Learning Configurations for Data-Driven Multi-Objective Optimization

Zhiyang Chen [1] **Hailong Yao** [2 3] **Xia Yin** [1]

Abstract

Multi-objective optimization problems arise widely in various fields. In practice, multi-objective optimization is generally solved by heuristics with tunable parameters that are highly application-specific. Tuning parameters based on real-world instances (a.k.a. algorithm configuration) are generally empirical without theoretical guarantees. In this work, we establish the theoretical foundation of data-driven multi-objective optimization through the lens of machine learning theory. We provide generalization guarantees on selecting parameters for multi-objective optimization algorithms based on sampled problem instances. Moreover, if the performance metric of the algorithm is the Pareto volume, we can PAC-learn the approximately optimal configuration in polynomial time. We apply our framework to various algorithms, including approximation algorithms, local search, and linear programming. Experiments on multiple problems verify our theoretical findings.

1. Introduction

Multi-objective optimization problems arise widely in various fields, including operations research (Herzel et al., 2021), machine learning (Sener & Koltun, 2018), and design automation (Chen & Young, 2020). A multi-objective optimization problem may involve multiple contradictory objectives that require simultaneous optimization. Therefore, a best-of-all-world solution that is optimal for every objective may not exist. Algorithm designers instead seek a trade-off between multiple objectives.

A notion of optimality for multi-objective optimization is the so-called Pareto optimality. However, computing the exact set of Pareto-optimal solutions is generally intractable for most multi-objective optimization problems, since the optimal Pareto set may contain an exponential number of solutions. In practice, heuristic algorithms are often applied to tackle multi-objective optimization. For example, approximation algorithms (Ravi & Goemans, 1996; Chen & Young, 2020) are designed to simultaneously approximate multiple objectives. Another widely used heuristic is the weighted sum method, e.g., turning a two-objective optimization problem $\max_x(c_1(x), c_2(x))$ into a single-objective one by optimizing a convex combination of objectives,

$$\max_x (1-\rho)\cdot c_1(x) + \rho \cdot c_2(x),$$

where $0 \le \rho \le 1$.

However, both approximation algorithms and the weighted sum method contain tunable parameters. An approximation algorithm (Ravi & Goemans, 1996; Chen & Young, 2020) may contain parameters to balance the approximation ratio of different objectives. The weighted sum method also requires setting the weight ρ. The user may select a parameter configuration to optimize the algorithmic performance, or even select multiple configurations to form a solution portfolio to provide more decision options. The selection of parameters significantly impacts the algorithmic performance. There is usually no universal best configuration, and the user needs to tune the parameters based on application-specific problem instances carefully. Many algorithm configuration methods (Schede et al., 2022) are proposed to automatically select parameters with good performances. Researchers even apply neural networks (Li et al., 2021) to predict good parameters for multi-objective optimization.

Although multi-objective optimization algorithms are widely used in practice, the theoretical foundation of parameter selection is limited. In this work, we study the problem of algorithm configuration for multi-objective optimization through the lens of machine learning theory. Recently, there has been a growing interest in the sample complexity of learning algorithm parameters for a given application domain, a.k.a., *data-driven algorithm design* (Balcan, 2020). Concretely, there is a probabilistic distribution over instances of a multi-objective optimization problem. Given a training set sampled from this distribution, the learning algorithm selects one or multiple parameters that optimize the algorithmic performance from a parameterized family

[1]Tsinghua University, China [2]University of Science and Technology Beijing, China [3]Key Laboratory of Advanced Materials and Devices for Post-Moore Chips, Ministry of Education of China. Correspondence to: Hailong Yao <hailongyao@ustb.edu.cn>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

using empirical risk minimization. We aim to provide *generalization guarantees* (a.k.a., *sample complexity* bounds) that bound the number of required samples to guarantee the performance difference to be small enough between training samples and the real distribution.

1.1. Main results

We summarize the contributions of this work as follows.

The first theoretical framework for data-driven multi-objective optimization. We establish a general framework for data-driven multi-objective optimization algorithms. Previous work (Gupta & Roughgarden, 2017; Balcan et al., 2018a; 2021a; Sakaue & Oki, 2022) on data-driven algorithm design shows that many parameterized algorithms, especially combinatorial algorithms, exhibit a piecewise structural property. The same structure also arises in multi-objective optimization algorithms. We present a general theorem that bounds the sample complexity of parameterized multi-objective optimization algorithms with respect to the complexity of the piecewise structure.

We also discuss how to efficiently learn the parameter from training samples (a.k.a., empirical risk minimization): If the performance metric function is the Pareto (hyper-)volume, we can $\left(1-\frac{1}{e}\right)$-approximately PAC-learn the optimal parameter in polynomial time by exploiting the submodular property of the Pareto volume. We further show that the approximation ratio is tight assuming $\mathsf{P} \neq \mathsf{NP}$.

Sample complexity bounds of learning configurations for various algorithms. We apply our general framework to various multi-objective optimization algorithms, including approximation algorithms (shallow-light Steiner trees and bi-criterion minimum spanning trees), local search (vanilla local search and simulated annealing), and linear programming (general LPs and Markov decision processes).

For local search and linear programs, we find that a technical challenge is that the complexities of the piecewise structures may be exponentially large in the worst case, leading to polynomial sample complexity with respect to the problem size. We can overcome this barrier by conducting a beyond-worst-case analysis. The main idea is to use *smoothed analysis*. If the problem instance distribution contains a small noise, we can reduce the number of pieces in performance metric functions to polynomial size, thus leading to logarithmic sample complexity bounds.

Table 1 gives an overview of our sample complexity results. As an extension, we also discuss the application of machine learning models in predicting parameters in Appendix D.

Empirical verifications of theoretical bounds. We verify our theoretical findings with experimental results. We show that the empirical risk minimization algorithm on shallow-light trees indeed improves the Pareto volume metric of multi-objective optimization. We also compute the number of pieces in the piecewise structure on two algorithms, local search for VLSI circuit partitioning and a resource-gathering Markov decision process. We find that, in practice, the intrinsic complexity of the performance metric function is indeed small, thus leading to small generalization bounds. This shows that the smoothed analysis setting is reasonable.

Table 1. An overview of sample complexity results for parameterized multi-objective optimization, where k is the number of parameters and ε is the generalization error.

Problem	Worst-case	Smoothed
Shallow-light trees	$\tilde{O}(k \cdot \varepsilon^{-2})$	N/A
Bi-criterion MSTs	$\tilde{O}(k \cdot \varepsilon^{-2})$	N/A
Local search	$\tilde{O}(k \cdot T \cdot \varepsilon^{-2})$	$\tilde{O}(k \cdot \varepsilon^{-2})$
Simulated annealing	$\tilde{O}(k \cdot T \cdot \varepsilon^{-2})$	$\tilde{O}(k \cdot \varepsilon^{-2})$
General linear programs	$\tilde{O}(k \cdot n \cdot \varepsilon^{-2})$	$\tilde{O}(k \cdot \varepsilon^{-2})$
Markov decision processes	$\tilde{O}(k \cdot \lvert S\rvert \cdot \varepsilon^{-2})$	$\tilde{O}(k \cdot \varepsilon^{-2})$

1.2. Organization

The rest of this paper is organized as follows: Section 2 introduces basic notations and the problem formulation of this work. Section 3 presents the general framework of sample complexity analysis for data-driven multi-objective optimization algorithms. In Sections 4, 5, and 6, we apply our framework to approximation algorithms, local search, and linear programs, respectively. Section 7 provides empirical verifications of our results. Finally, we conclude this paper and discuss directions for future work in Section 8.

A comparison with related work is discussed in Appendix A. Preliminaries are introduced in Appendix B. Technical proofs are given in Appendix C. Appendix D discusses some extensions of our framework. Appendix E supplements the detailed experimental settings.

2. Notations and Problem Formulation

Notations. We use $\tilde{O}(f(n)) = f(n) \cdot \operatorname{poly}\log n$ to suppress logarithmic factors. We also use $[n]$ to denote the set $\{1, 2, \ldots, n\}$ and $\mathbb{I}(\cdot)$ to denote the indicator function. Preliminary notations for multi-objective optimization is omitted to Appendix B.1.

Problem formulation. Given a multi-objective optimization problem Π, we use $\mathcal{I}$ to denote the set of possible problem instances that the algorithm takes as input. We assume there is an unknown, domain-specific probabilistic distribution $\mathcal{D}$ over $\mathcal{I}$.

We have an algorithm $\mathcal{A}$ to solve problem Π parameterized by $\rho \in \mathcal{P}$. Suppose the problem Π has d optimization objectives, we use $\mathcal{A} : \mathcal{P} \times \mathcal{I} \to \mathbb{R}_+^d$ to denote the objective

vector of algorithm $\mathcal{A}$'s output on a particular instance $I \in \mathcal{I}$ with parameter $\rho \in \mathcal{P}$.

The aim of algorithm configuration is to select k parameters $P = (\rho_1, \ldots, \rho_k) \in \mathcal{P}^k$ for $\mathcal{A}$ with good performance in expectation over the distribution $\mathcal{D}$. Suppose the outputs of algorithm $\mathcal{A}$ are $\mathcal{A}(\rho_1, I), \ldots, \mathcal{A}(\rho_k, I)$ using parameters $\rho_1, \ldots, \rho_k$ on an instance I. We use $u(P, I) : \mathcal{P}^k \times \mathcal{I} \to [0, H]$ to denote a performance metric for $\mathcal{A}(\rho_1, I), \ldots, \mathcal{A}(\rho_k, I)$, where H is a constant[1] that upper bounds the performance metric. For example, the performance metric can be the Pareto volume Vol (see Definition B.1). We aim to maximize the expected performance of $\mathcal{A}$, i.e., $\max_{P \in \mathcal{P}^k}(\mathbb{E}_{I \sim \mathcal{D}}[u(P, I)])$.

We provide generalization guarantees (a.k.a., sample complexity bounds) for learning the parameters P. Given m training data $I_1, \ldots, I_m \in \mathcal{I}$ that are independently sampled from $\mathcal{D}$, a generalization guarantee bounds the difference between the average performance on training data and the expected performance on $\mathcal{D}$. We use the idea of *uniform convergence* from statistical learning theory. Uniform convergence provides generalization guarantees for any choice of parameters $P \in \mathcal{P}^k$. Therefore, it is independent of the learning algorithm.

Definition 2.1 (Uniform convergence)**.** The parameterized algorithm $\mathcal{A}$ has *generalization error* $\varepsilon(m, \delta)$, if for any distribution $\mathcal{D}$ over $\mathcal{X}$, with probability at least $1 - \delta$ over m i.i.d. samples $I_1, \ldots, I_m \sim \mathcal{D}$, we have $\left| \left(\frac{1}{m} \sum_{i=1}^m u(P, I_i) \right) - \mathbb{E}_{I \sim \mathcal{D}}[u(P, I)] \right| \leq \varepsilon$, for any $P \in \mathcal{P}^k$. The least number of samples $m(\varepsilon, \delta)$ that ensures such a guarantee is called the *sample complexity* of $u(P, I)$ on distribution $\mathcal{D}$.

We are also interested in providing guarantees for the complete learning procedure. Given m training data $I_1, \ldots, I_m$, we use an *empirical risk minimization* algorithm to solve $\max_{P \in \mathcal{P}^k} \frac{1}{m} \sum_{i=1}^m u(P, I_i)$. By uniform convergence bounds, the learned P is *probably approximately correct* (PAC) for the optimal parameter.

Definition 2.2 (PAC-learning)**.** A learning algorithm (α, β, δ)-*PAC-learns* the (approximate) optimal parameter, if given m i.i.d. samples from $\mathcal{D}$, with probability at least $1 - \delta$, the learning algorithm finds k parameters P such that

$$\mathbb{E}_{I \sim \mathcal{D}}[u(P, I)] \geq \alpha \cdot \max_{P^* \in \mathcal{P}^k} \mathbb{E}_{I \sim \mathcal{D}}[u(P^*, I)] - \beta,$$

for some $0 < \alpha \leq 1$ and $\beta > 0$.

[1]This assumption is standard in previous work of data-driven algorithm design (Gupta & Roughgarden, 2017), and is indeed reasonable. Although the performance metric may depend on the problem size, we can always normalize the metric and consider the relative performance, given a fixed problem instance distribution.

We omit basic tools of statistical learning theory for proving sample complexity bounds to Appendix B.2.

3. General Framework for Sample Complexity

3.1. Uniform convergence

In our main results, we consider the simple case that the problem has only 2 objectives, and hence the algorithm generally has only one real parameter. We discuss the extension to more than 2 objectives in Section D.

We study the setting where the performance of the algorithm has a piecewise-constant structure with respect to the parameters. This structure widely appears in many optimization algorithms, including greedy heuristics (Gupta & Roughgarden, 2017), branch-and-bound (Balcan et al., 2018a), dynamic programming (Balcan et al., 2021a), A* search (Sakaue & Oki, 2022), etc. Formally, the piecewise-constant structure is defined as follows.

Definition 3.1 (Piecewise-constant algorithm)**.** For any problem instance I, we can partition the parameter space $\mathcal{P} \subseteq \mathbb{R}$ into t intervals, such that in each interval, the output of the algorithm $\mathcal{A}(\cdot, I)$ is a constant function.

Theorem 3.2. *Given m i.i.d. training samples $I_1, \ldots, I_m$ from the instance distribution $\mathcal{D}$, with probability at least $1 - \delta$, for any set of k parameters $P = (\rho_1, \ldots, \rho_k) \in \mathcal{P}^k$, we have* $\left| \left(\frac{1}{m} \sum_{i=1}^m u(P, I_i) \right) - \mathbb{E}_{I \sim \mathcal{D}}[u(P, I)] \right| = O\left(\sqrt{\frac{1}{m} \left(k \log t + \log \frac{1}{\delta} \right)} \right)$.

Equivalently, this generalization guarantee can be presented using the language of sample complexity: For any $\varepsilon > 0$ and $0 < \delta < 1$, $m = \Omega\left(\frac{1}{\varepsilon^2} (k \log t + \log \delta^{-1}) \right)$ samples are sufficient to ensure that with probability $1 - \delta$, the generalization error is at most ε. This theorem can be proved by classic pseudo-dimension techniques.

However, as we will show in the following sections, the number of pieces t can be exponential with respect to the problem size. This gives a polynomial bound on the sample complexity (or generalization gap). To overcome this barrier, we apply *smoothed analysis*. In practice, worst-case bounds are generally loose. These bounds are seldom tight since random noises appear in real-world instances. Smoothed analysis assumes the instance distribution $\mathcal{D}$ is not arbitrary, but a distribution of smoothed instances.

Concretely, to sample an instance I, we first sample a smoothed instance $\tilde{I}$ from some distribution $\tilde{\mathcal{D}}$. A smoothed instance is an instance with a small random perturbation, i.e., a probabilistic distribution. Then, we obtain instance I by sampling the random perturbation from $\tilde{I}$. We set a smoothness parameter κ to evaluate the randomness of the

perturbation. (As κ goes from 1 to $+\infty$, our analysis interpolates between average-case and worst-case analysis.) In this setting, the number of pieces t is a random variable with respect to $\tilde{I}$. We can also obtain generalization guarantees by upper bounds on $\mathbb{E}_{I\sim\tilde{I}}[t]$ for any smoothed instance $\tilde{I}$. Since t is a random variable, we can no longer use pseudo-dimension. Instead, we consider a smoothed version of Rademacher complexity and apply Massart's lemma.

Theorem 3.3. *Given m i.i.d. instances $I_1, \ldots, I_m$ where $I_i \sim \tilde{I}_i$ and $\tilde{I}_i \sim \tilde{\mathcal{D}}$ from a smoothed instance distribution $\tilde{\mathcal{D}}$ for each $i \in [m]$, with probability at least $1-\delta$, for any set of k parameters $P = (\rho_1, \ldots, \rho_k) \in \mathcal{P}^k$, we have* $\left|\left(\frac{1}{m}\sum_{i=1}^m u(P, I_i)\right) - \mathbb{E}_{\tilde{I}\sim\tilde{\mathcal{D}}}\mathbb{E}_{I\sim\tilde{I}}[u(P,I)]\right| = O\left(\sqrt{\frac{1}{m}\left(k\log\mathbb{E}_{I\sim\tilde{I}}[t] + k\log m + \log\frac{1}{\delta}\right)}\right)$.

3.2. Empirical risk minimization for Pareto volume

We also study the problem of empirical risk minimization to learn the optimal parameters. If the performance metric function u is the Pareto volume Vol (see Appendix B.1), i.e., $u(P, I) = \mathsf{Vol}(\{\mathcal{A}(\rho_1, I), \ldots, \mathcal{A}(\rho_k, I)\})$, we can approximately PAC-learn k parameters $P \in \mathcal{P}^k$ by exploiting the submodular property of the Pareto volume.

Definition 3.4. A set function $f : 2^S \to \mathbb{R}$ is *submodular*, if for any $T_1 \subseteq T_2 \subseteq S$, and any $x \in S \setminus T_2$, we have $f(T_1 \cup \{x\}) - f(T_1) \geq f(T_2 \cup \{x\}) - f(T_2)$.

Lemma 3.5. *Let $S = \{u_1, \ldots, u_n\}$ be a set of objective vectors. The function $\mathsf{Vol} : 2^S \to [0, H]$, which maps a subset $T \subseteq S$ to the Pareto volume of T, is monotone and submodular.*

Therefore, we can use the folklore $\left(1-\frac{1}{e}\right)$-approximation algorithm for submodular maximization to perform empirical risk minimization. Let $I_1, \ldots, I_m$ denote the training samples from instance distribution $\mathcal{D}$. Empirical risk minimization finds k parameters $P = (\rho_1, \ldots, \rho_k)$ to maximize the empirical Pareto volume, i.e.,

$$\max_{P\in\mathcal{P}^k} \frac{1}{m}\sum_{i=1}^m \mathsf{Vol}(\{\mathcal{A}(\rho_1, I_i), \ldots, \mathcal{A}(\rho_k, I_i)\}). \quad (1)$$

Theorem 3.6. *Given m i.i.d. samples from the instance distribution $\mathcal{D}$, and suppose for any instance I, we have an oracle to compute each piece of $\mathcal{A}(\cdot, I)$. There exists a learning algorithm that finds k parameters $P = (\rho_1, \ldots, \rho_k)$ that $\left(1-\frac{1}{e}, \varepsilon, \delta\right)$-PAC-learns the optimal parameter using $m = O(\frac{1}{\varepsilon^2}(k\log t + \log\delta^{-1}))$ samples in $\mathrm{poly}(m, t)$ time.*

Note that a similar result also holds in the smoothed analysis setting by taking an expectation. In practice, the oracle can be implemented by, e.g., performing a binary search on the parameter space. See Appendix E for details.

On the negative side, we show that $(1-\frac{1}{e})$-approximation is the best bound we can achieve assuming $\mathsf{P} \neq \mathsf{NP}$ by making a reduction from maximum k-set-cover.

Theorem 3.7. *The empirical risk minimization problem (1) cannot be $(1-\frac{1}{e}+\varepsilon)$-approximated in $\mathrm{poly}(m, t)$ time for any $\varepsilon > 0$, unless $\mathsf{P} = \mathsf{NP}$.*

4. Applications to Approximation Algorithms

We first apply our framework to approximation algorithms. We consider two approximation heuristics that are of practical values in VLSI design and network optimization.

4.1. Shallow-light Steiner trees

Given a weighted and undirected graph $G = (V, E)$ with a set of terminals $S \subseteq V$ and a root $r \in S$, a *Steiner tree* of G is a sub-tree that inter-connects S with root r. For any sub-graph $G' \subseteq G$, let $d_{G'}(u, v)$ denote the shortest path distance between u and v on G', and $w(G')$ denote the total edge weight of G'.

In the *shallow-light Steiner tree* (SLT) problem (Chen & Young, 2020), the objective is to construct a Steiner tree that simultaneously minimizes the total weight and the path length from the root r to each terminal in S. We say a Steiner tree T is α-shallow, if $d_T(r, s) \leq \alpha \cdot d_G(r, s)$ for each $s \in S$, and is β-light, if $w(T) \leq \beta \cdot w(T^*)$, where T^* is a Steiner tree with the minimum total weight (a.k.a., the minimum Steiner tree). Our goal is to find (α, β)-SLTs minimizing both α and β. An SLT is an interpolation between the minimum Steiner tree and the shortest path tree.

A direct application of SLTs is the construction of timing-driven routing trees for VLSI design. In VLSI design, we consider rectilinear routing on a 2D plane, i.e., SLTs in metric space $(\mathbb{R}^2, \|\cdot\|_1)$. (In fact, our results can be extended to the general case $(\mathbb{R}^d, \|\cdot\|_p)$ for any $d, p \geq 1$.)

Chen & Young (2020) propose a practical heuristic, namely SALT, for constructing rectilinear SLTs. For a parameter $\rho \geq 0$, it obtains a $(1+\rho, 2+\lceil\log\frac{2}{\rho}\rceil)$-SLT. However, this bound is generally loose in practice. It is important to tune the value of ρ to obtain a tighter Pareto curve. We study the problem of tuning the parameter ρ of SALT.

The SALT algorithm is described as follows: The algorithm begins by calling a heuristic (Chu & Wong, 2008) to obtain an approximate Steiner minimum tree T_0. It then performs a depth-first search on T_0. During the search process, the distance $\mathsf{dist}(v)$ from the root to each point v on T_0 is maintained. Whenever a node v is accessed such that $\mathsf{dist}(v) > (1+\rho)\cdot\|r - v\|_1$ (which induces shallowness larger than $1+\rho$), we call v a breakpoint and update $\mathsf{dist}(v) = \|r - v\|_1$. After the search, a shortest path tree is reconstructed for all breakpoints by calling a heuristic by

Córdova & Lee (1994). The algorithm finally post-optimizes the tree by performing some safe refinements.

Let $\mathcal{A}(\rho, I)$ be the output of SALT on an instance I with the parameter ρ, we have the following lemma.

Lemma 4.1. *For any SLT instance I, we can partition $[0, +\infty)$ into at most n^2 intervals, such that in each interval, $\mathcal{A}(\cdot, I)$ is constant.*

This lemma can be proved by counting the values of ρ that affect the breakpoints. This gives a sample complexity bound by Theorem 3.2.

Corollary 4.2. *Let $u(\cdot, I)$ denote any performance metric function for SLT instance I. The sample complexity of u for SALT is $\tilde{O}\left(\frac{k}{\varepsilon^2}\right)$.*

4.2. Bi-criterion minimum spanning trees

Given an undirected graph $G = (V, E)$ with each edge $e \in E$ associated with two positive weights $w_1(e)$ and $w_2(e)$, we consider the problem of *bi-criterion minimum spanning trees*, i.e., constructing spanning trees T such that the total weights $(W_1(T), W_2(T)) = (\sum_{e\in T} w_1(e), \sum_{e\in T} w_2(e))$ are both minimized. Ravi & Goemans (1996) consider the budget-constrained version of this problem, and propose a $(1, 2)$-approximation algorithm. Concretely, given a budget constraint $\bar{W}_2$, the algorithm considers the following problem,

$$\begin{aligned} \min_{x_e} \ & \sum_{e\in E} w_1(e) \cdot x_e, \\ \text{s.t} \ & \sum_{e\in E} w_2(e) \cdot x_e \le \bar{W}_2, \\ & x_e \in \{0, 1\}, \ \forall e \in E, \\ & \{e \,|\, x_e = 1, e \in E\} \text{ forms a spanning tree.} \end{aligned}$$

Let W_1^* be the optimal weight of this problem. The Ravi-Goemans algorithm finds a spanning tree with total weights (W_1, W_2) such that $W_1 \le W_1^*$ and $W_2 \le 2\bar{W}_2$ in almost-linear time. Note that Ravi-Goemans cannot find a solution T that exactly satisfies the constraint $W_2(T) \le \bar{W}_2$. To balance $W_1(T)$ and $W_2(T)$, we need to carefully tune the value of $\bar{W}_2$. In the following, we consider the problem of learning parameters $\bar{W}_2$ for Ravi-Goemans.

The algorithm can be described as follows: Prune all edges with $w_2(e) > \bar{W}_2$. Let

$$L(z) = \min_{\text{spanning tree } x} \sum_{e\in E} (w_1(e) + w_2(e) \cdot z) x_e - \bar{W}_2 \cdot z$$

denote the Lagrangian function of the problem. The algorithm solves the Lagrangian relaxation $\max_{z\ge 0} L(z)$. Let z^* denote the optimal solution. The algorithm finds the spanning tree T with the least $\sum_{e\in T} w_2(e)$ such that $\sum_{e\in T} w_2(e) \ge \bar{W}_2$. See Figure 1 for an illustration.

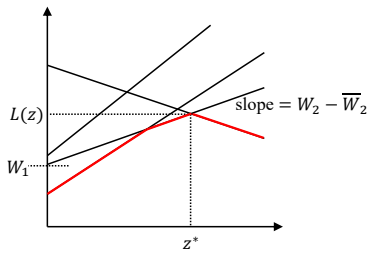

Figure 1. An illustration of Goemans-Ravi algorithm.

Let $\mathcal{A}(\rho, I)$ denote the output of Goemans-Ravi on an instance I with $\bar{W}_2$ equal to ρ, we have the following lemma.

Lemma 4.3. *For any instance I, $\mathcal{A}(\cdot, I)$ is a piecewise-constant function with at most $O(|V|^2)$ pieces.*

Corollary 4.4. *Let $u(\cdot, I)$ denote any performance metric function for bicriterion MST instance I. The sample complexity of u for Goemans-Ravi is $\tilde{O}\left(\frac{k}{\varepsilon^2}\right)$.*

5. Applications to Local Search

5.1. Vanilla local search

In this section, we consider multi-objective optimization using local search algorithms. Local search, as well as its variant, simulated annealing, is widely used in various combinatorial optimization problems. If the problem has multiple objectives, the most natural and popular method is to optimize a weighted sum of the objectives. Such examples include Sechen & Sangiovanni-Vincentelli (1986), Chen et al. (2008), Yu et al. (2022), etc.

We define the following general model for local search: Let $\mathcal{S}$ denote the states of the search algorithm. (Note that $|\mathcal{S}|$ can be very large, e.g., exponential in the problem size.) Each state $S \in \mathcal{S}$ has a set of neighbors $N(S) \subseteq \mathcal{S} \setminus \{S\}$. We use n to denote the maximum number of neighbors for all states S. Each state S is also associated with two costs $c_1(S), c_2(S)$. Without loss of generality, we assume $c_1(S), c_2(S) \in [0, 1]$. We use a weighted sum method to find a state $\hat{S}$ such that $c(\hat{S}) = (1 - \rho) \cdot c_1(\hat{S}) + \rho \cdot c_2(\hat{S})$ is minimized.

We consider the standard local search algorithm by Johnson et al. (1988). Initially, the local search algorithm starts from an arbitrary but fixed state $S_0 \in \mathcal{S}$. For each iteration $t \in [T]$, the algorithm enumerates the neighbors of the current state S, and finds $S' \in N(S)$ with the minimal $c(S')$. If $c(S') \le c(S)$, let $S \leftarrow S'$. Otherwise, the algorithm stops. Notice that a local search algorithm may converge after an exponential number of iterations. Thus, we define T, an upper bound of the iteration number. If the algorithm does not converge, we directly output the state S after iteration T, which is a technique generally applied in practice.

Let $\mathcal{D}$ be a distribution of problem instances, $\mathcal{A}(\rho, I)$ denote the output of the local search algorithm using parameter ρ on instance I, and $u(P, I)$ denote the performance metric. We have the following lemma.

Lemma 5.1. *For any instance I, we can partition $[0, 1]$ into at most $(n + 1)^T$ intervals, such that $\mathcal{A}(\cdot, I)$ is constant in each interval. Moreover, there exists an instance such that the number of pieces is at least $n^{\Omega(T)}$.*

By Theorem 3.2, this gives a sample complexity bound of $\tilde{O}\left(\frac{kT}{\varepsilon^2}\right)$. We cannot further improve this bound using Theorem 3.2 since Lemma 5.1 also gives an asymptotically tight lower bound.

However, we can overcome this barrier using smoothed analysis. The philosophy of smoothed analysis is that the instances in practice always contain some randomness so that the worst case almost never exists.

We say a real random variable is κ-*bounded*, if its probability density function is upper bounded by κ. For example, a uniform distribution over an interval of length $1/\kappa$ is κ-bounded. We call an instance κ-*smoothed*, if the costs $c_1(S)$ and $c_2(S)$ are sampled individually and independently from κ-bounded distributions on $[0, 1]$ for each $S \in \mathcal{S}$. Therefore, a κ-smoothed instance is an instance distribution such that the costs are perturbed with a small random noise.

Using smoothed analysis, we can reduce the number of pieces to polynomial size, leading to an $\tilde{O}\left(\frac{k}{\varepsilon^2}\right)$ sample complexity. The key proof technique is that, due to the random noise, the discontinuities produced in each iteration are $(\mathrm{poly}(T, n) \cdot \kappa)$-bounded. Thus, if the length of a piece is small, in the next iteration, it will not produce new pieces in expectation as much as in the worst case. By an induction on the iteration number, we show a polynomial upper bound on the expected number of pieces.

Theorem 5.2. *Let $\tilde{\mathcal{D}}$ be a distribution over κ-smoothed local search instances. The sample complexity of u on $\tilde{\mathcal{D}}$ is $\tilde{O}\left(\frac{k}{\varepsilon^2}\right)$.*

5.2. Simulated annealing

Simulated annealing (SA) is a variant of the local search method, which has gained great success in many practical problems. We extend our sample complexity results for multi-objective local search to simulated annealing.

We consider the following simulated annealing algorithm:

1. Initialize the state $S \leftarrow S_0$ for some arbitrary but fixed S_0 and $t \leftarrow 1$. Mark S_0 as visited.
2. Randomly sample a state $S' \in N(S)$, and compute $\Delta = c(S) - c(S')$.
3. If S' is not visited, mark S' as visited and with probability $\min\{\exp(\Delta/\tau_t), 1\}$, set $S \leftarrow S'$ (where $\tau_t > 0$ for any $t \in [T]$ is the temperature of the algorithm).
4. Let $t \leftarrow t + 1$. If $t < T$, goto Step 2.

Note that we assume the algorithm does not access any state twice. We make this assumption to provide enough probabilistic independence to make the smoothed analysis possible. This assumption is not very strong since the problem state is usually high-dimensional in practice, and the algorithm seldom repetitively accesses a state.

Since simulated annealing is a randomized algorithm, we also make the following assumption for simplicity of analysis. We assume the randomness of the algorithm is predetermined as part of the input. Concretely, in Step 2 of the algorithm, for each pair of state S and iteration t, we assume the sample $S' = \mathsf{sample}(S, t) \in N(S)$ is fixed as we vary the value of ρ. In Step 3, the algorithm accepts the new state by checking whether $\mathsf{prob}(t) \leq \min\{\exp(\Delta/\tau_t), 1\}$ where $\mathsf{prob}(t)$ is uniformly sampled over $[0, 1]$ for $t = 1, \ldots, T$ before executing the algorithm. Now, an instance I is in the form of $(\mathsf{inst}, \mathsf{sample}(\cdot, \cdot), \mathsf{prob}(\cdot))$, where inst denotes the local search problem instance, and sample, prob denote the randomness of the algorithm. Then, the distribution is a joint distribution over problem instances and algorithmic randomness. This formulation is reasonable, since in practice, for an instance I in the training set, we cannot obtain the expected performance of the algorithm on I, but rather observe the performance of the algorithm on I for some random seed. Thus, we can regard the random seed coming from the instance distribution.

Let $\mathcal{D}$ be a joint distribution over instances and algorithmic randomness, $\mathcal{A}(\rho, I)$ denote the output of the algorithm using parameter ρ on I, and $u(P, I)$ denote the performance metric. Using a similar analysis to Lemma 5.1, the worst-case sample complexity is also $\tilde{O}\left(\frac{kT}{\varepsilon^2}\right)$, and we can reduce the bound by smoothed analysis.

Theorem 5.3. *The sample complexity of u for κ-smoothed simulated annealing is $\tilde{O}\left(\frac{k}{\varepsilon^2}\right)$.*

6. Applications to Linear Programming

6.1. General LP

Another application of our framework is multi-objective linear programs. Consider the following problem,

$$\max_x c^{\mathsf{T}} \cdot x, \text{ s.t. } Ax \leq b,$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{r \times n}$, $b \in \mathbb{R}^r$, and $c = (1 - \rho) \cdot c_1 + \rho \cdot c_2 \in \mathbb{R}^n$ is the convex combination of two objectives c_1 and c_2. Let $\mathcal{A}(\rho, I)$ denote the output on instance I with parameter ρ. Let $u(P, I)$ denote a metric function on multi-objective LP instance I with $\rho \in P = \{\rho_1, \ldots, \rho_k\}$. We study the sample complexity of tuning P to optimize the metric $\mathbb{E}_I[u(\cdot, I)]$.

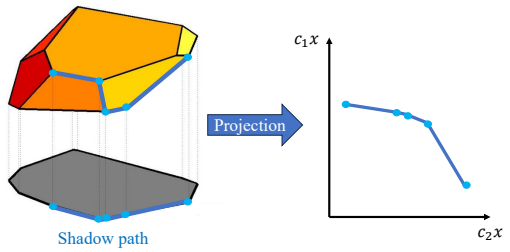

Figure 2. An illustration of shadow path (blue) and its projection to the $(c_1^\mathsf{T}x, c_2^\mathsf{T}x)$ plane.

The sample complexity of $u(\cdot, I)$ for multi-objective LPs is $\tilde{O}\left(\frac{kn}{\varepsilon^2}\right)$. This is obvious since the number of pieces for $\mathcal{A}(\cdot, I)$ is upper bounded by the number of vertices of the polyhedron $Q = \{x \mid Ax \le b\}$, which is $O(r^n)$.

We can also use smoothed analysis to improve the worst-case sample complexity bound. Let $Q[c]$ denote the face of Q maximizing $c^\mathsf{T}x$. We aim to bound the smoothed number of faces $Q[c]$ as ρ varies from zero to one. This is the same as the *shadow path* of Q, which is proposed in previous work studying the smoothed complexity of the Simplex method for LPs. We can hence apply off-the-shelf results of shadow paths to obtain sample complexity bounds.

Definition 6.1 (Shadow path)**.** Let $Q \subseteq \mathbb{R}^n$ be a polyhedron. Given two vectors $c_1, c_2 \in \mathbb{R}^n$, there exists a unique sequence of faces $Q(c_1, c_2) = (v_0, e_1, v_1, \ldots, e_l, v_l)$ of Q, and a sequence $0 = \rho_0 < \rho_1 < \cdots < \rho_l < \rho_{l+1} = 1$ such that (1) for $1 \le i \le l$, $e_i = Q[(1-\rho_i)c_1 + \rho_i c_2]$; (2) for $0 \le i \le l$ and $\rho \in (\rho_i, \rho_{i+1})$, $v_i = Q[(1-\rho)c_1 + \rho c_2]$; (3) for $1 \le i \le l-1$, $v_i = e_i \cap e_{i+1}$. $Q(c_1, c_2)$ is called the *shadow path* of Q with respect to c_1, c_2 (see Figure 2).

Under random perturbations, we have $\dim e_i = 1$ for $1 \le i \le l$ almost surely. Therefore, without loss of generality, the shadow path is indeed a "path" where $\{v_i\}$ are vertices and $\{e_i\}$ are edges.

To illustrate the optimization objectives, we can project the shadow path to the plane by mapping each point $x \in Q(c_1, c_2)$ to $(c_1^\mathsf{T}x, c_2^\mathsf{T}x)$. This results in a concave curve and we use $\mathrm{slope}(e_i)$ to denote the slope of e_i after projection.

Spielman & Teng (2004) study the shadow path size of the smoothed LP model, $\max c^\mathsf{T}x, (A + \hat{A})x \le b + \hat{b}$, where the rows of (A, b) are normalized with l_2 norms at most 1, and each entry of $\hat{A}, \hat{b}$ is an independent Gaussian noise[2] with zero mean and $1/\kappa^2$ variance.

Theorem 6.2 (Spielman & Teng (2004))**.** *The expected size of any shadow path for $Q = \{x \mid (A + \hat{A})x \le b + \hat{b}\}$ is upper bounded by* $\mathrm{poly}(n, r, \kappa)$.

[2]The Gaussian distribution $\mathcal{N}(0, \kappa^{-2})$ is $\frac{\kappa}{2\pi}$-bounded. Sadly, current results of smoothed analysis for linear programs only hold for particular distributions, such as Gaussian or Laplacian noises.

Corollary 6.3. *The sample complexity for smoothed multi-objective linear programs is $\tilde{O}\left(\frac{k}{\varepsilon^2}\right)$.*

6.2. Special LP: Markov decision processes

The smoothed shadow path bound for general linear programs requires Gaussian noises on all coefficient entries. We can relax the smoothed analysis model if the linear program has a special structure.

In the following, we study multi-objective *Markov decision processes* (MDPs), which can be formulated as a special kind of linear programming. MDP is the fundamental model for reinforcement learning and stochastic optimization. An MDP consists of the following components: A set of states S, a set of actions A, a transition probability map $\mathbb{P}[S_{t+1} \mid S_t, A_t]$, a reward function $R : S \times A \to [-1, 1]$, and a discount factor $\gamma \in (0, 1)$. We aim to find a policy $\pi : S \to A$ to maximize the expected accumulated reward using π. For the case of multi-objective MDPs, a natural approach is to optimize the weighted sum of multiple reward functions, e.g., Goldie & Mirhoseini (2020) and Yang et al. (2023).

It is well-known that MDPs can be solved by linear programming (Bertsekas & Tsitsiklis, 1996):

$$\begin{aligned} &\min_V \sum_{s\in S} d(s)\cdot V(s), \\ &\text{s.t. } V(s) \ge R(s,a) + \gamma \sum_{s'\in S} \mathbb{P}[s' \mid s, a] V(s'), \\ &\qquad \forall\, s \in S, a \in A. \end{aligned}$$

Here, V is the value of Bellman's equation, and d can be interpreted as a distribution over S such that $\sum_{s\in S} d(s) = 1$. Then, the objective is equal to the total expected accumulated reward if the initial state is sampled from this distribution. Since the optimal policy is independent of the initial state, we can set d as an arbitrary positive vector. For simplicity of analysis, we set $d(s) = 1/|S|$, i.e., a uniform distribution over S.

To define a multi-objective MDP, we assume the reward function $R(s, a) = (1-\rho)\cdot R_1(s, a) + \rho \cdot R_2(s, a)$ for $0 \le \rho \le 1$. We aim to learn a set of k values of ρ to maximize the performance metric u. In the worst case, an upper bound for the sample complexity of $u(\cdot, I)$ is $\tilde{O}(k|S|\varepsilon^{-2})$, since there are at most $|A|^{|S|}$ choices of the policy. Thus, we can partition $[0, 1]$ into at most $|A|^{|S|}$ intervals such that the optimal policy is constant for ρ in each interval.

Now, we consider the smoothed analysis setting. We assume that the rewards $R_1(s, a)$ and $R_2(s, a)$ for each pair (s, a) are sampled independently from κ-bounded distributions over $[-1, 1]$. Such instances are called κ-smoothed.

Theorem 6.4. *The sample complexity for κ-smoothed multi-objective MDPs is $\tilde{O}\left(\frac{k}{\varepsilon^2}\right)$.*

To apply the shadow path method, we instead consider the dual problem of the MDP:

$$\max_{\mu} \sum_{s\in S}\sum_{a\in A} R(s,a)\cdot\mu(s,a),$$
$$\text{s.t.} \sum_{a\in A}\mu(s,a) = d(s) + \gamma\sum_{s'\in S}\sum_{a\in A}\mathbb{P}[s\mid s',a]\mu(s',a),$$
$$\mu(s,a)\geq 0,\ \forall\, s\in S, a\in A.$$

Lemma 6.5. *The expected size of the shadow path of the κ-smoothed dual MDP problem is upper bounded by* $O\left(\frac{|S|^2|A|\kappa}{1-\gamma}\right)$.

Theorem 6.4 is a direct corollary of Lemma 6.5. The main proof technique is to observe that for each v_i on the shadow path, there exists some (s,a) such that $\mu(s,a)=0$ for v_i and $\mu(s,a)>0$ for v_{i+1}. We show that the contribution of such (s,a) to $\text{slope}(e_{i+1})$ is $O(|S|\kappa/(1-\gamma))$-bounded, thus leading to a polynomial bound in expectation.

Remark 6.6. Although we prove a polynomial bound for the shadow path size of MDPs in the smoothed setting, we are unable to prove a super-polynomial lower bound in the worst case. Note that the classic policy iteration method is known to solve MDPs in polynomial time (Ye, 2011). It is possible that the shadow path size is also polynomial in the worst case. We left such analysis as future work.

7. Experiments

In this section, we perform empirical verifications of our theoretical results. Due to the page limit, we omit the detailed experimental settings to Appendix E.

7.1. Generalization bounds

First, we verify the effectiveness of smoothed analysis in our sample complexity analysis. We evaluate the generalization bound on local search and Markov decision processes. For local search, we consider the VLSI circuit partitioning problem (Hu et al., 2022). For MDP, we consider a resource-gathering problem in previous work on multi-objective reinforcement learning (Barrett & Narayanan, 2008).

We tune the parameter ρ of the weighted sum method, and compute the maximum number of pieces in the piecewise structure of the solution found by the algorithm. We report the maximum number of pieces over problem instances of a particular size. We also report the generalization error. Since the performance metric may depend on the specific application, we simply compute the maximum gap of two objectives averaged on training and evaluation instances. Note that the generalization error depends on the range of the objective. We normalize the objective by the maximum value among data sets to make the error between $[0,1]$.

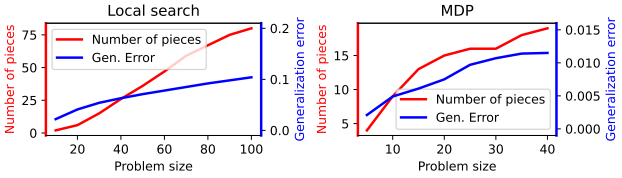

Figure 3. The number of discontinuities in the piecewise structure (red) and the generalization error (blue) for local search (left) and MDPs (right).

Table 2. The Pareto volume of learned parameters for SALT. Our empirical risk minimization algorithm achieves larger Pareto volumes, compared with the vanilla geometric series setting in SALT.

k	ERM (ours)	Vanilla SALT	k	ERM (ours)	Vanilla SALT
1	**0.1157**	0.0159	7	**0.1618**	0.1233
2	**0.1338**	0.0206	8	**0.1625**	0.1399
3	**0.1487**	0.0374	9	**0.1632**	0.1511
4	**0.1529**	0.0557	10	**0.1636**	0.1573
5	**0.1559**	0.0782	11	**0.1638**	0.1610
6	**0.1607**	0.1030	12	**0.1639**	0.1626

The results are illustrated in Figure 3. The problem size denotes the number of nodes (for circuit partitioning) and the number of rows and columns of the grid environment (for MDP). Although in the worst case, the number of pieces may be exponential in the problem size, the empirical results show that the number of pieces grows mildly. This indicates that the worst-case behavior does not appear in practice and our smoothed analysis matches experiments. As a consequence, the generalization error is also small.

7.2. Optimizing Pareto volume

Then, we experimentally show that the empirical risk minimization algorithm in Section 3.2 can optimize the Pareto volume of learned parameters. We conduct experiments on shallow-light Steiner tree benchmarks from real-world VLSI designs.

The original implementation of SALT uses a simple heuristic that selects the values of ρ from a geometric series. We compare our method with this baseline. Let k denote the number of learned parameters. The result is listed in Table 2. The empirical risk minimization algorithm obtains larger Pareto volume for the shallowness-lightness curve, especially when k is small. This indicates the strength of an empirical risk minimization algorithm in tuning a multi-objective optimization algorithm.

8. Conclusion

This work studies the sample complexity of data-driven algorithm configuration for multi-objective optimization algorithms. We present sample complexity bounds for several applications, in which the algorithmic behavior exhibits a piecewise-constant structure.

Our work suggests several directions for future research. First, as discussed in Section D.2, can we extend the smoothed analysis to the case of more than 2 objectives? We conjecture that a smoothed sample complexity of $\tilde{O}(kd\varepsilon^{-2})$ is possible for d objectives. Second, can we extend the sample complexity bound for MDPs to environments beyond tabular cases, e.g., MDPs with continuous features? Finally, can we prove sample complexity bounds for branch-and-bound solvers of integer linear programs? The analysis will be much harder since the algorithm solves a relaxed linear program on each search tree node.

Acknowledgements

This work was supported by the Key Program of National Natural Science Foundation of China (No. 62034005).

Impact Statement

This paper presents theoretical studies on data-driven algorithm configuration of multiple-objective optimization. There are no potential societal consequences of our work that must be specifically highlighted here.

References

Balcan, M.-F. Data-driven algorithm design. *arXiv preprint arXiv 2011.07177*, 2020.

Balcan, M. F. and Sharma, D. Learning accurate and interpretable decision trees. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

Balcan, M.-F., Dick, T., Sandholm, T., and Vitercik, E. Learning to branch. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 344–353. PMLR, 2018a.

Balcan, M.-F., Dick, T., and Vitercik, E. Dispersion for data-driven algorithm design, online learning, and private optimization. In *IEEE 59th Annual Symposium on Foundations of Computer Science*, pp. 603–614, 2018b.

Balcan, M.-F., Dick, T., and Pegden, W. Semi-bandit optimization in the dispersed setting. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence*, volume 124, pp. 909–918, 2020a.

Balcan, M.-F., Sandholm, T., and Vitercik, E. Refined bounds for algorithm configuration: The knife-edge of dual class approximability. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 580–590, 2020b.

Balcan, M.-F., DeBlasio, D., Dick, T., Kingsford, C., Sandholm, T., and Vitercik, E. 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*, pp. 919–932, 2021a.

Balcan, M.-F., Sandholm, T., and Vitercik, E. Generalization in portfolio-based algorithm selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12225–12232, 2021b.

Balcan, M.-F., Prasad, S., Sandholm, T., and Vitercik, E. Structural analysis of branch-and-cut and the learnability of Gomory mixed integer cuts. In *Advances in Neural Information Processing Systems*, volume 35, pp. 33890–33903, 2022.

Barrett, L. and Narayanan, S. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 41–47, 2008.

Bartlett, P., Indyk, P., and Wagner, T. Generalization bounds for data-driven numerical linear algebra. In *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178, pp. 2013–2040, 2022.

Bertsekas, D. P. and Tsitsiklis, J. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Blot, A., Hoos, H. H., Jourdan, L., Kessaci-Marmion, M.-É., and Trautmann, H. MO-ParamILS: A multi-objective automatic algorithm configuration framework. In *Learning and Intelligent Optimization*, pp. 32–47, 2016.

Blot, A., Marmion, M.-E., Jourdan, L., and Hoos, H. H. Automatic configuration of multi-objective local search algorithms for permutation problems. *Evolutionary Computation*, 27(1):147–171, 2019.

Chen, G. and Young, E. F. Y. SALT: Provably good routing topology by a novel Steiner shallow-light tree algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(6):1217–1230, 2020.

Chen, T.-C., Yuh, P.-H., Chang, Y.-W., Huang, F.-J., and Liu, T.-Y. MP-Trees: A packing-based macro placement algorithm for modern mixed-size designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(9):1621–1634, 2008.

Cheng, H., Khalife, S., Fiedorowicz, B., and Basu, A. Sample complexity of algorithm selection using neural networks and its applications to branch-and-cut. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Chu, C. and Wong, Y.-C. FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):70–83, 2008.

Córdova, J. and Lee, Y.-H. A heuristic algorithm for the rectilinear Steiner arborescence problem. 1994.

Feige, U. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

Goldie, A. and Mirhoseini, A. Placement optimization with deep reinforcement learning. In *Proceedings of the 2020 International Symposium on Physical Design*, pp. 3–7, 2020.

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

Herzel, A., Ruzika, S., and Thielen, C. Approximation methods for multiobjective optimization problems: A survey. *INFORMS Journal on Computing*, 33(4):1284–1299, 2021.

Hu, K.-S., Lin, I.-J., Huang, Y.-H., Chi, H.-Y., Wu, Y.-H., and Shen, C.-F. C. ICCAD 2022 CAD contest problem B: 3D placement with D2D vertical connections. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2022.

Johnson, D. S., Papadimitriou, C. H., and Yannakakis, M. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

Khodak, M., Chow, E., Balcan, M. F., and Talwalkar, A. Learning to relax: Setting solver parameters across a sequence of linear system instances. In *The Twelfth International Conference on Learning Representations*, 2024.

Kim, M.-C. and Hu, J. Incremental timing-driven placement and benchmark suite. Available at https://www.iccad-contest.org/2015/problem_c/default.html. 2015.

Li, W., Qu, Y., Chen, G., Ma, Y., and Yu, B. TreeNet: Deep point cloud embedding for routing tree construction. In *Asia and South Pacific Design Automation Conference*, pp. 164–169, 2021.

Lin, Y., Dhar, S., Li, W., Ren, H., Khailany, B., and Pan, D. Z. DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement. In *Proceedings of the 56th Annual Design Automation Conference*, pp. 1–6, 2019.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of Machine Learning*. The MIT Press, 2nd edition, 2018.

Rakhlin, A., Sridharan, K., and Tewari, A. Online learning: Stochastic, constrained, and smoothed adversaries. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

Ravi, R. and Goemans, M. X. The constrained minimum spanning tree problem (Extended abstract). In *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory*, pp. 66–75, 1996.

Röglin, H. and Teng, S.-H. Smoothed analysis of multiobjective optimization. In *50th Annual IEEE Symposium on Foundations of Computer Science*, pp. 681–690, 2009.

Sakaue, S. and Oki, T. Sample complexity of learning heuristic functions for greedy-best-first and A* search. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

Schede, E., Brandt, J., Tornede, A., Wever, M., Bengs, V., Hullermeier, E., and Tierney, K. A survey of methods for automated algorithm configuration. *Journal of Artificial Intelligence Research*, 75:425–487, 2022.

Sechen, C. and Sangiovanni-Vincentelli, A. TimberWolf3.2: A new standard cell placement and global routing package. In *Proceedings of the 23rd Annual Design Automation Conference*, pp. 432–439, 1986.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 525–536, 2018.

Spielman, D. A. and Teng, S.-H. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

Wang, J., Zheng, Y., Zhang, Z., Peng, H., and Wang, H. A novel multi-state reinforcement learning-based multi-objective evolutionary algorithm. *Information Sciences*, 688:121397, 2025.

Wolsey, L. A. and Nemhauser, G. L. *Integer and Combinatorial Optimization*. Wiley-Interscience, 1999.

Xue, K., Xu, J., Yuan, L., Li, M., Qian, C., Zhang, Z., and Yu, Y. Multi-agent dynamic algorithm configuration. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

Yang, L., Sun, G., and Ding, H. Towards timing-driven routing: An efficient learning based geometric approach. In *International Conference on Computer Aided Design*, pp. 1–9, 2023.

Ye, Y. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed

discount rate. *Mathematics of Operations Research*, 36: 593–603, 2011.

Yu, H.-C., Lin, Y.-H., Chen, Z., Li, B., Huang, X., Schlichtmann, U., Ho, T.-Y., and Yao, H. Contamination-aware synthesis for programmable microfluidic devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):5016–5029, 2022.

Yu, X., Xu, P., Wang, F., and Wang, X. Reinforcement learning-based differential evolution algorithm for constrained multi-objective optimization problems. *Engineering Applications of Artificial Intelligence*, 131:107817, 2024.

Zhang, T., Georgiopoulos, M., and Anagnostopoulos, G. C. SPRINT multi-objective model racing. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 1383–1390, 2015.

Zhang, X., Lin, X., Xue, B., Chen, Y., and Zhang, Q. Hypervolume maximization: A geometric view of Pareto set learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Zhao, Y., Wang, L., Yang, K., Zhang, T., Guo, T., and Tian, Y. Multi-objective optimization by learning space partition. In *International Conference on Learning Representations*, 2022.

A. Related Work

In the following, we briefly survey related work to ours, and discuss comparisons with them.

Data-driven algorithm design. Gupta & Roughgarden (2017) initially establish the framework of data-driven algorithm design, where both statistical learning and online learning settings are considered. For statistical learning, this framework has been applied to learning branch policies for branch-and-bound (Balcan et al., 2018a), learning cuts for branch-and-cut (Balcan et al., 2022; Cheng et al., 2024), parameter tuning for string alignment and mechanism design (Balcan et al., 2021a), learning embeddings for A* search (Sakaue & Oki, 2022), data-driven numerical linear algebra (Bartlett et al., 2022), tuning decision tree models (Balcan & Sharma, 2024), to name a few. For online learning, this framework has been applied to tuning greedy heuristics (Gupta & Roughgarden, 2017; Balcan et al., 2018b), tuning clustering algorithms (Balcan et al., 2020a), tuning linear system solvers (Khodak et al., 2024), etc. Our work focuses on statistical learning of multi-objective optimization algorithms.

Our main technical tool is smoothed analysis, which reduces the intrinsic complexity of the parameterized multi-objective optimization algorithm family. Note that smoothed analysis is also applied in online learning of data-driven algorithm design (Gupta & Roughgarden, 2017; Balcan et al., 2018b) (i.e., the *dispersion* property in Balcan et al. (2018b)). In fact, their regret bounds can be turned into statistical ones by applying online-to-batch results. However, we emphasize that our method is different from dispersion. Dispersion is proposed to overcome the impossibility of no-regret online learning for worst-case instances. For a family of parameterized algorithms with K discontinuities in the piecewise structure, the dispersion property says that these K discontinuities do not concentrate tightly in the smoothed analysis setting, and thus no-regret online learning is possible by making a discretization-based method. However, the dispersion property does not present tools for bounding the number of discontinuities K. Our work shows that the number of discontinuities in the piecewise structure can be greatly reduced (i.e., from exponential to polynomial) using smoothed analysis. Therefore, our technique is orthogonal to dispersion.

Balcan et al. (2020b) also propose a method to reduce the intrinsic complexity of the piecewise structures. They use a surrogate function with a simpler structure to approximate the original performance metric function. This approach yields a data-dependent numerical bound for the generalization error. In contrast, our work considers asymptotic bounds without accessing the concrete training data.

Finally, we remark that smoothed analysis is also applied to interpolate between distributional and adversarial settings of classic learning problems, such as online prediction (e.g., Rakhlin et al. (2011)). Our technique is different from classic learning problems since our analysis focuses on the combinatorial structure of multi-objective optimization algorithm configuration.

Algorithm configuration for multi-objective optimization. Algorithm configuration is concerned with the problem of searching parameter configurations of a parameterized algorithm. Several work (Zhang et al., 2015; Blot et al., 2016; 2019; Xue et al., 2022; Yu et al., 2024; Wang et al., 2025) have been devoted to algorithm configuration with multiple objectives. We refer readers to a comprehensive survey of algorithm configuration (Schede et al., 2022). These work are generally empirical, and the theoretical foundation of multi-objective algorithm configuration is lacking.

Machine learning for multi-objective optimization. Previous work also explores the integration of machine learning techniques into multi-objective optimization. Zhao et al. (2022) combine learning models with Monte-Carlo tree search to improve multi-objective black-box optimization algorithms. Zhang et al. (2023) consider modeling the Pareto set using neural networks. These work are different from ours since we focus on tuning the parameter of multi-objective optimization algorithms.

Smoothed analysis of algorithms. Smoothed analysis is proposed by Spielman & Teng (2004) to explain the efficiency of the Simplex method for linear programs in practice. They show that, with a small Gaussian noise on the linear programming instance, the expected time complexity of Simplex is polynomial, breaking the exponential lower bound in worst-case analysis. Smoothed analysis has then been applied to go beyond worst-case analysis of many algorithms.

A related work to ours is due to Röglin & Teng (2009), which shows that a multi-objective binary integer program has only a polynomial number of Pareto-optimal solutions under smoothed analysis. However, their work is different from ours: (1) They consider the optimal Pareto solution set of the problem, while we focus on the behavior of algorithms (which may not produce optimal solutions); (2) Their analysis focuses on binary integer programs, while our framework can be applied to other problems.

B. Preliminaries

In this section, we introduce preliminaries on multi-objective optimization and statistical learning theory, which serve as fundamental tools for our analysis. We refer readers to, e.g., Mohri et al. (2018) for more details of learning theory.

B.1. Preliminaries on multi-objective optimization

For a maximization problem[3] with d objectives, we use an objective vector $o \in \mathbb{R}_+^d$ to denote the d objectives of a solution. An objective vector o is said to be *dominated* by another o' if $o_i \leq o'_i$ for any $1 \leq i \leq d$. We use $o \preceq o'$ to denote domination.

Definition B.1 (Pareto volume)**.** For a set of solutions $S = \left\{o^{(1)}, \ldots, o^{(n)}\right\}$, the *Pareto volume* $\mathsf{Vol}(S)$ is the (hyper-)volume of the set that contains points dominated by at least one point in S, i.e.,

$$\mathsf{Vol}(S) = \int_{o \in \mathbb{R}_+^d} \mathbb{I}(\exists\, o' \in S,\, o \preceq o')\mathrm{d}o.$$

B.2. Preliminaries on learning theory

Definition B.2 (VC- and pseudo-dimension)**.** Let $\mathcal{H}$ denote a set of functions from some domain $\mathbb{X}$ to $\mathbb{R}$. A finite set $S = \{x_1, \ldots, x_m\} \subseteq \mathbb{X}$ is *shattered* by $\mathcal{H}$, if there exist *witnesses* $r_1, \ldots, r_m$ such that, for each of the 2^m subsets $T \subseteq S$, there exists $h \in \mathcal{H}$ such that $h(x_i) > r_i$ if and only if $x_i \in T$. The *pseudo-dimension* $\mathrm{Pdim}(\mathcal{H})$ is the cardinality of the largest subset of $\mathbb{X}$ shattered by $\mathcal{H}$. Specially, if $h \in \mathcal{H}$ are binary functions ($h : \mathbb{X} \to \{-1, 1\}$), the pseudo-dimension is also called the *VC-dimension* $\mathrm{VCdim}(\mathcal{H})$.

By definition, we have the following fact.

Fact B.3. *Let $\mathcal{H}$ be a function class from some domain $\mathbb{X}$ to $\mathbb{R}$, and $\mathcal{H}' = \{h' \,|\, h' : (x, r) \mapsto \mathrm{sgn}(h(x) - r), h \in \mathcal{H}, x \in \mathbb{X}, r \in \mathbb{R}\}$. Then, we have $\mathrm{VCdim}(\mathcal{H}) = \mathrm{Pdim}(\mathcal{H}')$.*

Lemma B.4 (Sauer)**.** *Let $\mathcal{H}$ be a function class from some domain $\mathbb{X}$ to $\{1, -1\}$ with $\mathrm{VCdim}(\mathcal{H}) = d$. For any $S = \{x_1, \ldots, x_m\} \subseteq \mathbb{X}$, we have*

$$|\{(h(x_1), \ldots, h(x_m)) \,|\, h \in \mathcal{H}\}| \leq \sum_{i=0}^{d} \binom{m}{i} = O(m^d).$$

Definition B.5 (Rademacher complexity)**.** Let $\mathcal{H}$ denote a set of functions from some domain $\mathbb{X}$ to $[0, H]$. The *empirical Rademacher complexity* of $\mathcal{H}$ for a finite set $S = \{x_1, \ldots, x_m\} \subseteq \mathbb{X}$ is

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

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 $\mathcal{D}$.

Lemma B.6 (Massart)**.** *For a function class $\mathcal{H}$ from $\mathbb{X}$ to $[0, H]$, and $S = \{x_1, \ldots, x_m\} \subseteq \mathbb{X}$, we have*

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

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

Theorem B.7 (Uniform convergence)**.** *Given m i.i.d. samples $S = \{x_1, \ldots, x_m\}$ from a distribution $\mathcal{D}$, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, for any $h \in \mathcal{H}$, we have*

$$\left|\frac{1}{m}\sum_{i=1}^{m} h(x_i) - \mathbb{E}_{x \sim \mathcal{D}}[h(x)]\right| \leq O(1) \cdot \left(\mathsf{complexity}(\mathcal{H}) + H\sqrt{\frac{1}{m}\log\frac{1}{\delta}}\right),$$

where $\mathsf{complexity}(\mathcal{H})$ is the Rademacher complexity $\mathcal{R}_{\mathcal{D}}(\mathcal{H})$, the empirical Rademacher complexity $\hat{\mathcal{R}}_S(\mathcal{H})$, or a pseudo-dimension-based term $H\sqrt{\frac{\mathrm{Pdim}(\mathcal{H})}{m}}$.

[3] For minimization problems, we can transform it into a maximization problem by selecting a baseline with objectives $(o_1^*, \ldots, o_d^*)$, and consider the Pareto volume of $(o_1^* - o_1, \ldots, o_d^* - o_d)$ for objectives $(o_1, \ldots, o_d)$.

C. Omitted Proofs

C.1. Proofs in Section 3.1

Proof of Theorem 3.2. Let $\mathcal{U} = \{u(P,\cdot) \mid P \in \mathcal{P}^k\}$. By Theorem B.7, it suffices to bound the pseudo-dimension $\mathrm{Pdim}(\mathcal{U}) = O(k \log t)$.

Suppose $\mathrm{Pdim}(\mathcal{U}) = m$. Let $S = \{I_1, \ldots, I_m\}$ be m instances that can be shattered by $\mathcal{U}$ and $r_1, \ldots, r_m \in \mathbb{R}$ be the witnesses. For every $S' \subseteq S$, there exists $P_{S'} \in \mathcal{P}^k$ such that $I_i \in S'$ if and only if $u(P, I_i) > r_i$. Let $M = \{P_{S'} \mid S' \subseteq S\}$ and $|M| = 2^m$.

By Definition 3.1, the parameter space $\mathcal{P}$ can be partitioned into t intervals such that in each interval, $\mathcal{A}(\cdot, I)$ is constant. Combining the intervals for m instances, we can partition $\mathcal{P}$ into $m \cdot t$ intervals such that the function is constant on all m instances. Since $P \in \mathcal{P}^k$, the vector $(u(P, I_1), \ldots, u(P, I_m))$ can take at most $(mt)^k$ values. Therefore, we have $2^m = |M| \le (mt)^k$. Solving for m yields $m = O(k \log t)$. □

Proof of Theorem 3.3. For simplicity, we use $\mathbb{E}_{I \sim \tilde{\mathcal{D}}}[\cdot]$ to denote $\mathbb{E}_{\tilde{I} \sim \tilde{\mathcal{D}}} \mathbb{E}_{I \sim \tilde{I}}[\cdot]$. Given instances $I_{[m]} = \{I_1, \ldots, I_m\}$, let $\hat{\mathcal{R}}_{I_{[m]}}(\mathcal{U}) = \mathbb{E}_\sigma \left[\sup_{P \in \mathcal{P}^k} \frac{1}{m} \sum_{i=1}^m \sigma_i u(P, I_i)\right]$ be the empirical Rademacher complexity of $\mathcal{U} = \{u(P,\cdot) \mid P \in \mathcal{P}^k\}$. Let $\mathcal{R}(\mathcal{U}) = \mathbb{E}_{I_{[m]} \sim \tilde{\mathcal{D}}^m}[\hat{\mathcal{R}}_{I_{[m]}}(\mathcal{U})]$ be the Rademacher complexity. Our goal is to bound the Rademacher complexity of $\mathcal{U}$.

Notice that

$$\mathcal{R}(\mathcal{U}) = \mathop{\mathbb{E}}_{\tilde{I}_{[m]} \sim \tilde{\mathcal{D}}^m} \mathop{\mathbb{E}}_{I_{[m]} \sim \tilde{I}_{[m]}} [\hat{\mathcal{R}}_{I_{[m]}}(\mathcal{U})] \le \sup_{\tilde{I}_{[m]}} \mathop{\mathbb{E}}_{I_{[m]} \sim \tilde{I}_{[m]}} [\hat{\mathcal{R}}_{I_{[m]}}(\mathcal{U})].$$

For any m smoothed instances $\tilde{I}_1, \ldots, \tilde{I}_m$, we define the smoothed empirical Rademacher complexity as $\tilde{\mathcal{R}}_{\tilde{I}_{[m]}}(\mathcal{U}) = \mathbb{E}_{I_{[m]} \sim \tilde{I}_{[m]}}[\hat{\mathcal{R}}_{I_{[m]}}(\mathcal{U})]$. Combining the t intervals for each instances, and considering the choice of k parameters, we have $\sum_{i=1}^m u(P, I_i)$ takes at most $(mt)^k$ values. By Massart's lemma, we have

$$\begin{aligned}
\tilde{\mathcal{R}}_{\tilde{I}_{[m]}}(\mathcal{U}) &\le O(1) \cdot \mathop{\mathbb{E}}_{I_{[m]} \sim \tilde{I}_{[m]}} \left[\sqrt{\frac{\log((mt)^k)}{m}}\right] \\
&= O(1) \cdot \mathop{\mathbb{E}}_{I_{[m]} \sim \tilde{I}_{[m]}} \left[\sqrt{\frac{k(\log t + \log m)}{m}}\right] \le O(1) \cdot \sqrt{\frac{k(\log \mathbb{E}_{I \sim \tilde{I}}[t] + \log m)}{m}},
\end{aligned}$$

where the last inequality is due to Jensen's inequality. This yields the desired result by Theorem B.7. □

C.2. Proofs in Section 3.2

Proof of Lemma 3.5. It is obvious that Vol is monotone. We only prove the submodularity. For any $T_1 \subseteq T_2 \subseteq S$, and $u_0 \in S \setminus T_2$, we have

$$\begin{aligned}
&(\mathsf{Vol}(T_1 \cup \{u_0\}) - \mathsf{Vol}(T_1)) - (\mathsf{Vol}(T_2 \cup \{u_0\}) - \mathsf{Vol}(T_2)) \\
=& \int_u (\mathbb{I}((\forall u' \in T_1, u \not\preceq u') \wedge (u \preceq u_0)) - \mathbb{I}((\forall u' \in T_2, u \not\preceq u') \wedge (u \preceq u_0))) \mathrm{d}u \\
=& \int_u \mathbb{I}((\forall u' \in T_1, u \not\preceq u') \wedge (\exists u' \in T_2 \setminus T_1, u \preceq u') \wedge (u \preceq u_0)) \mathrm{d}u \\
\ge& 0,
\end{aligned}$$

which satisfies Definition 3.4. □

Proof of Theorem 3.6. Since $\mathcal{A}(\rho, I)$ is piecewise-constant, it suffices to consider selecting ρ from at most $t \cdot m$ pieces. Note that the average of Vol is also monotone and submodular. We can use the greedy algorithm to approximately maximize the empirical Pareto volume.

Fact C.1 (Wolsey & Nemhauser (1999))**.** *The greedy algorithm that selects k elements to maximize a monotone and submodular function yields $(1 - \frac{1}{e})$-approximation to the optimal solution.*

Let P^* denote the optimal set of parameters that maximize $\mathbb{E}_{I\sim\mathcal{D}}[u(P,I)]$, $\hat{P}^*$ denote the optimal solution to problem (1), and $\hat{P}$ denote the solution found by the greedy algorithm in Fact C.1. We have, with probability at least $1-\delta$,

$$\begin{aligned}
\mathbb{E}_{I\sim\mathcal{D}}[u(\hat{P},I)] &\geq \frac{1}{m}\sum_{i=1}^{m} u(\hat{P},I_i) - O\left(\sqrt{\frac{1}{m}\left(k\log t+\log\delta^{-1}\right)}\right)\\
&\geq \left(1-\frac{1}{e}\right)\cdot\frac{1}{m}\sum_{i=1}^{m} u(\hat{P}^*,I_i) - O\left(\sqrt{\frac{1}{m}\left(k\log t+\log\delta^{-1}\right)}\right)\\
&\geq \left(1-\frac{1}{e}\right)\cdot\frac{1}{m}\sum_{i=1}^{m} u(P^*,I_i) - O\left(\sqrt{\frac{1}{m}\left(k\log t+\log\delta^{-1}\right)}\right)\\
&\geq \left(1-\frac{1}{e}\right)\cdot\mathbb{E}_{I\sim\mathcal{D}}[u(P^*,I)] - \left(2-\frac{1}{e}\right)\cdot O\left(\sqrt{\frac{1}{m}\left(k\log t+\log\delta^{-1}\right)}\right),
\end{aligned}$$

which is the desired result by plugging in m. The first and the last inequality comes from Theorem 3.2, the second inequality comes from Fact C.1, and the third inequality is due to the optimality of $\hat{P}^*$. □

Proof of Theorem 3.7. Proof by reduction from maximum k-set-cover. The max k-cover problem is defined as follows: Given a universe $U=\{a_1,\ldots,a_m\}$, t subsets $S_1,\ldots,S_t\subseteq U$, and an integer k. We aim to select k sets from $S_1,\ldots,S_t$, such that the union of the k sets has maximum cardinality.

Theorem C.2 (Theorem 5.3 of Feige (1998))**.** *For any $\varepsilon>0$, max k-cover cannot be approximated in polynomial time within a ratio of $(1-\frac{1}{e}+\varepsilon)$, unless $\mathsf{P}=\mathsf{NP}$.*

Given a max k-cover instance, we construct an instance of problem (1). Each element in the universe corresponds to a training sample I_i. The algorithm $\mathcal{A}(\cdot,I_i)$ contains t pieces, and we let the partition of pieces be the same for each sample x_i. In piece j of I_i, if $a_i\in S_j$, we let $\mathcal{A}(\cdot,I_i)=(1,1)$ (i.e., with Pareto volume 1). Otherwise, we let $\mathcal{A}(\cdot,I_i)=(0,0)$ (i.e., with Pareto volume 0). Therefore, selecting k subsets from $S_1,\ldots,S_t$ is equivalent to selecting k pieces to maximize the Pareto volume. Since the reduction preserves the approximation ratio, our result follows from Theorem C.2. □

C.3. Proofs in Section 4

Proof of Lemma 4.1. As the value of ρ varies from 0 to $+\infty$, the output of SALT changes as some point v becomes or is no longer a breakpoint. Such discontinuities happen as $\rho=\frac{\mathsf{dist}(v)}{\|r-v\|_1}-1$. Fix the point v. The value of $\|r-v\|_1$ is a constant, and the distance from the root r to v on the tree $\mathsf{dist}(v)$ is in the form of $\|r-v'\|_1+d_T(v',v)$ for some $v'\in S$, where $d_T(v',v)$ is the distance between v' and v on the initial Steiner minimum tree. Therefore, $\mathsf{dist}(v)$ can take at most $n-1$ values. Combining the breakpoints for each $v\in S$, the number of discontinuities is at most $n(n-1)+1\leq n^2$. □

Proof of Lemma 4.3. It is easy to notice that the behavior of the algorithm is fixed if z^* is fixed (see Figure 1). Therefore, it suffices to consider the number of pieces of $L(z)$ (the red segments in Figure 1).

The following property is crucial to our analysis.

Lemma C.3 (Lemma 3.2 of Ravi & Goemans (1996))**.** *Suppose T and T' are two spanning trees corresponding to two adjacent segments of $L(z)$. Then, T and T' differ by only a single edge swap (i.e., there exist $e\in T$ and $e'\in T$ such that $T'=(T-e)\cup e'$).*

Let $T_1,\ldots,T_n$ denote the spanning tree that minimizes $L(z)$ as z varies from 0 to $+\infty$. We use $(e_i,e_i')\in E^2$ to denote the edge swap between T_i and T_{i+1} in Lemma C.3 for $i\in[n-1]$. We claim that each pair of edges $(e,e')\in E^2$ only appears at most once in $\{(e_i,e_i')\,|\,i\in[n-1]\}$.

Proof by contradiction. Suppose there exist $i\neq j$ such that $(e_i,e_i')=(e_j,e_j')$. We consider the intersection point of $W_1+W_2\cdot z$ for T_i,T_{i+1} and T_j,T_{j+1}. The line $W_1(T_i)+W_2(T_i)\cdot z$ intersects $W_1(T_{i+1})+W_2(T_{i+1})\cdot z$ at $z=\frac{|W_1(T_i)-W_1(T_{i+1})|}{|W_2(T_i)-W_2(T_{i+1})|}=\frac{|w_1(e_i)+w_1(e_i')|}{|w_2(e_i)+w_2(e_i')|}$. Similarly, $W_1(T_j)+W_2(T_j)\cdot z$ also intersects $W_1(T_{j+1})+W_2(T_{j+1})\cdot z$ at

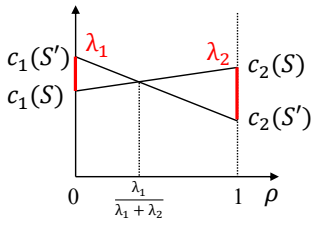

Figure 4. The discontinuity point of local search for two states S and S'.

$z = \frac{|w_1(e_j)+w_1(e'_j)|}{|w_2(e_j)+w_2(e'_j)|}$. Both points have the same z coordinate. However, it is impossible that $T_i, T_{i+1}, T_j, T_{j+1}$ all lie on the minimum segments of $L(z)$. This is a contradiction. □

C.4. Proof of Lemma 5.1

Proof. (Upper bound.) Proof by induction. We claim that after iteration t, the algorithmic behavior is piecewise-constant with at most $(n+1)^t$ pieces.

Initially, $t = 0$, and there is only one interval $[0, 1]$. Suppose our claim holds for iteration t, and consider the next iteration $t+1$. Fix an interval $[a, b]$ from the total of $(n+1)^t$ sub-intervals after iteration t. If $\rho \in [a, b]$, after t iterations, the algorithm finds a fixed state S, and explores the neighborhood $N(S)$. Consider the weighted sum of costs for states in $N(S)$. The cost $c(S') = (1-\rho) \cdot c_1(S') + \rho \cdot c_2(S')$ for each $S' \in \{S\} \cup N(S)$ is a linear function with respect to ρ. In iteration $t+1$, the algorithm finds the state S' with the least cost $c(S')$. (If $c(S)$ is the minimum, the algorithm terminates, and no more sub-intervals are created.) Note that the minimum of $n+1$ lines is a piecewise-linear function with at most $n+1$ pieces. Therefore, the interval $[a, b]$ is divided into at most $n+1$ sub-intervals such that in each sub-interval the algorithmic behavior is fixed in iteration $t+1$. Combining the sub-intervals for each interval $[a, b]$ yields a total of at most $(n+1) \cdot (n+1)^t = (n+1)^{t+1}$ sub-intervals for iteration $t+1$, which is the desired result.

(Lower bound.) We construct the instance as follows. The states and the neighborhood relations form a rooted tree. The root is the initial state S_0, and the tree is a full $(n-1)$-ary tree with depth T. Each non-leaf node has $n-1$ children.

The cost of the root is $c_1(S_0) = c_2(S_0) = 1$, and corresponds to the whole interval $[0, 1]$. We recursively set the costs of each node. Suppose a node p corresponding to state S and interval $[a, b]$ is not a leaf, and $c_1(S) = C_1, c_2(S) = C_2$. We let the $n-1$ children $p_1, \ldots, p_{n-1}$ of p correspond to states $S_1, \ldots, S_{n-1}$, respectively.

Let $\Delta = \frac{b-a}{n-1}$ and $\gamma > 0$ be a real number to be determined later. We set $c_2(S_i) = C_2 - (n-i) \cdot \gamma$ for any $i \in [n-1]$. Then, we set $c_1(S_1) = C_1 - \frac{\gamma(a+\Delta)}{1-a-\Delta}, c_1(S_2) = c_1(S_1) - \frac{\gamma(a+2\Delta)}{1-a-2\Delta}, \cdots, c_1(S_{n-1}) = c_1(S_{n-2}) - \frac{\gamma(a+(n-1)\Delta)}{1-a-(n-1)\Delta}$. We can let γ small enough such that $c_1(S_{n-1}) > 0$ and $c_2(S_1) > 0$. It is easy to verify that the solution of $(1-\rho) \cdot c_1(S_i) + \rho \cdot c_2(S_i) = (1-\rho) \cdot c_1(S_{i+1}) + \rho \cdot c_2(S_{i+1})$ is $\rho = a + i\Delta$. Since $C_1 > c_1(S_1) > \cdots > c_1(S_{n-1})$ and $c_2(S_1) < c_2(S_2) < \cdots < c_2(S_{n-1}) < C_2$, we have $p_1, \ldots, p_{n-1}$ correspond to intervals $[a, a+\Delta], [a+\Delta, a+2\Delta], \ldots, [b-\Delta, b]$, respectively.

Therefore, the tree has a total of $(n-1)^T$ leaves, and each leaf corresponds to an interval of length $1/(n-1)^T$. This gives the $(n-1)^T \geq n^{T/2} = n^{\Omega(T)}$ lower bound. □

C.5. Proof of Theorem 5.2

To prove Theorem 5.2, we first prove a few lemmas. We fix a smoothed instance $\tilde{I}$ and $I \sim \tilde{I}$. Let $\mathcal{E}(\alpha)$ denote the event that $\mathcal{A}(\alpha, I) = \mathcal{A}(\rho, I)$ for any $\rho \in [0, \alpha)$. That is to say, if $\mathcal{E}(\alpha)$ happens, we only need to consider $\rho \in [\alpha, 1]$. The value of α is to be determined later.

Lemma C.4. *For any $\alpha < 0.5$, we have $\mathbb{P}_I[\mathcal{E}(\alpha)] \geq 1 - 2(n+1)^2 T\kappa\alpha$.*

Proof. Let $\mathcal{E}_t(\alpha)$ denote the event that the behavior of the local search algorithm is fixed for $\rho \in [0, \alpha]$ in the t-th iteration. We first bound the probability $\mathbb{P}[\mathcal{E}_t(\alpha)]$.

Fix $t \in [T]$ and consider the iteration t. Let S_i denote the state after iteration i ($i \in [t]$). By the principle of deferred

decisions, we assume the costs $c(S')$ for any $S' \in \mathcal{S}_{\text{past}}$ are fixed, where $\mathcal{S}_{\text{past}} = \cup_{i=0}^{t-1}(N(S_i) \cup \{S_i\})$ contains the states the algorithm has already accessed. Notice that any state $S' \in \mathcal{S}_{\text{past}}$ satisfies $c(S') \geq c(S_{t-1})$. Let $\mathcal{S}_{\text{next}} = \{S' \mid S' \in N(S_{t-1}), S' \notin \mathcal{S}_{\text{past}}\}$. In iteration t, the algorithm chooses the state $S' \in \mathcal{S}_{\text{next}}$ with the minimal cost $c(S')$ and updates the state. Therefore, the behavior of the algorithm changes as ρ passes a discontinuity point satisfying $c(S) = c(S')$, i.e.,

$$(1-\rho)\cdot c_1(S) + \rho \cdot c_2(S) = (1-\rho)\cdot c_1(S') + \rho \cdot c_2(S'), \tag{2}$$

for some $S, S' \in \{S_t\} \cup \mathcal{S}_{\text{next}}$, $S \neq S'$. If $\rho \in [0,1]$, the discontinuity point is $\rho = \frac{\lambda_1}{\lambda_1+\lambda_2}$, where $\lambda_1 = |c_1(S) - c_1(S')|$, and $\lambda_2 = |c_2(S) - c_2(S')|$ (see Figure 4). If $\rho \in [0, \alpha]$, we have $0 \leq \lambda_1 \leq \frac{\alpha}{1-\alpha} \cdot \lambda_2 \leq 2\alpha$. Since λ_1 is 2κ-bounded (due to the absolute value), we have $\mathbb{P}[\mathcal{E}_t(\alpha)] \geq 1 - 2(n+1)^2\alpha\kappa$ using an union bound over choices of S, S'.

Finally, we consider the variance of t. A union bound over $t = 1, \ldots, T$ gives $\mathbb{P}[\mathcal{E}(\alpha)] \geq 1 - 2(n+1)^2 T\alpha\kappa$. □

Lemma C.5. *For a κ-smoothed instance $\tilde{I}$ and $I \sim \tilde{I}$, we can partition $[\alpha, 1]$ into $\#\mathsf{int}$ intervals, such that in each interval, $\mathcal{A}(\cdot, I)$ is fixed. We have $\mathbb{E}_I[\#\mathsf{int}] = O\left(\frac{n^2 T\kappa}{\alpha^2}\right)$.*

Proof. We bound the expectation of the interval number by induction. Let $\#\mathsf{int}_i$ denote the number of intervals in $[\alpha, 1]$ after iteration i. Similarly to Lemma C.4, we fix $t \in [T]$ and consider the iteration t. We follow the notations $S_t, \mathcal{S}_{\text{past}}, \mathcal{S}_{\text{next}}$ as in the proof of Lemma C.4.

Before iteration t, there are $\#\mathsf{int}_{t-1}$ intervals. Each interval $Q \subseteq [\alpha, 1]$ is divided into at most $(n+1)^2$ sub-intervals if the value of ρ satisfying (2) lies in Q. As shown in the proof of Lemma C.4, the value of the discontinuity point $\rho = \frac{\lambda_1}{\lambda_1+\lambda_2}$ (Figure 4), where λ_1 and λ_2 are random variables with support on $[0, 1]$ and probability density at most 2κ. By the principle of deferred decisions, we fix the value of λ_1. We claim that the probability density of the discontinuity point ρ is at most $O(\kappa\alpha^{-2})$, due to the following claim.

Claim C.6. *If a real random variable X has density at most κ, and $Y = z/(X + z')$ satisfies $Y \geq \alpha > 0$ almost surely. Then, Y has density at most $O(|z|\alpha^{-2}\kappa)$.*

Proof. Let $p_X(x)$ and $p_Y(y)$ denote the probability density function of X and Y. We have $X = g(Y) = z/Y - z'$. Therefore, $p_Y(y) = p_X(g(y)) \cdot |g'(y)| \leq \kappa \cdot \frac{|z|}{y^2} \leq \frac{|z|\kappa}{\alpha^2}$, which is the desired result. □

Suppose the length of interval Q is l. In expectation, Q produces at most $O(ln^2\kappa\alpha^{-2})$ new sub-intervals. Combining all intervals Q, the total number of new intervals after iteration t in expectation is at most $O\left(n^2\kappa\alpha^{-2}\right)$.

Therefore, we have $\mathbb{E}[\#\mathsf{int}_t] = \mathbb{E}[\#\mathsf{int}_{t-1}] + O\left(n^2\kappa\alpha^{-2}\right)$, which means $\mathbb{E}[\#\mathsf{int}] = O\left(\frac{n^2 T\kappa}{\alpha^2}\right)$. □

Now we are ready to prove Theorem 5.2. The general idea is to show that with high probability, the event $\mathcal{E}(\alpha)$ happens for all m instances. The total number of possible values for $\sum_{i=1}^m u(P, I_i)$ is upper bounded by $(m \cdot \#\mathsf{int})^k$, which gives an $\tilde{O}(\sqrt{k/m})$ bound for the smoothed Rademacher complexity.

Proof of Theorem 5.2. Given m κ-smoothed instances $\tilde{I}_1, \ldots, \tilde{I}_m$, we aim to bound the smoothed Rademacher complexity $\tilde{\mathcal{R}}_{\tilde{I}_{[m]}}(\mathcal{U})$, where $\mathcal{U}$ denotes the function class $\{u([\rho_1, \ldots, \rho_k], \cdot) \mid 0 \leq \rho_i \leq 1\}$.

By Lemma C.4 and a union bound over m instances, we have with probability at least $1 - 2m(n+1)^2 T\kappa\alpha$,

$$\sum_{i=1}^m u([\rho_1, \ldots, \rho_k], I_i) = \sum_{i=1}^m u([\alpha, \ldots, \alpha], I_i),\ \forall\, 0 \leq \rho_1, \ldots, \rho_k \leq \alpha. \tag{3}$$

Then, it suffices to consider $\rho \in [\alpha, 1]$. We use $\mathcal{E}_{[m]}$ to denote the event that (3) happens. To make the probability that $\mathcal{E}_{[m]}$ happens high enough, we set $\alpha = \frac{1}{m(n+1)^2 T^2\kappa}$. Therefore, we have $\mathbb{P}[\mathcal{E}_{[m]}] \geq 1 - \frac{1}{T}$. We also have $\mathbb{E}[\#\mathsf{int}] = O(n^6 T^5 \kappa^3)$, which is exponentially smaller than the worst-case $n^{O(T)}$ bound. This gives a high-probability logarithmic generalization error bound.

Let $\#\mathsf{int}_{[0,1]}$ denote the total number of intervals for $\rho \in [0,1]$. By Massart's lemma, we have

$$\begin{aligned}\tilde{\mathcal{R}}_{\tilde{I}_{[m]}}(\mathcal{U}) &\le O(1) \cdot \mathbb{E}\left[\sqrt{\frac{k\log(m \cdot \#\mathsf{int}_{[0,1]})}{m}}\right] \\ &\le O(1) \cdot \left(\mathbb{E}\left[\sqrt{\frac{k\log(m \cdot \#\mathsf{int})}{m}} \,\middle|\, \mathcal{E}_{[m]}\right]\mathbb{P}(\mathcal{E}_{[m]}) + \mathbb{E}\left[\sqrt{\frac{k\log(m \cdot \#\mathsf{int}_{[0,1]})}{m}} \,\middle|\, \bar{\mathcal{E}}_{[m]}\right]\mathbb{P}(\bar{\mathcal{E}}_{[m]})\right).\end{aligned}$$

Recall that in the worst-case, $\#\mathsf{int}_{[0,1]} \le (n+1)^T$. Since $\mathbb{P}[\mathcal{E}_{[m]}] > 1 - \frac{1}{T}$, we have

$$\begin{aligned}\tilde{\mathcal{R}}_{\tilde{I}_{[m]}}(\mathcal{U}) &\le O(1) \cdot \left(\mathbb{E}\left[\sqrt{\frac{k\log(m \cdot \#\mathsf{int})}{m}} \,\middle|\, \mathcal{E}_{[m]}\right] + \sqrt{\frac{kT\log(mn)}{m}} \cdot \frac{1}{T}\right) \\ &\le O(1) \cdot \left(\mathbb{E}\left[\sqrt{\frac{k\log(m \cdot \#\mathsf{int})}{m}}\right] + \sqrt{\frac{k\log(mn)}{Tm}}\right) \\ &\le O(1) \cdot \left(\sqrt{\frac{k\log(m \cdot \mathbb{E}[\#\mathsf{int}])}{m}} + \sqrt{\frac{k\log(mn)}{Tm}}\right) \\ &\le O\left(\sqrt{\frac{k}{m}(\log T + \log n + \log\kappa + \log m)}\right) = \tilde{O}\left(\sqrt{\frac{k}{m}}\right).\end{aligned}$$

The second inequality is due to

$$\mathbb{E}[X \mid \mathcal{E}_{[m]}] = \frac{\mathbb{E}[X \cdot \mathbb{I}(\mathcal{E}_{[m]})]}{\mathbb{P}[\mathcal{E}_{[m]}]} \le \frac{1}{1 - 1/T} \cdot \mathbb{E}[X] \le \left(1 + O\left(\frac{1}{T}\right)\right) \cdot \mathbb{E}[X]$$

for any random variable X. The third inequality is due to Jensen's inequality for concave functions. □

C.6. Proof of Theorem 5.3

Proof. The proof is similar to Theorem 5.2. Fix a smoothed instance $\tilde{I}$ and $I \sim \tilde{I}$. Let $\mathcal{E}(\alpha)$ denote the event that $\mathcal{A}(\alpha, I) = \mathcal{A}(\rho, I)$ for any $\rho \in [0, \alpha)$. Let $\#\mathsf{int}_t$ denote the number of intervals after iteration i. Initially, $\#\mathsf{int}_0 = 1$. We study the recursion on $\#\mathsf{int}_t$.

For any $t \in [T]$, let S denote the state after iteration t and S' be the sampled neighbor. If S' has not been visited, the behavior of the algorithm changes as ρ passes a discontinuity satisfying

$$\exp\left(\frac{(1-\rho)\cdot c_1(S) + \rho \cdot c_2(S) - (1-\rho)\cdot c_1(S') - \rho \cdot c_2(S')}{\tau_t}\right) = \mathsf{prob}(t).$$

Let $\lambda_1 = c_1(S') - c_1(S)$, $\lambda_2 = c_2(S') - c_2(S)$. Solving for ρ yields

$$\rho = \frac{\tau_t \cdot \log \mathsf{prob}(t) + \lambda_1}{\lambda_1 - \lambda_2}.$$

Lemma C.7. $\mathbb{P}_I[\mathcal{E}(\alpha)] \ge 1 - 4T\kappa\alpha$.

Proof. Note that since $0 \le c_1(S), c_2(S) \le 1$ for any S, we have

$$\tau_t \log \mathsf{prob}(t) = (1-\rho)\cdot c_1(S) + \rho \cdot c_2(S) - (1-\rho)\cdot c_1(S') - \rho \cdot c_2(S') \in [-1, 1]$$

as long as $\rho \in [0,1]$.

Since $|\lambda_1 - \lambda_2| \le 2$, we have $\rho > \alpha$ as long as $|\tau_t \cdot \log \mathsf{prob}(t) + \lambda_1| > 2\alpha$. Since λ_1 is κ-bounded, we have $\mathbb{P}[|\tau_t \cdot \log \mathsf{prob}(t) + \lambda_1| \le 2\alpha] \le 4\alpha\kappa$. A union bound over $t = 1, \ldots, T$ proves Lemma C.7. □

Lemma C.8. *We can partition* $[\alpha, 1]$ *into* $\#\mathsf{int}$ *intervals, such that* $\mathcal{A}(\cdot, I)$ *is constant in each interval. We have* $\mathbb{E}_I[\#\mathsf{int}] = O\left(\frac{T\kappa}{\alpha^2}\right)$.

Proof. Let $\#\mathsf{int}_t$ denote the number of intervals in $[\alpha, 1]$ after iteration t. By Claim C.6, the probability density of ρ on $[\alpha, 1]$ is at most $O(\kappa\alpha^{-2})$. Thus, the expected number of new intervals for iteration t is at most $O(\kappa\alpha^{-2})$, i.e., $\mathbb{E}[\#\mathsf{int}_t] = \mathbb{E}[\#\mathsf{int}_{t-1}] + O(\kappa\alpha^{-2})$. Solving the recurrence on t shows that $\mathbb{E}[\#\mathsf{int}] = O(T\kappa\alpha^{-2})$. □

Therefore, following the proof of Theorem 5.2, we can bound the smoothed Rademacher complexity by $\tilde{O}(\sqrt{k/m})$ similarly and obtain the desired result. □

C.7. Proof of Lemma 6.5

Proof. To prove this lemma, we first introduce a few notations. We use Q to denote the polyhedron of the dual MDP. Let $Q(R_1, R_2) = (v_0, e_1, v_1, \ldots, e_l, v_l)$ denote the shadow path of Q. Moreover, we use $Q^{s,a} = \{\mu \,|\, \mu \in P, \mu(s, a) = 0\}$ to denote the polyhedron P, but fixing $\mu(s, a) = 0$. We also use $Q^{s,a}(R_1, R_2) = (v_0^{s,a}, e_1^{s,a}, \ldots, v_{l_{s,a}}^{s,a})$ to denote the shadow path of $P^{s,a}$.

The following claim can be easily verified.

Claim C.9. *The optimal solution to the dual MDP is* $\mu^*(s, a) = \sum_{t\geq 0} \gamma^t \mathbb{P}[S_t = s, A_t = a]$, *where* $\mathbb{P}[S_t = s, A_t = a]$ *denotes the probability of* $S_t = s$ *and* $A_t = a$ *at time* t *with the initial state* S_0 *following* $\mathbb{P}[S_0 = s] = d(s)$ *and the policy being the optimal policy* π^*.

By Claim C.9, for each state s, the optimal solution $\mu^*(s, a) = 0$ if $a \neq \pi^*(s)$, since the optimal policy $\pi^* : S \to A$ is deterministic. (Note that in the smoothed analysis setting, the optimal policy is unique with probability 1.) Given a solution μ, we use $\text{inactive}(\mu) = \{(s, a) \,|\, \mu(s, a) = 0\}$ to denote the set of actions that is not used by the corresponding policy $\pi^\mu : s \mapsto \arg\max_a \mu(s, a)$.

Suppose the optimal solution moves from v_i to v_{i+1} through e_i as ρ increases. There must exist some action (s, a) such that $(s, a) \in \text{inactive}(v_i)$ and $(s, a) \notin \text{inactive}(v_{i+1})$ by Claim C.9. Thus, we have

$$|Q(R_1, R_2)| \leq \sum_{i=0}^{l-1} \sum_{s\in S} \sum_{a\in A} \mathbb{I}[(s, a) \in \text{inactive}(v_i) \wedge (s, a) \notin \text{inactive}(v_{i+1})]. \tag{4}$$

Interchange the order of summations and fix a pair of (s, a). We bound the expectation of $\sum_{i=0}^{l-1} \mathbb{I}[(s, a) \in \text{inactive}(v_i) \wedge (s, a) \notin \text{inactive}(v_{i+1})]$.

Since $v_{i-1}, e_i, v_i, e_{i+1}, v_{i+1}$ are on the shadow path $Q(R_1, R_2)$ (assuming $1 \leq i < l$), we have $\text{slope}(e_i) > \text{slope}(e_{i+1})$. If $(s, a) \in \text{inactive}(v_i)$, then, $v_i \in P^{s,a}$, i.e., v_i is also on the shadow path $Q^{s,a}(R_1, R_2)$. Suppose $v_j^{s,a} = v_i$ for some j. Note that $Q^{s,a} \subseteq Q$, we have $\text{slope}(e_j^{s,a}) \geq \text{slope}(e_i)$ and $\text{slope}(e_{i+1}) \geq \text{slope}(e_{j+1}^{s,a})$. Thus, we have

$$\mathbb{I}[(s, a) \in \text{inactive}(v_i) \wedge (s, a) \notin \text{inactive}(v_{i+1})] \leq \mathbb{I}[v_i \in P^{s,a} \wedge \text{slope}(e_j^{s,a}) \geq \text{slope}(e_{i+1}) \geq \text{slope}(e_{j+1}^{s,a})].$$

Plugging this bound into (4) yields

$$|Q(R_1, R_2)| \leq \sum_{s\in S} \sum_{a\in A} \left(2 + \sum_{j=1}^{l_{s,a}-1} \mathbb{I}[\text{slope}(e_{i+1}) \in [\text{slope}(e_{j+1}^{s,a}), \text{slope}(e_j^{s,a})]]\right), \tag{5}$$

where e_{i+1} depends on j. Note that $P^{s,a}$ fixes $\mu(s, a) = 0$. Therefore, the interval $[\text{slope}(e_{j+1}^{s,a}), \text{slope}(e_j^{s,a})]$ does not depend on the value of $R_1(s, a)$ and $R_2(s, a)$. By the principle of deferred decisions, we fix the reward on actions other than (s, a) and consider the randomness of $R_1(s, a)$ and $R_2(s, a)$. Note that

$$\text{slope}(e_{i+1}) = \frac{\langle R_2, v_i\rangle - \langle R_2, v_{i+1}\rangle}{\langle R_1, v_i\rangle - \langle R_1, v_{i+1}\rangle},$$

where $\langle R, \mu \rangle = \sum_{s,a} R(s,a)\mu(s,a)$ is the inner product. Let μ and μ' denote the solution of v_i and v_{i+1}, respectively. Recall that $\mu(s,a) = 0$ and $\mu'(s,a) \geq 0$. We can hence represent $\mathrm{slope}(e_{i+1})$ as

$$\frac{\mu'(s,a) \cdot R_2(s,a) + \text{deterministic term}}{\mu'(s,a) \cdot R_1(s,a) + \text{deterministic term}}.$$

Note that we set $d(s) = 1/|S|$ for any $s \in S$. By Claim C.9, we have $\mu'(s,a) \geq 1/|S|$. Since $R_2(s,a)$ is κ-bounded, and the absolute value of the denominator is upper bounded by $O(1/(1-\gamma))$, we claim that the probability density of $\mathrm{slope}(e_{i+1})$ is upper bounded by $O\left(\frac{|S|\kappa}{1-\gamma}\right)$. Therefore, taking the expectation of (5) yields

$$\begin{aligned}
\mathbb{E}|Q(R_1, R_2)| &\leq O(1) \cdot \sum_{s \in S} \sum_{a \in A} \left(2 + \sum_{j=1}^{l_{s,a}-1} (\mathrm{slope}(e_j^{s,a}) - \mathrm{slope}(e_{j+1}^{s,a})) \cdot \frac{|S|\kappa}{1-\gamma} \right) \\
&\leq O(1) \cdot \sum_{s \in S} \sum_{a \in A} \left(O(1) + \frac{|S|\kappa}{1-\gamma} \cdot (\mathrm{slope}(e_0^{s,a}) - \mathrm{slope}(e_{l_{s,a}}^{s,a})) \right).
\end{aligned}$$

By symmetry, we can break the shadow path into two parts. The slopes of the segments in the first part lie in $[-1, 0)$, and the ones in the second part lie in $(-\infty, -1)$. The expected sizes of the two parts are asymptotically equivalent since we can swap R_1 and R_2 without loss of generality. Therefore, we assume $\mathrm{slope}(e_0^{s,a}) - \mathrm{slope}(e_{l_{s,a}}^{s,a}) \leq 1$. We thus have

$$\mathbb{E}|Q(R_1, R_2)| \leq O\left(\frac{|S|^2|A|\kappa}{1-\gamma}\right),$$

which is the desired bound. □

D. Extensions of our Framework

D.1. Feature-based parameter predictor

A natural extension of algorithm configuration is to train a machine learning model to predict the algorithm parameter based on the features of the instance. Balcan et al. (2021b) present generalization bounds for selecting parameters from a portfolio of finite algorithm configurations. Cheng et al. (2024) study the sample complexity of predicting parameters using neural networks. In the following, we extend our framework for multi-objective optimization to the setting of predicting k parameters using any PAC-learnable (i.e., with finite pseudo-dimension) machine learning models.

Let $\mathcal{D}$ be a distribution of (maybe smoothed) instances, $\mathcal{A}(\rho, I)$ be the output of the algorithm with parameter ρ on instance I, and $u(P, I)$ be the performance metric for a set $P = \{\rho_1, \ldots, \rho_k\}$ of parameter values of ρ. Instead of directly learning P, we learn a predictor $\mathsf{pred}(\theta, I) : \Theta \times \mathcal{I} \to \mathcal{P}^k$ to predict P. For example, if $k = 1$ and x_I is the feature vector of instance I, we may choose a linear model $\mathsf{pred} : (\theta, I) \mapsto \theta^{\mathsf{T}} x_I$ to predict the parameter ρ. We study the sample complexity of learning $\theta \in \Theta$ for performance metric $u(\mathsf{pred}(\cdot, I), I)$. We assume the prediction model has a finite pseudo-dimension so that sample complexity bounds are possible: Let $\mathsf{pred}(\theta, I) = (\mathsf{pred}_1(\theta, I), \ldots, \mathsf{pred}_k(\theta, I))$ be the predicted parameters. We assume $\mathrm{Pdim}\left(\{\mathsf{pred}_i(\theta, \cdot) \mid \theta \in \Theta\}\right) \leq d$ for any $i \in [k]$.

Theorem D.1. *Suppose $\mathcal{A}(\cdot, I)$ is piecewise-constant with at most or in expectation t pieces (in worst- or smoothed-case). The sample complexity of $u(\mathsf{pred}(\cdot, I), I)$ is $\tilde{O}\left(\frac{kd \log t}{\varepsilon^2}\right)$.*

Proof. Fix an instance I. With a little abuse of notations, we use I to denote either an instance or a sample from a smoothed instance. By definition, we can partition the parameter space $\mathcal{P}$ into t intervals, such that the algorithmic behavior is fixed on I in each interval. Let $\rho_0 < \rho_1 < \cdots < \rho_t$ denote the terminals of these intervals. Let $\mathrm{sgn}(\mathsf{pred}_i(\theta, I) - \rho_j)$ denote the sign of $\mathsf{pred}_i(\theta, I) - \rho_j$ for some $1 \leq i \leq k$, $1 \leq j < t$. As we vary the prediction model parameter θ, $u(\mathsf{pred}(\cdot, I), I)$ is fixed if

$$\Phi = (\mathrm{sgn}(\mathsf{pred}_i(\theta, I) - \rho_j))_{1 \leq i \leq k,\, 1 \leq j < t} \in \{-1, 1\}^{kt}$$

is fixed due to the piecewise-constant structure of $\mathcal{A}$. Since the pseudo-dimension of pred_i is at most d, by Fact B.3, the VC-dimension of $\{\mathrm{sgn}(\mathsf{pred}_i(\theta, I) - \rho_j) \mid \theta \in \Theta\}$ is also at most d. Therefore, by Sauer's lemma, the number of values of Φ is at most $O(t^{kd})$.

Now, we vary the fixed instance I. Given m instances, the number of values of

$$(\operatorname{sgn}(\mathsf{pred}_i(\theta, I_l) - \rho_j))_{1\le i\le k,\, 1\le j\le t\cdot m, 1\le l\le m}$$

is at most $O((mt)^{kd})$. Here, ρ_j denotes the union of all interval terminals for m instances. Using Massart's lemma, we obtain that the Rademacher complexity is at most $O\left(\sqrt{\frac{kd\log(mt)}{m}}\right)$. This proves the desired result. □

D.2. Extension to more than two objectives

Previous results focus on learning configurations for multi-objective optimization problems with two objectives. Now, we consider the case of three or more objectives. We show that we can prove similar worst-case sample complexity bounds. For example, we have the following result for local search.

Theorem D.2. *Suppose $\mathcal{D}$ is a distribution over local search instances with d objectives $c_1, \ldots, c_d$. We use local search to optimize the weighted sum objective $c = \sum_{i=1}^d \rho_i \cdot c_i$, with $\sum_{i=1}^d \rho_i = 1$. Let $\mathcal{A}(\boldsymbol{\rho}, I)$ denote the result on instance I with parameter $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_d)$. The sample complexity of $u(\cdot, I)$ is $\tilde{O}\left(\frac{kdT}{\varepsilon^2}\right)$.*

Proof. Fix an instance I. Since each state has at most n neighbors, there are at most n^T trajectories of the state in T local search iterations.

Fix a trajectory $S_0, S_1, \ldots, S_T$, where S_i denote the state after iteration i. We show that the set of parameters $\boldsymbol{\rho}$ lie in the union of $T \cdot n$ half-spaces. For each $t = 1, \ldots, T$, the algorithm selects $S_t \in N(S_{t-1})$ in iteration t if and only if for any $S' \in \{S_{t-1}\} \cup N(S_t) \setminus \{S_t\}$, we have

$$\sum_{i=1}^d \rho_i \cdot c_i(S') \le \sum_{i=1}^d \rho_i \cdot c_i(S_t),$$

which is a half-space with respect to $\boldsymbol{\rho}$. Altogether, there are $T \cdot n$ half-spaces. Combining the half-spaces for n^T different trajectories yields a total number of Tn^{T+1}.

Now, we vary the instance I. Given m instances $I_1, \ldots, I_m$, there are at most mTn^{T+1} half-spaces partitioning the simplex $\Delta = \{\boldsymbol{\rho} \mid \sum_{i=1}^d \rho_i = 1\}$ such that in each region, $\mathcal{A}(\boldsymbol{\rho}, I_i)$ is fixed for any $i \in [m]$. Note that t half-spaces in $\mathbb{R}^d$ partition the space into at most $O(t^d)$ sub-regions. Therefore, we can partition Δ^k into at most $O((mTn^{T+1})^{kd})$ regions such that $u(P, I_i)$ is fixed for any $i \in [m]$. Similar to the proof of Theorem 3.2, this yields a pseudo-dimension bound $\operatorname{Pdim}(\mathcal{U}) = O(kdT \log n)$, thus leading to $\tilde{O}\left(\frac{kdT}{\varepsilon^2}\right)$ sample complexity. □

Similar results also hold for simulated annealing and linear programs. However, the smoothed analysis technique cannot be directly extended to 3 or more objectives. The reason is that in the 2-objective case, we can directly bound the probability that a discontinuity point lies in a length-l interval by $O(\operatorname{poly}(\kappa) \cdot l)$. However, this approach does not work for a multi-dimensional space. Even if a set of half-spaces are close to each other, the (hyper-)volume of the union of half-spaces can be large. We left the smoothed analysis of multi-dimensional parameter spaces to future work.

E. Details of Experiments

In this section, we present the implementation details of our experiments in Section 7.

Local search. We consider the VLSI circuit partitioning problem for 3D chips (Hu et al., 2022). Given a circuit netlist (represented by a hypergraph), we aim to partition the netlist into two parts. The first and second parts correspond to the top and bottom dies, respectively. If a net (i.e., a hyperedge in the hypergraph) crosses both dies, a cross-die terminal is required to connect both dies. We need to minimize the number of terminals (corresponding to the routability and the cost of the chip), as well as the total wirelength after circuit placement (corresponding to the performance of the chip).

A direct method for circuit partitioning is local search. We initialize an arbitrary partition, and improve the solution by locally moving a cell (i.e., a node in the hypergraph) from the current die to the other. We optimize a weighted sum of both objectives, where the number of terminals can be directly computed and the total wirelength is evaluated by running a VLSI placer. Notice that VLSI placers (e.g., Lin et al. (2019)) are randomized algorithms, and the outcomes of the

wirelength naturally contain random perturbations. This shows that the smoothed analysis setting is indeed reasonable. In our experiments, we fix the random seed of the placer to make the placement result deterministic.

We evaluate the generalization guarantee for local search on ICCAD-22 benchmark (Hu et al., 2022). Since the benchmark consists of only several testcases, we randomly sample subgraphs of the original circuit to obtain instances with various sizes.

Markov decision processes. We consider a resource-gathering Markov decision process in previous work on multi-objective reinforcement learning (Barrett & Narayanan, 2008). The environment is a grid graph with a home base. The agent starts from the home base, gathers resources in the environment, and takes them back to the home. There are multiple kinds of resources, R1, R2, etc. If the agent reaches a grid node with resource Ri, it picks up one unit of Ri. The agent can simultaneously carry at most one unit for each kind of resource. When the agent reaches the home base, it gets one reward for each kind of resource it carries. Some grid nodes have enemies. If the agent passes a node with enemies, with a particular probability, it receives a penalty, loses the resources it carries, and is reset to the home base. The objective is to maximize the reward and minimize the penalty.

We set the discount factor $\gamma = 0.9$, and find the optimal policy by optimizing a weighted sum of the reward and the penalty. We randomly synthesize the environment with different sizes to test the generalization ability. We synthesize 100 instances for each problem size (since finding all pieces is time-consuming), with 50 being the training set, and the rest being the test set.

For both local search and MDP, we compute the number of pieces in the piecewise-constant structure. To compute this number, starting from $\rho = 0$, we use binary search to find the next value of ρ such that the algorithm outcome varies until $\rho = 1$.

Shallow-light trees. The open-source code of SALT (Chen & Young, 2020) and the ICCAD-15 benchmark (Kim & Hu, 2015) for timing-driven VLSI placement are used. The ICCAD-15 benchmark consists of 9.9×10^6 shallow-light tree instances, in which about 4.8×10^4 instances are large-degree instances, i.e., with more than 32 nodes. The experiments are conducted on large-degree instances, since the tradeoff between shallowness and lightness is not significant for small-degree nets. We randomly select 80% nets as the training set and the rest as the test set. We learn a set of parameter values ρ to optimize the Pareto volume of shallowness and lightness.

Let α and β denote the shallowness and lightness of a solution. Since the shallow-light tree is a minimization problem, we define the Pareto volume as the area of domination for $(\alpha_{\mathsf{FLUTE}} - \alpha, \beta_{\mathsf{CL}} - \beta)$, where α_{FLUTE} and β_{CL} are baselines for single-objective Steiner trees in the original paper (Chen & Young, 2020).