, a random seed distribution $\mathcal{G}$ defined over space $\mathcal{V}$ and a utility function $f_\theta : \mathcal{Z} \times \mathcal{V} \rightarrow [L, U]$ that measures the quality of θ at (z, v), the algorithm configuration problem is to find a configuration $\theta^\star$ from $\mathbf{\Theta}$ with the best performance:*

$$\theta^\star \in \arg\min_{\theta \in \mathbf{\Theta}} u(\theta),$$

where $u(\theta) = \mathbb{E}_{z\sim\mathcal{D}, v\sim\mathcal{G}}[f_\theta(z, v)]$.

In practice, $\mathcal{D}$ is usually unknown and the analytical solution of $u(\theta)$ is unachievable. Instead, usually we have a set of problem instances $\{z_1, ..., z_K\}$, called training instances, which are assumed to be sampled $i.i.d$ from $\mathcal{D}$. To estimate $u(\theta)$, a series of experiments of θ on $\{z_1, z_2, ..., z_K\}$ could

[1]To appropriately evaluate AAC methods, in the literature, including widely used benchmarks (e.g., AClib (Hutter et al. 2014)) and major contests (e.g., the Configurable SAT Solver Challenge (CSSC) (Hutter et al. 2017)), the common scheme is to use an independent test set that has never been used during the configuration procedures to test the found configurations.

be run. As presented in Definition 2, an experimental setting S_N to estimate $u(\theta)$ is to run θ on $\{z_1, ..., z_K\}$ for N times, each time with a random seed sampled $i.i.d$ from $\mathcal{G}$.

Definition 2 (Experimental Setting S_N). *Given a configuration θ, a set of K training instances $\{z_1, ..., z_K\}$ and the total number N of runs of θ, an experimental setting S_N to estimate $u(\theta)$ is a list of N tuples, in which each tuple (z, v) consists of an instance z and a random seed v, meaning a single run of θ with v on z. Let n_i denote the number of runs performed on z_i (note n_i could be 0, meaning θ will not be run on z_i). It holds that $\sum_{i=1}^{K} n_i = N$ and S_N could be written as:*

$$S_N = [(z_1, v_{1,1}), ..., (z_1, v_{1,n_1}), ..., (z_i, v_{i,1}), ..., (z_i, v_{i,n_i}), ..., (z_K, v_{K,1}), ..., (z_K, v_{K,n_K})].$$

After performing the N runs of θ as specified in S_N, the utility scores of these runs are aggregated to estimate $u(\theta)$. The following estimator $\hat{u}_{S_N}(\theta)$, which calculates the mean utility across all runs and is widely adopted in AAC methods (Hutter et al. 2009; López-Ibáñez et al. 2016; Hutter, Hoos, and Leyton-Brown 2011), is presented in Definition 3.

Definition 3 (Estimator $\hat{u}_{S_N}(\theta)$). *Given a configuration θ and an experimental setting S_N, the training performance of θ, which is an estimate of $u(\theta)$, is given by:*

$$\hat{u}_{S_N}(\theta) = \frac{1}{N} \sum_{i=1}^{K} \sum_{j=1}^{n_i} f_\theta(z_i, v_{i,j}).$$

Since different experimental settings represent different performance estimators, which have different behaviors. It is thus necessary to investigate which S_N is the best.

Universal Best Performance Estimator

To determine the values of $n_1, ..., n_K$ in S_N, Birattari (2004) analyzed expectations and variances of $\hat{u}_{S_N}(\theta)$, and concluded that $\hat{u}_{S_N^\circ}(\theta)$ with $S_N^\circ := [(z_1, v_{1,1}), (z_2, v_{2,1}), ..., (z_N, v_{N,1})]$ has the minimal variance. It is noted that the analysis in (Birattari 2004) assumes that infinite problem instances could be sampled from $\mathcal{D}$; thus for performing N runs of θ, as specified in S_N°, it is always the best to sample N instances from $\mathcal{D}$ and perform one single run of θ on each instance. In other words, S_N° is established on the basis that the number of the training instances K could always be set equal to N. However, in practice usually we only have a finite number of training instances. In the case that $K \neq N$, which S_N is the best? Theorem 1 answers this question for arbitrary relationship between K and N. Before presenting Theorem 1, some necessary definitions are introduced.

Given a configuration θ and an instance z, the expected utility of θ within z, denoted as $u_z(\theta)$, is $\mathbb{E}_{\mathcal{G}}[f_\theta(z, v)|z]$. The variance of the utility of θ within z, denoted as $\sigma_z^2(\theta)$, is $\mathbb{E}_{\mathcal{G}}[(f_\theta(z, v) - u_z(\theta))^2|z]$. Based on $u_z(\theta)$ and $\sigma_z^2(\theta)$, the expected within-instance variance $\bar{\sigma}_{WI}^2(\theta)$ of θ and the across-instance variance $\bar{\sigma}_{AI}^2(\theta)$ of θ are defined in Definition 4 and Definition 5, respectively.

Definition 4 (Expected within-instance Variance of θ). *$\bar{\sigma}_{WI}^2(\theta)$ is the expected value of $\sigma_z^2(\theta)$ over instance distribution $\mathcal{D}$:*

$$\bar{\sigma}_{WI}^2(\theta) = \mathbb{E}_{\mathcal{D}}[\sigma_z^2(\theta)].$$

Definition 5 (Across-instance Variance of θ). *$\bar{\sigma}_{AI}^2(\theta)$ is the variance of $u_z(\theta)$ over instance distribution $\mathcal{D}$:*

$$\bar{\sigma}_{AI}^2(\theta) = \mathbb{E}_{\mathcal{D}}[(u_z(\theta) - u(\theta))^2].$$

The expectation and the variance of an estimator $\hat{u}_{S_N}(\theta)$ are presented in Lemma 1 and Lemma 2, respectively.

Lemma 1. *The expectation of $\hat{u}_{S_N}(\theta)$ is $u(\theta)$, that is, $\hat{u}_{S_N}(\theta)$ is an unbiased estimator of $u(\theta)$ no matter how $n_1, ..., n_K$ in S_N are set:*

$$\mathbb{E}_{S_N}[\hat{u}_{S_N}(\theta)] = u(\theta).$$

Lemma 2. *The variance of $\hat{u}_{S_N}(\theta)$ is given by:*

$$\mathbb{E}_{S_N}[(\hat{u}_{S_N}(\theta) - u(\theta))^2] = \frac{1}{N}\bar{\sigma}_{WI}^2(\theta) + \frac{\sum_{i=1}^{K} n_i^2}{N^2}\bar{\sigma}_{AI}^2(\theta). \tag{1}$$

Theorem 1. *Given a configuration θ, a training set of K instances and the total number N runs of θ, the universal best estimator $\hat{u}_{S_N^*}(\theta)$ for $u(\theta)$ is obtained by setting $S_N^* := n_i \in \{\lfloor \frac{N}{K} \rfloor, \lceil \frac{N}{K} \rceil\}$ for all $i \in \{1, 2, ..., K\}$, s.t. $\sum_{i=1}^{K} n_i = N$. $\hat{u}_{S_N^*}(\theta)$ is an unbiased estimator of $u(\theta)$ and is with the minimal variance among all possible estimators.*

Proof. By Lemma 1, $\hat{u}_{S_N^*}(\theta)$ is an unbiased estimator of $u(\theta)$. We now prove $\hat{u}_{S_N^*}(\theta)$ has the minimal variance. By Lemma 2, the variance of $\hat{u}_{S_N}(\theta)$ is $\frac{1}{N}\bar{\sigma}_{WI}^2(\theta) + \frac{\sum_{i=1}^{K} n_i^2}{N^2}\bar{\sigma}_{AI}^2(\theta)$. Since N and K are fixed, and $\bar{\sigma}_{WI}^2(\theta)$ and $\bar{\sigma}_{AI}^2(\theta)$ are constants for a given θ, we need to minimize $\sum_{i=1}^{K} n_i^2$, s.t. $\sum_{i=1}^{K} n_i = N$. Define $Q_n = \sqrt{\sum_{k=1}^{K} n_i^2}$ and $\bar{n} = \frac{\sum_{i=1}^{K} n_i}{K} = \frac{N}{K}$, it then follows that $Q_n^2 = K\bar{n}^2 + \sum_{i=1}^{K}(n_i - \bar{n}^2)$. Then it suffices to prove that Q_n^2 is minimized on the condition $n_i \in \{\lfloor \frac{N}{K} \rfloor, \lceil \frac{N}{K} \rceil\}$ for all $i \in \{1, 2, ..., K\}$. Assuming Q_n^2 is minimized while the condition not satisfied, then there must exist n_i and n_j, such that $n_i - n_j > 1$; then we have $(n_i - \bar{n})^2 + (n_j - \bar{n})^2 > (n_i - \bar{n})^2 + (n_j - \bar{n})^2 - 2(n_i - n_j) + 2 = (n_i - \bar{n} - 1)^2 + (n_j - \bar{n} + 1)^2$. This contradicts the assumption that Q_n^2 is minimized. The proof is complete. $\square$

Theorem 1 states that it is always the best to distribute the N runs of θ to all training instances as evenly as possible, in which case $\max_{i,j \in \{1,...,K\}} |n_i - n_j| \leq 1$, no matter $K \neq N$ or $K = N$. When $K = N$, $S_N^*(\theta)$ is actually equivalent to S_N° that performs one single run of θ on each instance. When $K \neq N$, $S_N^*(\theta)$ will perform $\lceil \frac{N}{K} \rceil$ runs of θ on each of $(N \bmod K)$ instances and perform $\lfloor \frac{N}{K} \rfloor$ runs on each of the rest instances. It is worth mentoring that practical AAC methods including ParamILS, SMAC and irace actually adopt the same or quite similar estimators as $S_N^*(\theta)$. Theorem 1 provides a theoretical guarantee for these estimators, and $S_N^*(\theta)$ will be further evaluated in the experiments.

Bounds on Estimation Error

Although Theorem 1 presents the estimator with the universal minimal variance, it cannot provide any information about how large the estimation error, i.e., $u(\theta) - \hat{u}_{S_N}(\theta)$, could be. Bounds on estimation error are useful in both theory and practice because we could use them to establish bounds on the true performance $u(\theta)$, given that in algorithm configuration process the training performance $\hat{u}_{S_N}(\theta)$ is actually known. In general, given a configuration θ, its training performance $\hat{u}_{S_N}(\theta)$ is a random variable because the training instances and the random seeds specified in S_N are drawn from distributions $\mathcal{D}$ and $\mathcal{G}$, respectively. Thus we focus on establishing probabilistic inequalities for $u(\theta) - \hat{u}_{S_N}(\theta)$, i.e., for any $0 < \delta < 1$, with probability at least $1 - \delta$, there holds $u(\theta) - \hat{u}_{S_N}(\theta) \leq A(\delta)$. In particular, probabilistic bounds on uniform estimation error, i.e., $\sup_{\theta \in \mathbf{\Theta}}[u(\theta) - \hat{u}_{S_N}(\theta)]$, that hold for all $\theta \in \mathbf{\Theta}$ are established. Recalling that Lemma 1 states $\mathbb{E}_{S_N}[\hat{u}_{S_N}(\theta)] = u(\theta)$, the key technique for deriving bounds on $u(\theta) - \hat{u}_{S_N}(\theta)$ is the concentration inequality presented in Lemma 4 that bounds how $\hat{u}_{S_N}(\theta)$ deviates from its expected value $u(\theta)$.

Lemma 3 (Bernstein's Inequality (Bernstein 1927))**.** *Let $X_1, X_2, ..., X_n$ be independent centered bounded random variables, i.e.,* $\text{Prob}\{|X_i| \leq a\} = 1$ *and* $\mathbb{E}[X_i] = 0$*. Let $\sigma^2 = \frac{1}{n}\sum_{i=1}^n Var[X_i]$ where $Var[X_i]$ is the variance of X_i. Then for any $\epsilon > 0$ we have*

$$\text{Prob}\{\frac{1}{n}\sum_{i=1}^n X_i \geq \epsilon\} \leq \exp(-\frac{n\epsilon^2}{2\sigma^2 + \frac{2a\epsilon}{3}}).$$

Lemma 4. *Given a configuration θ, an experimental setting $S_N = [(z_1, v_{1,1}), ..., (z_K, v_{K,n_K})]$ and a performance estimator $\hat{u}_{S_N}(\theta) = \frac{1}{N}\sum_{i=1}^K \sum_{j=1}^{n_i} f_\theta(z_i, v_{i,j})$. Let $\tau_\theta^2 = \bar{\sigma}_{WI}^2(\theta) + \frac{\sum_{i=1}^K n_i^2}{N}\bar{\sigma}_{AI}^2(\theta)$. Let $C = U - L$, where L, U are the lower bound and the upper bound of f_θ respectively (see Definition 1), and let $n = \max\{n_1, n_2, ..., n_K\}$. Then for any $\epsilon > 0$, we have*

$$\text{Prob}\{u(\theta) - \hat{u}_{S_N}(\theta) \geq \epsilon\} \leq \exp(-\frac{N\epsilon^2}{2\tau_\theta^2 + \frac{2nC\epsilon}{3}}).$$

Proof. Define random variables $x_{i,j} = u(\theta) - f_\theta(z_i, v_{i,j})$, and define random variables $X_i = \sum_{j=1}^{n_i} x_{i,j}$. First we prove that $X_1, ..., X_K$ satisfy the conditions in Lemma 3. $\mathbb{E}[X_i] = \sum_{j=1}^{n_i}\mathbb{E}[x_{i,j}] = \sum_{j=1}^{n_i}[u(\theta) - \mathbb{E}[f_\theta(z_i, v_{i,j})]] = 0$. By Definition 1, $\text{Prob}\{L \leq f_\theta(z_i, v_{i,j}) \leq U\} = 1$, it holds that $L \leq u(\theta) \leq U$ (since $u(\theta) = \mathbb{E}[f_\theta(z_i, v_{i,j})]$). Thus we have $\text{Prob}\{|x_{i,j}| \leq U - L\} = 1$ and $\text{Prob}\{|X_i| \leq n(U - L)\} = 1$. For any $p \neq q$, X_p and X_q are independent. Thus $X_1, X_2, ..., X_K$ are independent random variables.

Let $\bar{X} = \frac{1}{K}\sum_{i=1}^K X_i$. By Lemma 3, it holds that, for any $\epsilon > 0$, $\text{Prob}\{\bar{X} > \epsilon\} \leq \exp(-\frac{K\epsilon^2}{2\sigma^2 + \frac{2C\epsilon}{3}})$, where $\sigma^2 = \frac{1}{K}\sum_{i=1}^K Var[X_i]$. Notice that $\frac{K}{N}\bar{X} = u(\theta) - \hat{u}_{S_N}$; thus it holds that for any $\epsilon > 0$,

$$\text{Prob}\{u(\theta) - \hat{u}_{S_N} > \epsilon\} \leq \exp(-\frac{N\epsilon^2}{\frac{2K}{N}\sigma^2 + \frac{2nC\epsilon}{3}}).$$

It remains to analyze σ^2. Since $E[x_i] = 0$, $Var[X_i] = \mathbb{E}[X_i^2]$. Substitute X_i with $\sum_{j=1}^{n_i} x_{i,j}$ and we have $Var[X_i] = \sum_{j=1}^{n_i}\mathbb{E}[x_{i,j}^2] + \sum_{1 \leq j < l \leq n_i} 2\mathbb{E}[x_{i,j}x_{i,l}]$. $\mathbb{E}[x_{i,j}^2] = Var[x_{i,j}] + \mathbb{E}[x_{i,j}]^2 = Var[x_{i,j}] + 0 = \bar{\sigma}_{WI}^2(\theta) + \bar{\sigma}_{AI}^2(\theta)$ (by setting $N, K = 1$ in Eq. (1)). $\mathbb{E}[x_{i,j}x_{i,l}] = \mathbb{E}[(f_\theta(z_i, v_{i,j}) - u(\theta))(f_\theta(z_i, v_{i,l}) - u(\theta))] = \mathbb{E}[(f_\theta(z_i, v_{i,j})(f_\theta(z_i, v_{i,l})] - u(\theta)^2$. Given an instance z_i, $f_\theta(z_i, v_{i,j})$ and $f_\theta(z_i, v_{i,l})$ are independent. There holds:

$$\begin{aligned}
&\mathbb{E}[(f_\theta(z_i, v_{i,j})(f_\theta(z_i, v_{i,l})] \\
&= \mathbb{E}_{\mathcal{D}}[\mathbb{E}_{\mathcal{G}}[f_\theta(z_i, v_{i,j})f_\theta(z_i, v_{i,l})|z_i]] \\
&= \mathbb{E}_{\mathcal{D}}[\mathbb{E}_{\mathcal{G}}[f_\theta(z_i, v_{i,j})|z_i]\mathbb{E}_{\mathcal{G}}[f_\theta(z_i, v_{i,l})|z_i]] \\
&= \mathbb{E}_{\mathcal{D}}[u_{z_i}(\theta)^2].
\end{aligned}$$

By the fact $\mathbb{E}_{\mathcal{D}}[u_z(\theta)] = u(\theta)$, $\mathbb{E}_{\mathcal{D}}[u_{z_i}(\theta)^2] - u(\theta)^2 = \mathbb{E}_{\mathcal{D}}[(u_{z_i}(\theta) - u(\theta))^2] = \bar{\sigma}_{AI}^2(\theta)$. The last step is by Definition 5. Summing up the above results, we have $Var[X_i] = n_i(\bar{\sigma}_{WI}^2(\theta) + \bar{\sigma}_{AI}^2(\theta)) + n_i(n_i - 1)\bar{\sigma}_{AI}^2(\theta) = n_i\bar{\sigma}_{WI}^2(\theta) + n_i^2\bar{\sigma}_{AI}^2(\theta)$. Thus $\sigma^2 = \frac{1}{K}\sum_{i=1}^K Var[X_i] = \frac{N}{K}\bar{\sigma}_{WI}^2(\theta) + \frac{\sum_{i=1}^K n_i^2}{K}\bar{\sigma}_{AI}^2(\theta)$. The proof is complete. □

On Configuration Space with Finite Cardinality

Theorem 2 presents the uniform bound for estimation error when $\mathbf{\Theta}$ is of finite cardinality.

Theorem 2. *Given a performance estimator $\hat{u}_{S_N}(\theta)$. Let $\theta^\dagger = \arg\max_{\theta \in \mathbf{\Theta}} \tau_\theta^2$, where $\tau_\theta^2 = \bar{\sigma}_{WI}^2(\theta) + \frac{\sum_{i=1}^K n_i^2}{N}\bar{\sigma}_{AI}^2(\theta)$, and let $\tau^2 = \tau_{\theta^\dagger}^2$, $\bar{\sigma}_{WI}^2 = \bar{\sigma}_{WI}^2(\theta^\dagger)$ and $\bar{\sigma}_{AI}^2 = \bar{\sigma}_{AI}^2(\theta^\dagger)$. Let $n = \max\{n_1, n_2, ..., n_K\}$ and $C = U - L$. Given that $\mathbf{\Theta}$ is of finite cardinality, i.e., $\mathbf{\Theta} = \{\theta_1, \theta_2, ..., \theta_m\}$, then for any $0 < \delta < 1$, with probability at least $1 - \delta$, there holds:*

$$\begin{aligned}
&\sup_{\theta \in \mathbf{\Theta}}[u(\theta) - \hat{u}_{S_N}(\theta)] \\
&\leq \frac{2nC\ln\frac{m}{\delta}}{3N} + \sqrt{2\ln\frac{m}{\delta}(\frac{1}{N}\bar{\sigma}_{WI}^2 + \frac{\sum_{i=1}^K n_i^2}{N^2}\bar{\sigma}_{AI}^2)}. \quad (2)
\end{aligned}$$

Proof. By Lemma 4, for a given configuration θ, for any $\epsilon > 0$, it holds that $\text{Prob}\{u(\theta) - \hat{u}_{S_N}(\theta) \geq \epsilon\} \leq \exp(-\frac{N\epsilon^2}{2\tau_\theta^2 + \frac{2nC\epsilon}{3}})$. By union bound, $\text{Prob}\{\sup_{\theta \in \mathbf{\Theta}}[u(\theta) - \hat{u}_{S_N}(\theta)] \geq \epsilon\} \leq \sum_{i=1}^m \text{Prob}\{u(\theta_i) - \hat{u}_{S_N}(\theta_i) \geq \epsilon\} \leq m\exp(-\frac{N\epsilon^2}{2\tau^2 + \frac{2nC\epsilon}{3}})$. Let $\delta = m\exp(-\frac{N\epsilon^2}{2\tau^2 + \frac{2nC\epsilon}{3}})$, and ϵ is solved as: $\epsilon = \frac{1}{2N}[\frac{2nC}{3}\ln\frac{m}{\delta} + \sqrt{(\frac{2nC}{3}\ln\frac{1}{\delta})^2 + 8N\tau^2\ln\frac{m}{\delta}}] \leq \frac{2nC\ln\frac{m}{\delta}}{3N} + \sqrt{2\ln\frac{m}{\delta}(\frac{\tau^2}{N})}$. Substituting τ^2 with $\bar{\sigma}_{WI}^2 + \frac{\sum_{i=1}^K n_i^2}{N}\bar{\sigma}_{AI}^2$ proves Theorem 2. □

Note that for different S_N, the bounds on the right side of Eq. (2) are different. The proof of Theorem 1 shows that $\sum_{i=1}^K n_i^2$, s.t. $\sum_{i=1}^K n_i = N$, is minimized on the condition $n_i \in \{\lfloor\frac{N}{K}\rfloor, \lceil\frac{N}{K}\rceil\}$ for all $i \in \{1, 2, ..., K\}$. Moreover, it is easy to verify that $n = \max\{n_1, n_2, ..., n_K\}$ is also minimized on the same condition, in which case $n = \lceil\frac{N}{K}\rceil$. Thus we can immediately obtain Corollary 1.

Corollary 1. *The estimator $\hat{u}_{S_N^*}$ established in Theorem 1, has the best bound for uniform estimation error in Theorem 2. Given that K divides N, for any $0 < \delta < 1$, with probability at least $1 - \delta$, there holds:*

$$\sup_{\theta \in \mathbf{\Theta}} [u(\theta) - \hat{u}_{S_N^*}(\theta)] \leq \frac{2C \ln \frac{m}{\delta}}{3K} + \sqrt{2 \ln \frac{m}{\delta} (\frac{1}{N} \bar{\sigma}_{WI}^2 + \frac{1}{K} \bar{\sigma}_{AI}^2)}.$$

On Configuration Space with Infinite Cardinality

Since in practice the cardinality of $\mathbf{\Theta}$ could be considerably large (e.g., 10^{12}), in which case the bound provided by Theorem 2 could be very loose. Moreover, when the cardinality of $\mathbf{\Theta}$ is infinite, Theorem 2 does not apply anymore. To address these issues, we establish new uniform error bound without dependence on the cardinality of $\mathbf{\Theta}$ based on two mild assumptions given below.

Assumption 1. *(a) We assume there exists $R > 0$ such that $\mathbf{\Theta} \subseteq B_R$, where $B_R = \{\mathbf{w} \in \mathbb{R}^h : \|\mathbf{w}\|_2 \leq R\}$ is a ball of radius R and $\|\mathbf{w}\|_2 = \sum_{i=1}^h w_i^2$ for $\mathbf{w} = (w_1, \ldots, w_h)$.*

(b) We assume for any $(z, v) \in \mathcal{Z} \times \mathcal{V}$, the utility function f is L-Lipschitz continuous, i.e., $|f_\theta(z, v) - f_{\tilde{\theta}}(z, v)| \leq L||\theta - \tilde{\theta}||_2$ for all $\theta, \tilde{\theta} \in \mathbf{\Theta}$.

Part (a) of Assumption 1 means the ranges of the values of all parameters considered are bounded, which holds in nearly all practical algorithm configuration scenarios (Hutter et al. 2014). Part (b) of Assumption 1 poses limitations on how fast f_θ can change across $\mathbf{\Theta}$. This assumption is also mild in the sense that it is expected that configurations with similar parameter values would result in similar behaviors of $\mathcal{A}$, thus getting similar performances. The key technique for deriving the new bound is *covering numbers* as defined in Definition 6, and the new bound is established in Theorem 3.

Definition 6. *Let $\mathcal{F}$ be a set and d be a metric. For any $\eta > 0$, a set $\mathcal{F}^\triangle \subset \mathcal{F}$ is called an η-cover of $\mathcal{F}$ if for every $f \in \mathcal{F}$ there exists an element $g \in \mathcal{F}^\triangle$ satisfying $d(f, g) \leq \eta$. The covering number $\mathcal{N}(\eta, \mathcal{F}, d)$ is the cardinality of the minimal η-cover of $\mathcal{F}$:*

$$\mathcal{N}(\eta, \mathcal{F}, d) := \min\{|\mathcal{F}^\triangle| : \mathcal{F}^\triangle \textit{ is an } \epsilon\textit{-cover of } \mathcal{F}\}.$$

Lemma 5 presents a covering number bound on B_R.

Lemma 5 ((Gilles 1999))**.**

$$\ln \mathcal{N}(\eta, B_R, d_2) \leq h \ln(3R/\eta),$$

where $d_2(\mathbf{w}, \tilde{\mathbf{w}}) = \|\mathbf{w} - \tilde{\mathbf{w}}\|_2$.

Since $\mathbf{\Theta} \subset B_R$, it is easy to verify that $\ln \mathcal{N}(\eta, \mathbf{\Theta}, d_2) \leq \ln \mathcal{N}(\eta, B_R, d_2)$. Based on the L-Lipschitz continuity assumption, Lemma 6 establishes a bound for $\mathcal{N}(\eta, \mathcal{F}, d_\infty)$, where $\mathcal{F} = \{f_\theta : \theta \in \mathbf{\Theta}\}$.

Lemma 6. *Let $\mathcal{F} = \{f_\theta : \theta \in \mathbf{\Theta}\}$ and $d_\infty(f_\theta, f_{\tilde{\theta}}) = \sup_{(z,v) \in \mathcal{Z} \times \mathcal{V}} |f_\theta(z, v) - f_{\tilde{\theta}}(z, v)|$. If Assumption 1 holds, then $\ln \mathcal{N}(\eta, \mathcal{F}, d_\infty) \leq h \ln(3RL/\eta)$.*

Proof. For any $\theta, \tilde{\theta} \in \mathbf{\Theta}$, by the Lipschitz continuity we know $d_\infty(f_\theta, f_{\tilde{\theta}}) \leq L\|\theta - \tilde{\theta}\|_2$. Then, any (ϵ/L)-cover of B_R w.r.t. d_2 would imply an ϵ-cover of $\mathcal{F}$ w.r.t. d_∞. This together with Lemma 5 implies the stated result. The proof is complete. □

Lemma 7. *For any positive constants k, l, b, c, the inequality $\epsilon^k l + b \ln \epsilon \geq c$ has a solution*

$$\epsilon_0 = \left(\frac{c + b \max(\ln l - \ln c, 0)/k}{l} \right)^{1/k}.$$

Theorem 3. *If Assumption 1 holds and $h \ln(12LR) \geq 1$, then for any $0 < \delta < 1$, with probability $1 - \delta$ there holds:*

$$\sup_{\theta \in \mathbf{\Theta}} [u(\theta) - \hat{u}_{S_N}(\theta)] \leq \sqrt{\frac{[h \ln(12LR) + \ln(\frac{1}{\delta}) + \frac{1}{2} h \ln \frac{N}{\frac{8\tau^2 + 4nC}{3}}](\frac{24\tau^2 + 4nC}{3})}{N}},$$

where $n, C, \tau^2, \bar{\sigma}_{WI}^2, \bar{\sigma}_{AI}^2$ are defined as in Theorem 2.

Proof. Without loss of generality we can assume $\epsilon \leq 1$. Let $\{f_{\theta_1}, ..., f_{\theta_m}\}$ be a $\epsilon/4$-cover of $\mathcal{F}$ with $m = \mathcal{N}(\epsilon/4, \mathcal{F}, d_\infty)$, where $\mathcal{F} = \{f_\theta : \theta \in \mathbf{\Theta}\}$. By Definition 6, for any $f_\theta \in \mathcal{F}$ there exists $f_{\theta_j} \in \{f_{\theta_1}, ..., f_{\theta_m}\}$, such that $d_\infty(f_\theta, f_{\theta_j}) = \sup_{(z,v) \in \mathcal{Z} \times \mathcal{V}} |f_\theta(z, v) - f_{\theta_j}(z, v)| \leq \epsilon/4$; it follows that $|\mathbb{E}[f_\theta(z, v)] - \mathbb{E}[f_{\theta_j}(z, v)]| = |u(\theta) - u(\theta_j)| \leq \epsilon/4$ and $|\hat{u}_{S_N}(\theta) - \hat{u}_{S_N}(\theta_j)| = \frac{1}{N} \sum_{i=1}^K \sum_{j=1}^{n_i} |f_\theta(z_i, v_{i,j}) - f_{\theta_j}(z_i, v_{i,j})| \leq \epsilon/4$. Then, $\sup_{\theta \in \mathbf{\Theta}} \left[u(\theta) - \hat{u}_{S_N}(\theta)\right] \leq \sup_{\theta \in \mathbf{\Theta}} \Big[u(\theta) - u(\theta_j) + u(\theta_j) - \hat{u}_{S_N}(\theta_j) + \hat{u}_{S_N}(\theta_j) - \hat{u}_{S_N}(\theta)\Big] \leq \frac{\epsilon}{2} + \max_{j \in \{1,..,m\}} [u(\theta_j) - \hat{u}_{S_N}(\theta_j)]$. It then follows that $\mathrm{Prob}\{\sup_{\theta \in \mathbf{\Theta}} [u(\theta) - \hat{u}_{S_N}(\theta)] \geq \epsilon\} \leq \mathrm{Prob}\{\max_{j \in \{1,..,m\}} [u(\theta_j) - \hat{u}_{S_N}(\theta_j)] \geq \epsilon/2\} \leq \sum_{j=1}^m \mathrm{Prob}\{[u(\theta_j) - \hat{u}_{S_N}(\theta_j)] \geq \epsilon/2\} \leq m\exp(-\frac{\frac{N}{4}\epsilon^2}{2\tau^2 + \frac{nC\epsilon}{3}})$, where the last inequality is due to Lemma 4. We need to find a ϵ satisfying $\exp(h \ln(12RL/\epsilon) - \frac{N\epsilon^2}{8\tau^2 + \frac{4nC\epsilon}{3}}) \leq \delta$, for which it suffices to find a solution of (by $\epsilon \leq 1$) $\frac{N\epsilon^2}{8\tau^2 + \frac{4nC}{3}} + h \ln \epsilon \geq h \ln(12LR) + \ln(1/\delta)$. This inequality takes the form of the inequality in Lemma 7. We can apply Lemma 7 to show that a solution is (note $h \ln(12LR) \geq 1$)

$$\epsilon = \left(\frac{h \ln(12LR) + \ln(1/\delta) + 2^{-1} h \ln \frac{N}{8\tau^2 + \frac{4nC}{3}}}{\frac{N}{8\tau^2 + \frac{4nC}{3}}} \right)^{\frac{1}{2}}.$$

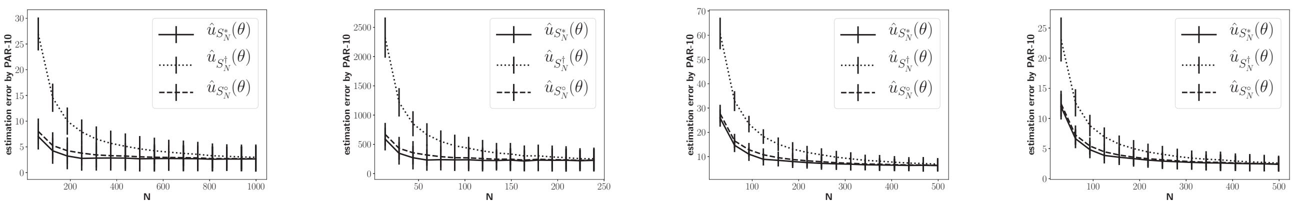
□

Discussion

There are some important findings from the above results. First, both Theorem 2 and Theorem 3 relate the bounds on $u(\theta) - \hat{u}_{S_N}(\theta)$ with the complexity of $\mathbf{\Theta}$, and the bounds deteriorate as the complexity increases. This means as the considered configuration space gets more complex, there is a

Table 1: Summary of the configuration scenarios and gathered performance matrix in each scenario. h is the #parameters of the target algorithm. Tmax is the cutoff time. The $portgen$ generator (Johnson and McGeoch 2007) was used to generate the TSP instances (in which the cities are randomly distributed). For each scenario, $\mathbf{\Theta}_M$ was composed of the default parameter configuration and $M-1$ random configurations.

Scenario	Algorithm	Domain	Bechmark	M	P	Tmax
SATenstein-QCP	SATenstein (KhudaBukhsh et al. 2016), h = 54	SAT	Randomly selected from QCP (Gomes and Selman 1997)	500	500	5s
clasp-weighted-sequence	clasp (Gebser et al. 2007), h=98	ASP	"small" type weighted-sequence (Lierler et al. 2012)	500	120	25s
LKH-uniform-400	LKH (Helsgaun 2000), h=23	TSP	Generated by $portgen$ (Johnson and McGeoch 2007), #city=400	500	250	10s
LKH-uniform-1000	LKH (Helsgaun 2000), h=23	TSP	Generated by $portgen$ (Johnson and McGeoch 2007), #city=1000	500	250	10s



(a) SATenstein-QCP (b) clasp-weighted-sequence (c) LKH-uniform-400 (d) LKH-uniform-1000

Figure 1: Estimation error for different estimators in different scenarios at $r_1 = 0.5$.

possibility that the estimation error could be larger. Second, as expected, as N and K get larger, the estimation error gets smaller, and $\hat{u}_{S_N}(\theta)$ will converge to $u(\theta)$ with probability 1 with $N \to \infty$ and $K \to \infty$. Third, Corollary 1 shows that, for the estimator $\hat{u}_{S_N}(\theta^*)$ which are widely used in current AAC methods, the gain on error reduction decreases rapidly as N and K get larger (which are also shown in Figure 2 in the experiments), and the effects of increasing N and K also depend on $\bar{\sigma}^2_{WI}$ and $\bar{\sigma}^2_{AI}$, two quantities varying across different algorithm configuration scenarios. Thus for enhancing current AAC methods, instead of fixing N as a large number (e.g., SMAC sets N to 2000 by default) and using as many training instances as possible, it is more desirable to use different N and K according to the configuration scenario considered, in which case N and K may be adjusted dynamically in the configuration process as more data are gathered to estimate $\bar{\sigma}^2_{WI}$ and $\bar{\sigma}^2_{AI}$.

Experiments

In this section, we present our experimental studies. First we introduce our experiment setup. Then, we verify our theoretical results in two facets: 1) comparison of different performance estimators; 2) the effects of different values of m (the number of considered configurations), N (the number of runs of θ to estimate $u(\theta)$) and K (the number of training instances) on the estimation error.

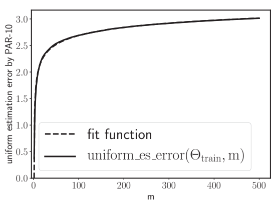
We conducted experiments based on a re-sampling approach (Birattari 2004), which is often used for time-consuming empirical analysis. Specifically, we considered 4 different scenarios. We selected two scenarios SATenstein-QCP and clasp-weighted-sequence from the Algorithm Configuration Library (AClib) (Hutter et al. 2014) and built two new scenarios LKH-uniform-400/1000. For each scenario, we gathered a $M \times P \times 5$ matrix containing the performances of M configurations on P instances, with each configuration running on each instance for 5 times. Let $\mathbf{\Theta}_M$ be the set of the M configurations and $\mathcal{Z}_P$ be the set of the P instances. In the experiments, when acquiring the performance of a configuration θ on an instance, instead of actually running θ, the value stored in the corresponding entry of the matrix was used. The details of the scenarios and the performance matrices are summarized in Table 1. In the experiments the optimization goal considered is the runtime needed to solve the problem instances (for SAT and ASP) or to find the optima of the problem instances (for TSP). In particular, the performance metric was set to Penalized Average Runtime–10 (PAR-10) (Hutter et al. 2009), which counts a timeout as 10 times the given cutoff time.

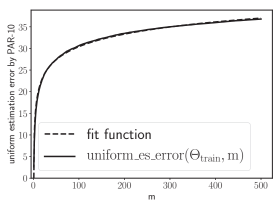
For convenience henceforth we will use "$\text{split}_{P_1|P_2}$" to denote that we subsequently randomly select, without replacement, P_1 and P_2 instances from $\mathcal{Z}_P$ as training instances and test instances respectively. For a given θ, we always used the performance obtained by an estimator on the training instances as its training performance, and used its performance on the test instances as its true performance. We use $\text{uniform\_es\_error}(\Theta)$ to denote the maximal estimation error across the configurations in $\mathbf{\Theta}$.

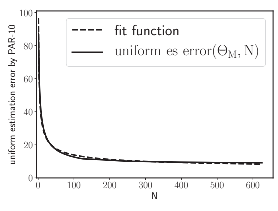
All the experiments were conducted on a Xeon machines with 128 GB RAM and 24 cores each (2.20 GHz, 30 MB Cache), running CentOS. The code was implemented based on AClib (Hutter et al. 2014) [2].

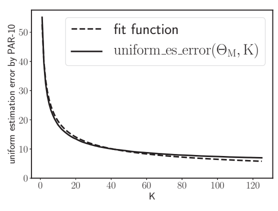
Comparison of Different Estimators. We compared $\hat{u}_{S_N^*}(\theta)$ with two estimators $\hat{u}_{S_N^\dagger}(\theta)$ and $\hat{u}_{S_N^\circ}(\theta)$. For evaluating θ, $\hat{u}_{S_N^\dagger}(\theta)$ repeatedly randomly selects an instance from $\mathcal{Z}_P$ without replacement, and runs θ for 5 times on the instance as long as the total number of runs of θ not exceeding N. $\hat{u}_{S_N^\dagger}(\theta)$ is greedier than $\hat{u}_{S_N^*}(\theta)$ in the sense that it ensures the estimated performance of θ on the used instances is as accurate as possible. Another estimator $\hat{u}_{S_N^\circ}(\theta)$ is the one presented in (Birattari 2004), which repeatedly

[2]The code and the complete experiment results are available at https://github.com/EEAAC/ac_estimation_error



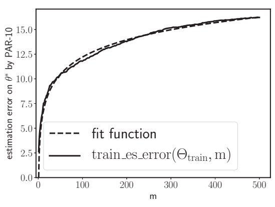


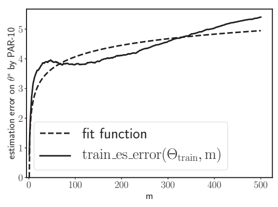


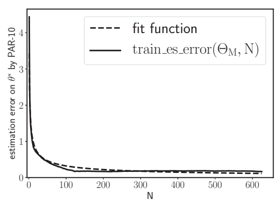


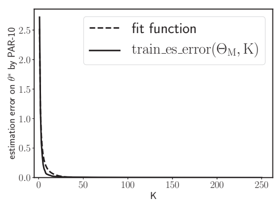
(a) SATenstein-QCP (b) clasp-weighted-sequence (c) LKH-uniform-400 (d) LKH-uniform-1000

Figure 2: Uniform estimation error at different m, N and K and the fit functions based on the theoretical results.









(a) clasp-weighted-sequence (b) LKH-uniform-1000 (c) LKH-uniform-400 (d) SATenstein-QCP

Figure 3: Estimation error on θ^* at different m, N and K and the fit functions based on the theoretical results.

randomly selects an instance from $\mathcal{Z}_P$ with replacement, and runs θ for a single time on the instance. $\hat{u}_{S_N^\circ}(\theta)$ has more randomness than $\hat{u}_{S_N^*}(\theta)$ since it does not ensure that N runs of θ are distributed evenly on all instances. We set $K = r_1 P$ and $N = r_2 K$, and ranged r_1 from 0.1 to 0.5 with a step of 0.05, r_2 from 0.25 to 4.0 with a step of 0.25. To reduce the variations of our experiments, for each combination of r_1 and r_2, we $\text{split}_{K|P/2}$ for 2500 times, and on each split, we obtained the estimation error of an estimator on each $\theta \in \mathbf{\Theta}$, and then calculated the mean value, which was further averaged over all splits. That is, for each combination of r_1 and r_2, we obtained a mean estimation error for each estimator. Due to space limitations, we only present the results in terms of error bars (mean $\pm$ std) at $r_1 = 0.5$ in Figure 1. The results at other values of r_1 are similar. Figure 1 is in line with Theorem 1. Overall $\hat{u}_{S_N^*}(\theta)$ is the best estimator among the three, and its performance advantage is remarkable when N is small. When N gets larger, it is expected, and as shown in Figure 1, that the estimation error for all three estimators will converge to 0. The fact that $\hat{u}_{S_N^*}(\theta)$ is better than $\hat{u}_{S_N^\circ}(\theta)$ indicates that it is necessary to distribute N runs of θ as evenly as possible over all instances.

Estimation Error at Different m, N and K. We always fixed two values while ranging the other one. We ranged m from 1 to M, while setting $K = P/2$ and $N = 5K$. We ranged N from 1 to $5K$ while setting $K = P/2$ and $m = M$. We ranged K from 1 to $P/2$ while setting $N = 5K$ and $m = M$. For a given m, we $\text{split}_{K|P/2}$ for 2500 times, and on each split, we started with an empty set $\mathbf{\Theta}_{train}$ of configurations and then repeatedly expanded $\mathbf{\Theta}_{train}$ by adding a configuration randomly selected from $\mathbf{\Theta}_M \setminus \mathbf{\Theta}_{train}$. Each time a new configuration θ was added to $\mathbf{\Theta}_{train}$, $\text{uniform\_es\_error}(\Theta_{\text{train}})$ was recorded, which was further averaged over all 2500 splits. That is, for each m, we obtained a mean value of $\text{uniform\_es\_error}(\Theta_{\text{train}})$, denoted as $\text{uniform\_es\_error}(\Theta_{\text{train}}, \text{m})$. Similarly, for a given N or a given K, we always $\text{split}_{K|P/2}$ for 2500 times, and on each split, we obtained $\text{uniform\_es\_error}(\Theta_{\text{M}})$, and then averaged it over all splits. Thus for each considered N and K, we obtained $\text{uniform\_es\_error}(\Theta_{\text{M}}, \text{N})$ and $\text{uniform\_es\_error}(\Theta_{\text{M}}, \text{K})$, respectively. Due to space limitations, we only present parts of the results in Figure 2 and other results are very similar. To verify whether our analysis (Theorem 2) correctly captures the dependence of estimation error on N, M and K, we also plot the function $f(m) = a \ln m + b\sqrt{\ln m}$ for m, $f(N) = a + b\sqrt{1/N}$ for N and $f(K) = a/K + b\sqrt{1/K}$ for K in Figure 2, where the parameters a, b are computed by fitting f with the data collected in the experiments, i.e., $\{m \mapsto \text{uniform\_es\_error}(\Theta_{\text{train}}, \text{m}) : m \in \{1, ..., M\}\}$, $\{N \mapsto \text{uniform\_es\_error}(\Theta_{\text{M}}, \text{N}) : N \in \{1, ..., \frac{5}{2}P\}\}$ and $\{K \mapsto \text{uniform\_es\_error}(\Theta_{\text{M}}, \text{K}) : K \in \{1, ..., \frac{1}{2}P\}\}$, respectively. Overall Figure 2 demonstrates that our analysis managed to capture the dependence of uniform estimation error on m, N and K. It is worth noting that in the experiments the effects of increasing m, N and K depend on $\bar{\sigma}^2_{WI}$ and $\bar{\sigma}^2_{AI}$, which vary across configuration scenarios. Moreover, the estimation error becomes very low and quite stable when N approaches $K/2$, which means running θ on half of the training instances could already obtain a reliable estimate of $u(\theta)$.

It is also meaningful to investigate whether our analysis could reflect how the estimation error on θ^*, i.e., the configuration with the best training performance in $\mathbf{\Theta}$, denoted as $\text{train\_es\_error}(\Theta)$, would change. We conducted the same experiments as described above to gather $\text{train\_es\_error}(\Theta_{\text{train}}, \text{m})$, $\text{train\_es\_error}(\Theta_{\text{M}}, \text{N})$ and $\text{train\_es\_error}(\Theta_{\text{M}}, \text{K})$. Figure 3 plots the results and the fit functions. It could be seen that although the bounds are not directly established for $\mathrm{train\_es\_error}(\Theta)$, the findings also apply to it to a considerable extent.

Conclusion

The main results of this paper include the universal best performance estimator and bounds on the uniform estimation error, which were verified in extensive experiments. Possible future directions include data-dependent bounds that are tighter and computable from realization of training instances and analysis of the notorious over-tuning phenomenon based on the results in this paper.

Acknowledgments

This work is supported partially by the National Key Research and Development Program of China (Grant No. 2017YFC0804003), the National Natural Science Foundation of China (Grant Nos. 61672478, 61806090), the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams(Grant No. 2017ZT07X386) and the Shenzhen Peacock Plan (Grant No. KQTD2016112514355531).

References

Ansótegui, C.; Malitsky, Y.; Samulowitz, H.; Sellmann, M.; and Tierney, K. 2015. Model-Based Genetic Algorithms for Algorithm Configuration. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI'2015*, 733–739. Buenos Aires, Argentina: AAAI Press.

Ansótegui, C.; Sellmann, M.; and Tierney, K. 2009. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming, CP'2009*, 142–157. Lisbon, Portugal: Springer.

Bernstein, S. 1927. *Theory of probability*. Moscow.

Birattari, M. 2004. On the Estimation of the Expected Performance of a Metaheuristic on a Class of Instances. Technical report, Technical Report TR/IRIDIA/2004-01, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J. T.; Blum, M.; and Hutter, F. 2015. Efficient and Robust Automated Machine Learning. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems, NIPS'2015*, 2962–2970. Montreal, Quebec, Canada: Curran Associates, Inc.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. *clasp* : A Conflict-Driven Answer Set Solver. In *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR'2007*, 260–265. Tempe, AZ: Springer.

Gilles, P. 1999. *The Volume of Convex Bodies and Banach Space Geometry*, volume 94. Cambridge University Press.

Gomes, C. P., and Selman, B. 1997. Problem Structure in the Presence of Perturbations. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI'97*, 221–226. Rhode Island: AAAI Press / The MIT Press.

Helsgaun, K. 2000. An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research* 126(1):106–130.

Hoos, H. H. 2012. Automated Algorithm Configuration and Parameter Tuning. In Hamadi, Y.; Monfroy, E.; and Saubion, F., eds., *Autonomous Search*. Springer. 37–71.

Hutter, F.; Hoos, H. H.; Leyton-Brown, K.; and Stützle, T. 2009. ParamILS: An Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research* 36(1):267–306.

Hutter, F.; López-Ibáñez, M.; Fawcett, C.; Lindauer, M. T.; Hoos, H. H.; Leyton-Brown, K.; and Stützle, T. 2014. AClib: A Benchmark Library for Algorithm Configuration. In *Proceedings of the 8th International Conference on Learning and Intelligent Optimization, LION'2014*, 36–40. Gainesville, FL: Springer.

Hutter, F.; Lindauer, M.; Balint, A.; Bayless, S.; Hoos, H. H.; and Leyton-Brown, K. 2017. The Configurable SAT Solver Challenge (CSSC). *Artificial Intelligence* 243:1–25.

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization, LION'2011*, 507–523. Rome, Italy: Springer.

Johnson, D. S., and McGeoch, L. A. 2007. Experimental Analysis of Heuristics for the STSP. In Gutin, G., and Punnen, A. P., eds., *The Traveling Salesman Problem and Its Variations*. Springer. 369–443.

KhudaBukhsh, A. R.; Xu, L.; Hoos, H. H.; and Leyton-Brown, K. 2016. SATenstein: Automatically Building Local Search SAT Solvers from Components. *Artificial Intelligence* 232:20–42.

Kleinberg, R.; Leyton-Brown, K.; and Lucier, B. 2017. Efficiency Through Procrastination: Approximately Optimal Algorithm Configuration with Runtime Guarantees. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'2017*, 2023–2031. Melbourne, Australia: ijcai.org.

Kotthoff, L.; Thornton, C.; Hoos, H. H.; Hutter, F.; and Leyton-Brown, K. 2017. Auto-WEKA 2.0: Automatic Model Selection and Hyperparameter Optimization in WEKA. *Journal of Machine Learning Research* 18:25:1–25:5.

Lierler, Y.; Smith, S.; Truszczynski, M.; and Westlund, A. 2012. Weighted-Sequence Problem: ASP vs CASP and Declarative vs Problem-Oriented Solving. In *Proceedings of 14th International Symposium on Practical Aspects of Declarative Languages, PADL'2012*, 63–77. Philadelphia, PA: Springer.

Liu, S.; Tang, K.; and Yao, X. 2019. Automatic Construction of Parallel Portfolios via Explicit Instance Grouping. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI' 2019*, 1560–1567. Honolulu, Hawaii: AAAI Press.

López-Ibáñez, M.; Dubois-Lacoste, J.; Pérez Cáceres, L.; Stützle, T.; and Birattari, M. 2016. The irace Package: Iterated Racing for Automatic Algorithm Configuration. *Operations Research Perspectives* 3:43–58.

Weisz, G.; György, A.; and Szepesvári, C. 2018. LEAPSANDBOUNDS: A Method for Approximately Optimal Algorithm Configuration. In *Proceedings of the 35th International Conference on Machine Learning, ICML'2018*, 5254–5262. Stockholmsmässan, Stockholm, Sweden: PMLR.

Weisz, G.; György, A.; and Szepesvári, C. 2019. CapsAndRuns: An Improved Method for Approximately Optimal Algorithm Configuration. In *Proceedings of the 36th International Conference on Machine Learning, ICML'2019*, 6707–6715. Long Beach, California: PMLR.