Refined Bounds for Algorithm Configuration: The Knife-edge of Dual Class Approximability

Maria-Florina Balcan [1] **Tuomas Sandholm** [1 2 3 4] **Ellen Vitercik** [1]

Abstract

Automating algorithm configuration is growing increasingly necessary as algorithms come with more and more tunable parameters. It is common to tune parameters using machine learning, optimizing algorithmic performance (runtime or solution quality, for example) using a *training set* of problem instances from the specific domain at hand. We investigate a fundamental question about these techniques: how large should the training set be to ensure that a parameter's average empirical performance over the training set is close to its expected, future performance? We answer this question for algorithm configuration problems that exhibit a widely-applicable structure: the algorithm's performance as a function of its parameters can be approximated by a "simple" function. We show that if this approximation holds under the L^∞-norm, we can provide strong sample complexity bounds, but if the approximation holds only under the L^p-norm for $p < \infty$, it is not possible to provide meaningful sample complexity bounds in the worst case. We empirically evaluate our bounds in the context of integer programming, obtaining sample complexity bounds that are up to 700 times smaller than the previously best-known bounds (Balcan et al., 2018a).

1. Introduction

Algorithms typically have tunable parameters that significantly impact their performance, measured in terms of runtime, solution quality, and so on. Machine learning is often used to automate parameter tuning (Horvitz et al., 2001; Hutter et al., 2009; Kadioglu et al., 2010; Sandholm, 2013): given a *training set* of problem instances from the application domain at hand, this automated procedure returns a parameter setting that will ideally perform well on future, unseen instances.

It is important to be careful when using this automated approach: if the training set is too small, a parameter setting with strong average empirical performance over the training set may have poor future performance on unseen instances. *Generalization bounds* provide guidance when it comes to selecting the training set size. They bound the difference between an algorithm's performance on average over the training set (drawn from an unknown, application-specific distribution) and its expected performance on unseen instances. These bounds can be used to evaluate a parameter setting returned by any black-box procedure: they bound the difference between that parameter's average performance on the training set and its expected performance.

At a high level, we provide generalization bounds that hold when an algorithm's performance as a function of its parameters exhibits a widely-applicable structure: it can be approximated by a "simple" function. We prove that it is possible to provide strong generalization bounds when the approximation holds under the L^∞-norm. Meanwhile, it is not possible to provide strong guarantees in the worst-case if the approximation only holds under the L^p-norm for $p < \infty$. Therefore, this connection between learnability and approximability is balanced on a knife-edge.

Our analysis is based on structure exhibited by *primal* and *dual* functions (Assouad, 1983), which we now describe at a high level. To provide generalization bounds, a common strategy is to bound the *intrinsic complexity* of the following function class $\mathcal{F}$: for every parameter vector $\boldsymbol{r}$ (such as a CPLEX parameter setting) there is a function $f_{\boldsymbol{r}} \in \mathcal{F}$ that takes as input a problem instance x (such as an integer program) and returns $f_{\boldsymbol{r}}(x)$, the algorithm's *performance* on input x when parameterized by $\boldsymbol{r}$. Performance is measured by runtime, solution quality, or some other metric. The functions $f_{\boldsymbol{r}}$ are called *primal functions*.

The class $\mathcal{F}$ is gnarly: in the case of integer programming algorithm configuration, the domain of every function in $\mathcal{F}$

[1]Computer Science Department, Carnegie Mellon University [2]Strategy Robot, Inc. [3]Optimized Markets, Inc. [4]Strategic Machine, Inc.. Correspondence to: Maria-Florina Balcan <ninamf@cs.cmu.edu>, Tuomas Sandholm <sandholm@cs.cmu.edu>, Ellen Vitercik <vitercik@cs.cmu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

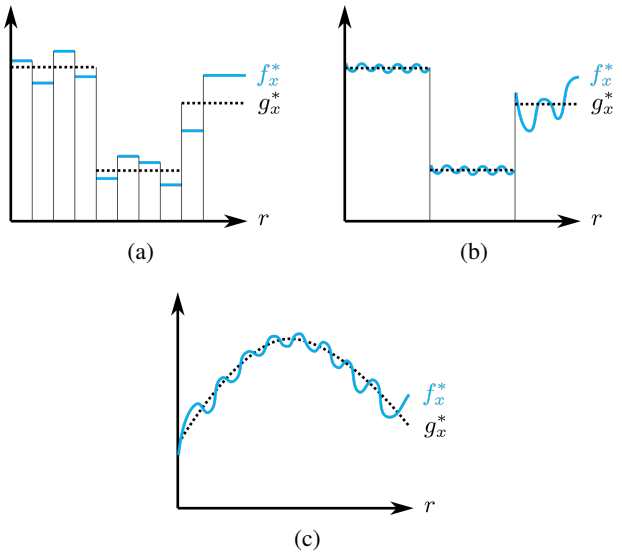

Figure 1. Examples of dual functions $f_x^* : \mathbb{R} \to \mathbb{R}$ (solid blue lines) which are approximated by simpler functions g_x^* (dotted black lines).

consists of integer programs, so it is unclear how to visualize or plot these functions, and there are no obvious notions of Lipschitzness or smoothness to rely on. Rather than fixing a parameter setting $\boldsymbol{r}$ and varying the input x (as under the function $f_{\boldsymbol{r}}$), it can be enlightening to instead fix the input x and analyze the algorithm's performance as a function of $\boldsymbol{r}$. This *dual function* is denoted $f_x^*(\boldsymbol{r})$. The dual functions have a simple, Euclidean domain, they are typically easy to plot, and they often have ample structure we can use to bound the intrinsic complexity of the class $\mathcal{F}$.

Our contributions. We observe that for many configuration problems, the dual functions can be closely approximated by "simple" functions, as in Figure 1. This raises the question: can we exploit this structure to provide strong generalization guarantees? We show that if the dual functions are approximated by simple functions under the L^∞-norm (meaning the maximum distance between the functions is small), then we can provide strong generalization guarantees. However, this is no longer true when the approximation only holds under the L^p-norm for $p < \infty$: we present a set of functions whose duals are well-approximated by the simple constant function $g(x) = \frac{1}{2}$ under the L^p-norm (meaning $\sqrt[p]{\int \left|f_x^*(\boldsymbol{r}) - \frac{1}{2}\right|^p \, d\boldsymbol{r}}$ is small), but which are not learnable.

We provide an algorithm that finds approximating simple functions in the following widely-applicable setting: the dual functions are piecewise-constant with a large number of pieces, but can be approximated by simpler piecewise-constant functions with few pieces, as in Figure 1(a). This is the case in our integer programming experiments.

In our experiments, we demonstrate significant practical implications of our analysis. We configure CPLEX, one of the most widely-used integer programming solvers. Integer programming has diverse applications throughout science. Prior research has shown that the dual functions associated with various CPLEX parameters are piecewise constant and has provided generalization bounds that grow with the number of pieces (Balcan et al., 2018a). However, the number of pieces can be so large that these bounds can be quite loose. We show that these dual functions can be approximated under the L^∞-norm by simple functions (as in Figure 1(a)), so our theoretical results imply strong generalization guarantees. In our experiments, we demonstrate that in order to obtain the same generalization bound, the training set size required under our analysis is up to 700 times smaller than that of Balcan et al. (2018a). Improved sample complexity guarantees imply faster learning algorithms, since the learning algorithm needs to analyze fewer training instances.

Related research. In algorithm configuration, several papers have provided generalization guarantees for specific algorithm families, including greedy algorithms (Gupta and Roughgarden, 2017; Balcan et al., 2018b), clustering algorithms (Balcan et al., 2017; 2018c; 2020a), and integer programming algorithms (Balcan et al., 2018a). In contrast, we provide general guarantees that apply to any configuration problem that satisfies a widely-applicable structure: the dual functions are approximately simple. A strength of our results is that they are not tied to any specific algorithm family, though we show that our guarantees can be empirically much stronger than the best-known bounds. Balcan et al. (2019) show that if the dual functions are simple—for example, they are piecewise-constant with few pieces—then it is possible to provide strong generalization bounds. We observe, however, that often the dual functions themselves are not particularly simple, but can be approximated by simple functions. We exploit this structure to provide more general guarantees. The analysis tools from prior research do not apply to this more general structure, so we require new, refined proof techniques.

Our guarantees are configuration-procedure-agnostic: no matter how one tunes the parameters using the training set, we bound the difference between the resulting parameter setting's performance on average over the training set and its expected performance on unseen instances. A related line of research has provided learning-based algorithm configuration procedures with provable guarantees (Kleinberg et al., 2017; 2019; Weisz et al., 2018; 2019; Balcan et al., 2020b). Unlike the results in this paper, their guarantees are not configuration-procedure-agnostic: they apply to the specific configuration procedures they propose. Moreover, their procedures only apply to finding configurations that minimize computational resource usage, such as runtime,

whereas the guarantees in this paper apply to more general measures of algorithmic performance, such as solution quality.

A related line of research has studied integer programming algorithm configuration (Hutter et al., 2009; Sabharwal et al., 2012; Sandholm, 2013; He et al., 2014; Balafrej et al., 2015; Khalil et al., 2016; 2017; Di Liberto et al., 2016; Lodi and Zarpellon, 2017; Alvarez et al., 2017; Kruber et al., 2017; Balcan et al., 2018a), as do we, though our results apply more generally. The results in these papers are primarily empirical, with the exception of the paper by Balcan et al. (2018a), with which we compare extensively in Section 4.2.

2. Notation and Background

We study functions that map an abstract domain $\mathcal{X}$ to $[0, 1]$. We denote the set of all such functions as $[0, 1]^{\mathcal{X}}$. The learning algorithms we analyze have sample access to an unknown distribution $\mathcal{D}$ over examples $x \in \mathcal{X}$ and aim to find a function $f \in \mathcal{F}$ with small expected value $\mathbb{E}_{x\sim\mathcal{D}}[f(x)]$.

2.1. Problem Definition

We provide *generalization guarantees*, which bound the difference between the expected value $\mathbb{E}_{x\sim\mathcal{D}}[f(x)]$ of any function $f \in \mathcal{F}$ and its empirical average value $\frac{1}{N}\sum_{i=1}^{N} f(x_i)$ over a training set $x_1, \ldots, x_N \sim \mathcal{D}$. We focus on functions that are parameterized by a set of vectors $\mathcal{R} \subseteq \mathbb{R}^d$. Given a vector $\boldsymbol{r} \in \mathcal{R}$, we denote the corresponding function as $f_{\boldsymbol{r}} : \mathcal{X} \to [0, 1]$, and we define $\mathcal{F} = \{f_{\boldsymbol{r}} \mid \boldsymbol{r} \in \mathcal{R}\}$.

Generalization guarantees are particularly useful for analyzing the expected performance of empirical risk minimization learning algorithms for the following reason. Suppose we know that for any function $f \in \mathcal{F}$, $\left|\frac{1}{N}\sum_{i=1}^{N} f(x_i) - \mathbb{E}_{x\sim\mathcal{D}}[f(x)]\right| \leq \epsilon$. Let $\hat{f}$ be the function in $\mathcal{F}$ with smallest average value over the training set: $\hat{f} = \text{argmin}_{f\in\mathcal{F}} \sum_{i=1}^{N} f(x_i)$. Then $\hat{f}$ has nearly optimal expected value: $\mathbb{E}_{x\sim\mathcal{D}}\left[\hat{f}(x)\right] - \min_{f\in\mathcal{F}} \mathbb{E}_{x\sim\mathcal{D}}[f(x)] \leq 2\epsilon$.

2.2. Integer Programming Algorithm Configuration

We use integer programming algorithm configuration as a running example, though our results are much more general. An *integer program (IP)* is defined by a matrix $A \in \mathbb{R}^{m\times n}$, a constraint vector $\boldsymbol{b} \in \mathbb{R}^m$, an objective vector $\boldsymbol{c} \in \mathbb{R}^n$, and a set of indices $I \subseteq [n]$. The goal is to find a vector $\boldsymbol{z} \in \mathbb{R}^n$ such that $\boldsymbol{c} \cdot \boldsymbol{z}$ is maximized, subject to the constraints that $A\boldsymbol{z} \leq \boldsymbol{b}$ and for every index $i \in I$, $z_i \in \{0, 1\}$.

In our experiments, we tune the parameters of branch-and-bound (B&B) (Land and Doig, 1960), the most widely-used algorithm for solving IPs. It is used under the hood by commercial solvers such as CPLEX and Gurobi. We provide a brief, high-level overview of B&B, and refer the reader to the textbook by Nemhauser and Wolsey (1999) for more details. B&B builds a search tree to solve an input IP x. At the tree's root is the original IP x. At each round, B&B chooses a leaf of the search tree, which represents an IP x'. It does so using a *node selection policy*; common choices include depth- and best-first search. Then, it chooses an index $i \in I$ using a *variable selection policy*. It next *branches* on z_i: it sets the left child of x' to be that same integer program x', but with the additional constraint that $z_i = 0$, and it sets the right child of x' to be that same integer program, but with the additional constraint that $z_i = 1$. It solves both LP relaxations, and if either solution satisfies the integrality constraints of the original IP x, it constitutes a feasible solution to x. B&B *fathoms* a leaf—which means that it never will branch on that leaf—if it can guarantee that the optimal solution does not lie along that path. B&B terminates when it has fathomed every leaf. At that point, we can guarantee that the best solution to x found so far is optimal. In our experiments, we tune the parameters of the variable selection policy, which we describe in more detail in Section 4.2.

In this setting, $\mathcal{X}$ is a set of IPs and the functions in $\mathcal{F}$ are parameterized by CPLEX parameter vectors $\boldsymbol{r} \in \mathbb{R}^d$, denoted $\mathcal{F} = \left\{f_{\boldsymbol{r}} \mid \boldsymbol{r} \in \mathbb{R}^d\right\}$. In keeping with prior work (Balcan et al., 2018a), $f_{\boldsymbol{r}}(x)$ equals the size of the B&B tree CPLEX builds given the parameter setting $\boldsymbol{r}$ and input IP x, normalized to fall in $[0, 1]$. The learning algorithms we study take as input a training set of IPs sampled from $\mathcal{D}$ and return a parameter vector. Since our goal is to minimize tree size, ideally, the size of the trees CPLEX builds using that parameter setting should be small in expectation over $\mathcal{D}$.

3. Dual Functions

Our goal is to provide generalization guarantees for the function class $\mathcal{F} = \{f_{\boldsymbol{r}} \mid \boldsymbol{r} \in \mathcal{R}\}$. To do so, we use structure exhibited by the *dual function class*. Every function in the dual class is defined by an element $x \in \mathcal{X}$, denoted $f_x^* : \mathcal{R} \to [0, 1]$. Naturally, $f_x^*(\boldsymbol{r}) = f_{\boldsymbol{r}}(x)$. The dual class $\mathcal{F}^* = \{f_x^* \mid x \in \mathcal{X}\}$ is the set of all dual functions.

The dual functions are intuitive in our integer programming example. For any IP x, the dual function f_x^* measures the size of the tree CPLEX builds (normalized to lie in the interval $[0, 1]$) when given x as input, as a function of the CPLEX parameters. Duals are also straightforward in more abstract settings: if $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{F}$ is the set of linear functions $f_{\boldsymbol{r}}(\boldsymbol{x}) = \boldsymbol{r} \cdot \boldsymbol{x}$, each dual function $f_{\boldsymbol{x}}^*(\boldsymbol{r}) = \boldsymbol{r} \cdot \boldsymbol{x}$ is also linear. When $\mathcal{F}$ consists of the constant functions $f_{\boldsymbol{r}}(x) = \boldsymbol{r}$, each dual function is the identity function $f_x^*(\boldsymbol{r}) = \boldsymbol{r}$.

Prior research shows that when the dual functions are simple—for example, they are piecewise-constant with a

small number of pieces—it is possible to provide strong generalization bounds (Balcan et al., 2019). In many settings, however, we find that the dual functions themselves are not simple, but are approximated by simple functions, as in Figure 1. We formally define this concept as follows.

Definition 3.1 ((γ, p)-approximate)**.** Let $\mathcal{F} = \{f_{\boldsymbol{r}} \mid \boldsymbol{r} \in \mathcal{R}\}$ and $\mathcal{G} = \{g_{\boldsymbol{r}} \mid \boldsymbol{r} \in \mathcal{R}\}$ be two sets of functions mapping $\mathcal{X}$ to $[0,1]$. We assume that all dual functions f_x^* and g_x^* are integrable over the domain $\mathcal{R}$. We say that the dual class $\mathcal{G}^*$ (γ, p)-*approximates* the dual class $\mathcal{F}^*$ if for every element x, the distance between the functions f_x^* and g_x^* is at most γ under the L^p-norm. For $p \in [1, \infty)$, this means that $\|f_x^* - g_x^*\|_p := \sqrt[p]{\int_{\mathcal{R}} |f_x^*(\boldsymbol{r}) - g_x^*(\boldsymbol{r})|^p \, d\boldsymbol{r}} \leq \gamma$ and when $p = \infty$, this means that $\|f_x^* - g_x^*\|_\infty := \sup_{\boldsymbol{r} \in \mathcal{R}} |f_x^*(\boldsymbol{r}) - g_x^*(\boldsymbol{r})| \leq \gamma$.

4. Learnability and Approximability

In this section, we investigate the connection between learnability and approximability. In Section 4.1, we prove that when the dual functions are approximable under the L_∞-norm by simple functions, we can provide strong generalization bounds. In Section 4.2, we empirically evaluate these improved guarantees in the context of integer programming. Finally, in Section 4.3, we prove that it is not possible to provide non-trivial generalization guarantees (in the worst case) when the norm under which the dual functions are approximable is the L_p-norm for $p < \infty$.

4.1. Data-dependent Generalization Guarantees

We now show that if the dual class $\mathcal{F}^*$ is (γ, ∞)-approximated by the dual of a "simple" function class $\mathcal{G}$, we can provide strong generalization bounds for the class $\mathcal{F}$. There are many different tools for measuring how "simple" a function class is. We use *Rademacher complexity* (Koltchinskii, 2001), which intuitively measures the extent to which functions in $\mathcal{F}$ match random noise vectors $\boldsymbol{\sigma} \in \{-1, 1\}^N$.

Definition 4.1 (Rademacher complexity)**.** The *empirical Rademacher complexity* of a function class $\mathcal{F} = \{f_{\boldsymbol{r}} \mid \boldsymbol{r} \in \mathcal{R}\}$ given a set $\mathcal{S} = \{x_1, \ldots, x_N\} \subseteq \mathcal{X}$ is $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F}) = \frac{1}{N} \mathbb{E}_{\boldsymbol{\sigma} \sim \{-1,1\}^N} \left[\sup_{\boldsymbol{r} \in \mathcal{R}} \sum_{i=1}^N \sigma_i f_{\boldsymbol{r}}(x_i)\right]$, where each σ_i equals -1 or 1 with equal probability.

The summation $\sum_{i=1}^N \sigma_i f_{\boldsymbol{r}}(x_i)$ measures the correlation between the vector $(f_{\boldsymbol{r}}(x_1), \ldots, f_{\boldsymbol{r}}(x_N))$ and the random noise vector $\boldsymbol{\sigma}$. By taking the supremum over all parameter vectors $\boldsymbol{r} \in \mathcal{R}$, we measure how well functions in the class $\mathcal{F}$ correlate with $\boldsymbol{\sigma}$ over the sample $\mathcal{S}$. Therefore, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F})$ measures how well functions in the class $\mathcal{F}$ correlate with random noise on average over $\mathcal{S}$. Rademacher complexity thus provides a way to measure the intrinsic complexity of $\mathcal{F}$ because the more complex the class $\mathcal{F}$ is, the better its functions can correlate with random noise. For example, if the class $\mathcal{F}$ consists of just a single function, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F}) = 0$. At the other extreme, if $\mathcal{X} = [0,1]$ and $\mathcal{F} = [0,1]^{[0,1]}$, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F}) = \frac{1}{2}$.

Classic learning-theoretic results provide guarantees based on Rademacher complexity, such as the following.

Theorem 4.2 (e.g., Mohri et al. (2012))**.** *For any $\delta \in (0,1)$, with probability $1 - \delta$ over the draw of N samples $\mathcal{S} = \{x_1, \ldots, x_N\} \sim \mathcal{D}^N$, for all functions $f_{\boldsymbol{r}} \in \mathcal{F}$, $\left|\frac{1}{N}\sum_{i=1}^N f_{\boldsymbol{r}}(x_i) - \mathbb{E}[f_{\boldsymbol{r}}(x)]\right| = \tilde{O}\left(\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F}) + \sqrt{\frac{1}{N}}\right)$ (the dependence on δ is logarithmic.)*

Theorem 4.2 is a *generalization guarantee* because it measures the extent to which a function's empirical average value over the samples generalizes to its expected value.

Ideally, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F})$ converges to zero as the sample size N grows so the bound in Theorem 4.2 also converges to zero. If the class $\mathcal{F}$ consists of just a single function, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F}) = 0$, and Theorem 4.2 recovers Hoeffding's bound. If, for example, $\mathcal{X} = [0,1]$ and $\mathcal{F} = [0,1]^{[0,1]}$, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F}) = \frac{1}{2}$, and the bound from Theorem 4.2 is meaningless.

We show that if the dual class $\mathcal{F}^*$ is (γ, ∞)-approximated by the dual of a class $\mathcal{G}$ with small Rademacher complexity, then the Rademacher complexity of $\mathcal{F}$ is also small. The full proof of the following theorem in Appendix B.1.

Theorem 4.3. *Let $\mathcal{F} = \{f_{\boldsymbol{r}} \mid \boldsymbol{r} \in \mathcal{R}\}$ and $\mathcal{G} = \{g_{\boldsymbol{r}} \mid \boldsymbol{r} \in \mathcal{R}\}$ consist of functions mapping $\mathcal{X}$ to $[0,1]$. For any $\mathcal{S} \subseteq \mathcal{X}$, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F}) \leq \widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{G}) + \frac{1}{|\mathcal{S}|}\sum_{x \in \mathcal{S}} \|f_x^* - g_x^*\|_\infty$.*

Proof sketch. To prove this theorem, we use the fact that for any parameter vector $\boldsymbol{r} \in \mathcal{R}$, any element $x \in \mathcal{X}$, and any binary value $\sigma \in \{-1, 1\}$, $\sigma f_{\boldsymbol{r}}(x) = \sigma f_x^*(\boldsymbol{r}) \leq \sigma g_x^*(\boldsymbol{r}) + \|f_x^* - g_x^*\|_\infty = \sigma g_{\boldsymbol{r}}(x) + \|f_x^* - g_x^*\|_\infty$. □

If the class $\mathcal{G}^*$ (γ, ∞)-approximates the class $\mathcal{F}^*$, then $\frac{1}{|\mathcal{S}|}\sum_{x \in \mathcal{S}} \|f_x^* - g_x^*\|_\infty$ is at most γ. If this term is smaller than γ for most sets $\mathcal{S} \sim \mathcal{D}^N$, then the bound on $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F})$ in Theorem 4.3 will often be even better than $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{G}) + \gamma$.

Theorems 4.2 and 4.3 imply that with probability $1 - \delta$ over the draw of the set $\mathcal{S} \sim \mathcal{D}^N$, for all parameter vectors $\boldsymbol{r} \in \mathcal{R}$, the difference between the empirical average value of $f_{\boldsymbol{r}}$ over $\mathcal{S}$ and its expected value is at most $\tilde{O}\left(\frac{1}{N}\sum_{x \in \mathcal{S}} \|f_x^* - g_x^*\|_\infty + \widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{G}) + \sqrt{\frac{1}{N}}\right)$. In our integer programming experiments, we show that this data-dependent generalization guarantee can be much tighter than the best-known worst-case guarantee.

Algorithm for finding approximating functions. We provide a dynamic programming (DP) algorithm (Algo-

rithm 1 in Appendix B.2) for the widely-applicable case where the dual functions f_x^* are piecewise constant with a large number of pieces. Given an integer k, the algorithm returns a piecewise-constant function g_x^* with at most k pieces such that $\|f_x^* - g_x^*\|_\infty$ is minimized, as in Figure 1(a). Letting t be the number of pieces in the piecewise decomposition of f_x^*, the DP algorithm runs in $O\left(kt^2\right)$ time. As we describe in Section 4.2, when k and $\|f_x^* - g_x^*\|_\infty$ are small, Theorem 4.3 implies strong guarantees. We use this DP algorithm in our integer programming experiments.

Structural risk minimization. Theorem 4.3 illustrates a fundamental tradeoff in machine learning. The simpler the class $\mathcal{G}$, the smaller its Rademacher complexity, but—broadly speaking—the worse functions from its dual will be at approximating functions in $\mathcal{F}^*$. In other words, the simpler $\mathcal{G}$ is, the worse the approximation $\frac{1}{N}\sum_{x\in\mathcal{S}}\|f_x^* - g_x^*\|_\infty$ will likely be. Therefore, there is a tradeoff between generalizability and approximability. It may not be *a priori* clear how to balance this tradeoff. *Structural risk minimization (SRM)* is a classic, well-studied approach for optimizing tradeoffs between complexity and generalizability which we use in our experiments.

Our SRM approach is based on the following corollary of Theorem 4.3. Let $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \ldots$ be a countable sequence of function classes where each $\mathcal{G}_j = \{g_{j,\boldsymbol{r}} \mid \boldsymbol{r} \in \mathcal{R}\}$ is a set of functions mapping $\mathcal{X}$ to $[0,1]$. We use the notation $g_{j,x}^*$ to denote the duals of the functions in $\mathcal{G}_j$, so $g_{j,x}^*(\boldsymbol{r}) = g_{j,\boldsymbol{r}}(x)$.

Corollary 4.4. *With probability $1-\delta$ over the draw of the set $\mathcal{S} \sim \mathcal{D}^N$, for all $\boldsymbol{r} \in \mathcal{R}$ and all $j \geq 1$,*

$$\left|\frac{1}{N}\sum_{x\in\mathcal{S}} f_{\boldsymbol{r}}(x) - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[f_{\boldsymbol{r}}(x)\right]\right| = \tilde{O}\left(\frac{1}{N}\sum_{x\in\mathcal{S}}\left\|f_x^* - g_{j,x}^*\right\|_\infty + \widehat{\mathscr{R}}_{\mathcal{S}}\left(\mathcal{G}_j\right) + \sqrt{\frac{1}{N}}\right). \quad (1)$$

The proof of this corollary is in Appendix B.1.

In our experiments, each dual class $\mathcal{G}_j^*$ consists of piecewise-constant functions with at most j pieces. This means that as j grows, the class $\mathcal{G}_j^*$ becomes more complex, or in other words, the Rademacher complexity $\widehat{\mathscr{R}}_{\mathcal{S}}\left(\mathcal{G}_j\right)$ also grows. Meanwhile, the more pieces a piecewise-constant function g_x^* has, the better it is able to approximate the dual function f_x^*. In other words, as j grows, the approximation term $\frac{1}{N}\sum_{x\in\mathcal{S}}\left\|f_x^* - g_{j,x}^*\right\|_\infty$ shrinks. SRM is the process of finding the level j in the nested hierarchy that minimizes the sum of these two terms, and therefore obtains the best generalization guarantee via Equation (1).

Remark 4.5. We conclude by noting that the empirical average $\frac{1}{N}\sum_{x\in\mathcal{S}}\left\|f_x^* - g_{j,x}^*\right\|_\infty$ in Equation (1) can be replaced by the expectation $\mathbb{E}_{x\sim\mathcal{D}}\left[\left\|f_x^* - g_{j,x}^*\right\|_\infty\right]$. See Corollary B.1 in Appendix B.1 for the proof.

4.2. Improved Integer Programming Guarantees

In this section, we demonstrate that our data-dependent generalization guarantees from Section 4.1 can be much tighter than worst-case generalization guarantees provided in prior research. We demonstrate these improvements in the context of integer programming algorithm configuration, which we introduced in Section 2.2. Our formal model is the same as that of Balcan et al. (2018a), who studied worst-case generalization guarantees. Each element of the set $\mathcal{X}$ is an IP. The set $\mathcal{R}$ consists of CPLEX parameter settings. We assume there is an upper bound κ on the size of the largest tree we allow B&B to build before we terminate, as in prior research (Hutter et al., 2009; Kleinberg et al., 2017; Balcan et al., 2018a; Kleinberg et al., 2019). In Appendix B.2, we describe our methodology for choosing κ. Given a parameter setting r and an IP x, we define $f_r(x)$ to be the size of the tree CPLEX builds, capped at κ, divided by κ (this way, $f_r(x) \in [0,1]$). We define the set $\mathcal{F} = \{f_r \mid r \in \mathcal{R}\}$.

We tune the parameter of B&B's variable selection policy (VSP). We described the purpose of VSPs in Section 2.2. We study *score-based VSPs*, defined as follows. Let $\texttt{score}$ be a function that takes as input a partial B&B tree $\mathcal{T}$, a leaf of $\mathcal{T}$ representing an IP x, and an index $i \in [n]$, and returns a real-valued $\texttt{score}(\mathcal{T}, x, i)$. Let V be the set of variables that have not been branched on along the path from the root of $\mathcal{T}$ to x. A *score-based VSP* branches on the variable $\text{argmax}_{z_i\in V}\{\texttt{score}(\mathcal{T}, x, i)\}$ at the node x.

We study how to learn a high-performing convex combination of any two scoring rules. We focus on four scoring rules in our experiments. To define them, we first introduce some notation. For an IP x with objective function $\boldsymbol{c}\cdot\boldsymbol{z}$, we denote an optimal solution to the LP relaxation of x as $\breve{\boldsymbol{z}}_x = (\breve{z}_{x,1}, \ldots \breve{z}_{x,n})$. We also use the notation $\breve{c}_x = \boldsymbol{c}\cdot\breve{\boldsymbol{z}}_x$. Finally, we use the notation x_i^+ (resp., x_i^-) to denote the IP x with the additional constraint that $z_i = 1$ (resp., $z_i = 0$).[1]

We study four scoring rules $\texttt{score}_L$, $\texttt{score}_S$, $\texttt{score}_A$, and $\texttt{score}_P$:

- $\texttt{score}_L(\mathcal{T}, x, i) = \max\left\{\breve{c}_x - \breve{c}_{x_i^+}, \breve{c}_x - \breve{c}_{x_i^-}\right\}$. Under $\texttt{score}_L$, B&B branches on the variable leading to the **L**argest change in the LP objective value.
- $\texttt{score}_S(\mathcal{T}, x, i) = \min\left\{\breve{c}_x - \breve{c}_{x_i^+}, \breve{c}_x - \breve{c}_{x_i^-}\right\}$. Under $\texttt{score}_S$, B&B branches on the variable leading to the **S**mallest change.

[1] If x_i^+ (resp., x_i^-) is infeasible, then we define $\breve{c}_x - \breve{c}_{x_i^+}$ (resp., $\breve{c}_x - \breve{c}_{x_i^-}$) to be some large number greater than $||\boldsymbol{c}||_1$.

- $\texttt{score}_A(\mathcal{T}, x, i) = \frac{1}{6}\texttt{score}_L(\mathcal{T}, x, i) + \frac{5}{6}\texttt{score}_S(\mathcal{T}, x, i)$. This is a scoring rule that Achterberg (2009) recommended. It balances the optimistic approach to branching under $\texttt{score}_L$ with the pessimistic approach under $\texttt{score}_S$.
- $\texttt{score}_P(\mathcal{T}, x, i) = \max\left\{\breve{c}_x - \breve{c}_{x_i^+}, 10^{-6}\right\} \cdot \max\left\{\breve{c}_x - \breve{c}_{x_i^-}, 10^{-6}\right\}$. This is known as the ***Product scoring rule***. Comparing $\breve{c}_x - \breve{c}_{x_i^-}$ and $\breve{c}_x - \breve{c}_{x_i^+}$ to 10^{-6} allows the algorithm to compare two variables even if $\breve{c}_x - \breve{c}_{x_i^-} = 0$ or $\breve{c}_x - \breve{c}_{x_i^+} = 0$. After all, suppose the scoring rule simply calculated the product $\left(\breve{c}_x - \breve{c}_{x_i^-}\right) \cdot \left(\breve{c}_x - \breve{c}_{x_i^+}\right)$ without comparing to 10^{-6}. If $\breve{c}_x - \breve{c}_{x_i^-} = 0$, then the score equals 0, canceling out the value of $\breve{c}_x - \breve{c}_{x_i^+}$ and thus losing the information encoded by this difference.

Fix any two scoring rules $\texttt{score}_1$ and $\texttt{score}_2$. We define $f_r(x)$ to be the size of the tree B&B builds (normalized to lie in $[0,1]$) when it uses the score-based VSP defined by $(1-r)\texttt{score}_1 + r\texttt{score}_2$. Our goal is to learn the best convex combination of the two scoring rules. When $\texttt{score}_1 = \texttt{score}_L$ and $\texttt{score}_2 = \texttt{score}_S$, prior research has proposed several alternative settings for the parameter r (Gauthier and Ribière, 1977; Bénichou et al., 1971; Beale, 1979; Linderoth and Savelsbergh, 1999; Achterberg, 2009), though no one setting is optimal across all applications. Balcan et al. (2018a) prove the following lemma about the structure of the functions f_x^*.

Lemma 4.6. *For any IP x with n variables, the dual function f_x^* is piecewise-constant with at most $n^{2(\kappa+1)}$ pieces.*

Lemma 4.6 implies the following worst-case bound on $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F})$. See Lemma B.4 in Appendix B.2 for the proof.

Corollary 4.7. *For any set $\mathcal{S} \subseteq \mathcal{X}$ of integer programs, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{F}) \leq \sqrt{\frac{2\ln\left(|\mathcal{S}|\left(n^{2(\kappa+1)}-1\right)+1\right)}{|\mathcal{S}|}}$.*

This corollary and Theorem 4.2 imply the following worst-case generalization bound: with probability $1-\delta$ over the draw of N samples $\mathcal{S} \sim \mathcal{D}^N$, for all $r \in [0,1]$, $\left|\frac{1}{N}\sum_{x\in\mathcal{S}} f_r(x) - \mathbb{E}_{x\sim\mathcal{D}}\left[f_r(x)\right]\right|$ is bounded above by

$$2\sqrt{\frac{2\ln(N\left(n^{2(\kappa+1)}-1\right)+1)}{N}} + 3\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}. \quad (2)$$

This worst-case bound can be large when κ is large. We find that although the duals f_x^* are piecewise-constant with many pieces, they can be approximated piecewise-constant functions with few pieces, as in Figure 1(a). As a result, we improve over Equation (2) via Theorem 4.3, our data-dependent bound.

To make use of Theorem 4.3, we now formally define the function class whose dual (γ,∞)-approximates $\mathcal{F}^*$. We first define the dual class, then the primal class. To this end, fix some integer $j \geq 1$ and let $\mathcal{H}_j$ be the set of all piecewise-constant functions mapping $[0,1]$ to $[0,1]$ with at most j pieces. For every IP x, we define $g_{j,x}^* \in \operatorname{argmin}_{h\in\mathcal{H}_j}\|f_x^* - h\|_\infty$, breaking ties in some fixed but arbitrary manner. The function $g_{j,x}^*$ can be found via dynamic programming; see Algorithm 1 in Appendix B.2. We define the dual class $\mathcal{G}_j^* = \left\{g_{j,x}^* \mid x \in \mathcal{X}\right\}$. Therefore, the dual class $\mathcal{G}_j^*$ is consists of piecewise-constant functions with at most j pieces. In keeping with the definition of primal and dual functions from Section 3, for every parameter $r \in [0,1]$ and IP x, we define $g_{j,r}(x) = g_{j,x}^*(r)$. Finally, we define the primal class $\mathcal{G}_j = \{g_{j,r} \mid r \in [0,1]\}$.

To apply our results from Section 4.1, we must bound the Rademacher complexity of the set $\mathcal{G}_j$. Doing so is simple due to the structure of the dual class $\mathcal{G}_j^*$. The following lemma[2] is a corollary of Lemma B.4 in Appendix B.2.

Lemma 4.8. *For any set $\mathcal{S} \subseteq \mathcal{X}$ of integer programs, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{G}_j) \leq \sqrt{\frac{2\ln(|\mathcal{S}|(j-1)+1)}{|\mathcal{S}|}}$.*

This lemma together with Remark 4.5 and Corollary 4.4 imply that with probability $1-\delta$ over $\mathcal{S} \sim \mathcal{D}^N$, for all parameters $r \in [0,1]$ and $j \geq 1$, $\left|\frac{1}{N}\sum_{x\in\mathcal{S}} f_r(x) - \mathbb{E}_{x\sim\mathcal{D}}\left[f_r(x)\right]\right|$ is upper-bounded by the minimum of Equation (2) and

$$2\gamma_j + 2\sqrt{\frac{2\ln(N(j-1)+1)}{N}} + \sqrt{\frac{2}{N}\ln\frac{2(\pi j)^2}{3\delta}}, \quad (3)$$

where $\gamma_j = \mathbb{E}_{x\sim\mathcal{D}}\left[\left\|f_x^* - g_{j,x}^*\right\|_\infty\right]$. As j grows, $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{G}_j)$ grows, but the dual class $\mathcal{G}_j^*$ is better able to approximate $\mathcal{F}^*$. In our experiments, we optimize this tradeoff between generalizability and approximability.

Experiments. We analyze distributions over IPs formulating the combinatorial auction winner determination problem under the OR-bidding language (Sandholm, 2002), which we generate using the Combinatorial Auction Test Suite (CATS) (Leyton-Brown et al., 2000). We use the "arbitrary" generator with 200 bids and 100 goods, resulting in IPs with 200 variables, and the "regions" generator with 400 bids and 200 goods, resulting in IPs with 400 variables.

We use the algorithm described in Appendix D.1 of the paper by Balcan et al. (2018a) to compute the functions f_x^*. It overrides the default VSP of CPLEX 12.8.0.0 using the C API. We use Algorithm 1 in Appendix B.2 to compute the approximating duals. All experiments were run on a 64-core machine with 512 GB of RAM.

[2]This bound on $\widehat{\mathscr{R}}_{\mathcal{S}}(\mathcal{G}_j)$ could potentially be optimized even further using a data-dependent approach, such as the one summarized by Theorem E.3 in the paper by Balcan et al. (2018a).

$$\min_{j\in[1600]}\left\{2\left(\frac{1}{M}\sum_{i=1}^{M}\left\|f^*_{x_i}-g^*_{j,x_i}\right\|_\infty+\frac{1}{40}\right)+2\sqrt{\frac{2\ln(N(j-1)+1)}{N}}+\sqrt{\frac{2}{N}\ln\frac{(20\pi j)^2}{3}}\right\}. \tag{4}$$

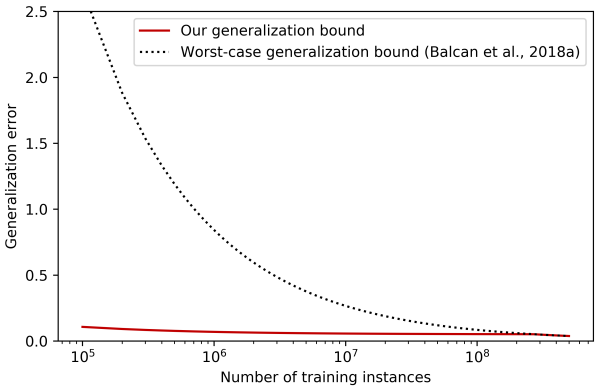

Figure 2. Results using SRM on the CATS "regions" generator with $\texttt{score}_1 = \texttt{score}_L$ and $\texttt{score}_2 = \texttt{score}_S$.

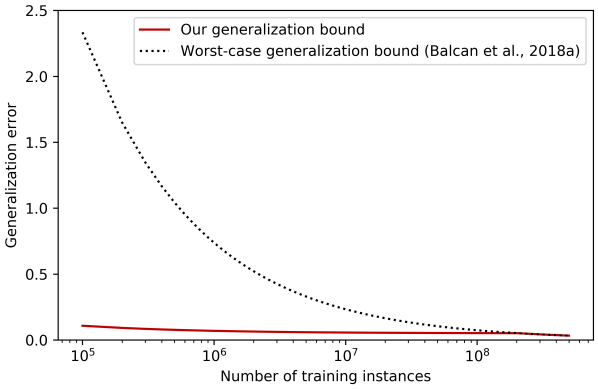

Figure 3. Results using SRM on the CATS "arbitrary" generator with $\texttt{score}_1 = \texttt{score}_L$ and $\texttt{score}_2 = \texttt{score}_S$.

In Figures 2-4, we select $\texttt{score}_1, \texttt{score}_2 \in \{\texttt{score}_L, \texttt{score}_S, \texttt{score}_A, \texttt{score}_P\}$ and compare the worst-case and data-dependent bounds. First, we plot the worst-case bound from Equation (2), with $\delta = 0.01$, as a function of the number of training examples N. This is the black, dotted line in Figures 2-4.

Next, we plot the data-dependent bound, which is the red, solid line in Figures 2-4. To calculate the data-dependent bound in Equation (3), we have to estimate $\mathbb{E}_{x\sim\mathcal{D}}\left[\left\|f^*_x - g^*_{j,x}\right\|_\infty\right]$ for all $j \in [1600]$.[3] To do so, we draw $M = 6000$ IPs $x_1, \ldots, x_M$ from the distribution $\mathcal{D}$.

[3]We choose the range $j \in [1600]$ because under these distributions, the functions f^*_x generally have at most 1600 pieces.

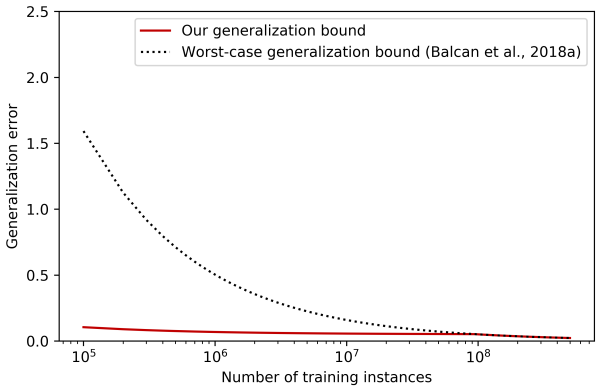

Figure 4. Results using SRM on the CATS "arbitrary" generator with $\texttt{score}_1 = \texttt{score}_P$ and $\texttt{score}_2 = \texttt{score}_A$.

We estimate $\mathbb{E}_{x\sim\mathcal{D}}\left[\left\|f^*_x - g^*_{j,x}\right\|_\infty\right]$ via the empirical average $\frac{1}{M}\sum_{i=1}^{M}\left\|f^*_{x_i} - g^*_{j,x_i}\right\|_\infty$. A Hoeffding bound guarantees that with probability 0.995, for all $j \in [1600]$,

$$\mathbb{E}\left[\left\|f^*_x - g^*_{j,x}\right\|_\infty\right] \le \frac{1}{M}\sum_{i=1}^{M}\left\|f^*_{x_i} - g^*_{j,x_i}\right\|_\infty + \frac{1}{40}. \tag{5}$$

We prove this inequality in Lemma B.3. We thereby estimate our data-dependent bound Equation (3) using Equation (4); the only difference between these bounds is that Equation (3) relies on the left-hand-side of Equation (5) and Equation (4) relies on the right-hand-side of Equation (5) and sets $\delta = 0.005$.[4] In Figures 2-4, the red solid line equals the minimum of Equations (2) and (4) as a function of the number of training examples N.

In Figures 2, 3, and 4, we see that our bound significantly beats the worst-case bound up until the point there are approximately 100,000,000 training instances. At this point, the worst-case guarantee is better than the data-dependent bound, which makes sense because it goes to zero as N goes to infinity, whereas the term $\frac{1}{M}\sum_{i=1}^{M}\left\|f^*_{x_i} - g^*_{j,x_i}\right\|_\infty + \frac{1}{40}$ in our bound (Equation (4)) is a constant.

Figure 2 also illustrates that even when there are only 10^5 training instances, our bound provides a generalization guarantee of approximately 0.1. Meanwhile, $7 \cdot 10^7$ training instances are necessary to provide a generalization guarantee of 0.1 under the worst-case bound, so the sample

[4]Like the worst-case bound, Equation (4) holds with probability 0.99, because with probability 0.995, Equation (5) holds, and with probability 0.995, the bound from Equation (3) holds.

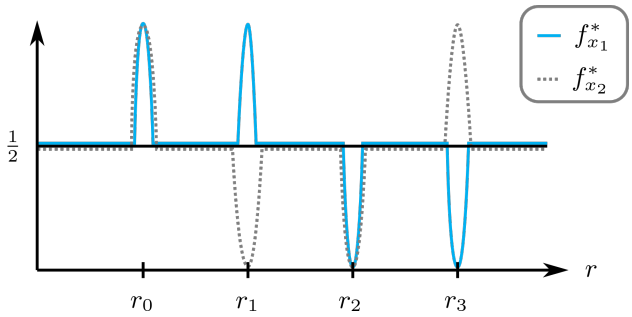

Figure 5. The dual functions $f^*_{x_1}$ and $f^*_{x_2}$ are well-approximated by the constant function $r \mapsto \frac{1}{2}$ under, for example, the L^1-norm because the integrals $\int_{\mathcal{R}} \left|f^*_{x_i}(r) - \frac{1}{2}\right| dr$ are small; for most r, $f^*_{x_i}(r) = \frac{1}{2}$. The approximation is not strong under the L^∞-norm, since $\max_{r\in\mathcal{R}} \left|f^*_{x_i}(r) - \frac{1}{2}\right| = \frac{1}{2}$. The function class $\mathcal{F}$ corresponding to these duals has a large Rademacher complexity.

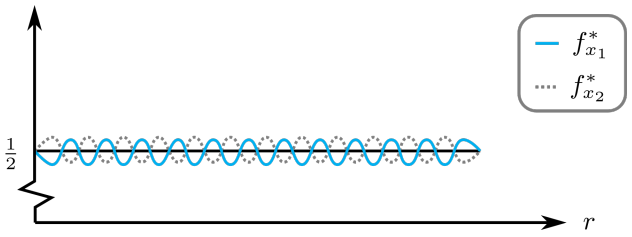

Figure 6. The dual functions $f^*_{x_1}$ and $f^*_{x_2}$ are well-approximated by the constant function $r \mapsto \frac{1}{2}$ under the L^∞-norm since $\max_{r\in\mathcal{R}} \left|f^*_{x_i}(r) - \frac{1}{2}\right|$ is small. The function class $\mathcal{F}$ corresponding to these duals has a small Rademacher complexity.

complexity implied by our analysis is 700 times better. Similarly, in Figure 3, 500 times fewer samples are required to obtain a generalization guarantee of 0.1 under our bound versus the worst-case bound. In Figure 4, 250 times fewer samples are required.

In this section, we approximated the dual functions f^*_x with piecewise constant functions that have a small number of pieces — say, j pieces. We used SRM to find the value for j which leads to the strongest bounds, as in Equation (4). In Appendix B.2.1, we compare against another baseline where we do not use SRM, but simply set j to be the maximum number of pieces we observe over our training set. Of course, this bound is much tighter than the worst-case bound by Balcan et al. (2018a), the baseline in Figures 2-4. However, we still observe that for a target generalization error, the number of samples required according to our bound is up to four times smaller than the number of samples required by this baseline.

4.3. Rademacher Complexity Lower Bound

In this section, we show that (γ, p)-approximability with $p < \infty$ does not necessarily imply strong generalization guarantees of the type we saw in Section 4.1. We show that it is possible for a dual class $\mathcal{F}^*$ to be well-approximated by the dual of a class $\mathcal{G}$ with $\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G}) = 0$, yet for the primal $\mathcal{F}$ to have high Rademacher complexity.

Figures 5 and 6 help explain why there is this sharp constrast between the L^∞- and L^p-norms for $p < \infty$. Figure 5 illustrates two dual functions $f^*_{x_1}$ (the blue solid line) and $f^*_{x_2}$ (the grey dotted line). Let $\mathcal{G}$ be the extremely simple function class $\mathcal{G} = \{g_r : r \in \mathbb{R}\}$ where $g_r(x) = \frac{1}{2}$ for every $x \in \mathcal{X}$. It is easy to see that $\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G}) = 0$ for any set $\mathcal{S}$. Moreover, every dual function g^*_x is also simple, because $g^*_x(r) = g_r(x) = \frac{1}{2}$. From Figure 5, we can see that the functions $f^*_{x_1}$ and $f^*_{x_2}$ are well approximated by the constant function $g^*_{x_1}(r) = g^*_{x_2}(r) = \frac{1}{2}$ under, for example, the L^1-norm because the integrals $\int_{\mathcal{R}} \left|f^*_{x_i}(r) - \frac{1}{2}\right| dr$ are small. However, the approximation is not strong under the L^∞-norm, since $\max_{r\in\mathcal{R}} \left|f^*_{x_i}(r) - \frac{1}{2}\right| = \frac{1}{2}$ for $i \in \{1, 2\}$.

Moreover, despite the fact that $\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G}) = 0$, we have that $\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}) = \frac{1}{2}$ when $\mathcal{S} = \{x_1, x_2\}$, which makes Theorem 4.2 meaningless. At a high level, this is because when $\sigma_1 = 1$, we can ensure that $\sigma_1 f_r(x_1) = \sigma_1 f^*_{x_1}(r) = 1$ by choosing $r \in \{r_0, r_1\}$ and when $\sigma_1 = -1$, we can ensure that $\sigma_1 f_r(x_1) = 0$ by choosing $r \in \{r_2, r_3\}$. A similar argument holds for σ_2. In summary, (γ, p)-approximability for $p < \infty$ does not guarantee low Rademacher complexity.

Meanwhile, in Figure 6, $g^*_{x_i}(r) = \frac{1}{2}$ and $f^*_{x_i}(r)$ are close for every parameter r. As a result, for any noise vector $\boldsymbol{\sigma}$, $\sup_{r\in\mathbb{R}} \left\{\sigma_1 f^*_{x_1}(r) + \sigma_2 f^*_{x_2}(r)\right\}$ is close to $\sup_{r\in\mathbb{R}} \left\{\sigma_1 g^*_{x_1}(r) + \sigma_2 g^*_{x_2}(r)\right\}$. This implies that the Rademacher complexities $\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G})$ and $\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F})$ are close. This illustration exemplifies Theorem 4.3: (γ, ∞)-approximability implies strong Rademacher bounds.

We now prove that (γ, p)-approximability by a simple class for $p < \infty$ does not guarantee low Rademacher complexity.

Theorem 4.9. *For any $\gamma \in (0, 1/4)$ and any $p \in [1, \infty)$, there exist function classes $\mathcal{F}, \mathcal{G} \subset [0, 1]^{\mathcal{X}}$ such that the dual class $\mathcal{G}^*$ (γ, p)-approximates $\mathcal{F}^*$ and for any $N \geq 1$, $\sup_{\mathcal{S}:|\mathcal{S}|=N} \widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G}) = 0$ and $\sup_{\mathcal{S}:|\mathcal{S}|=N} \widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}) = \frac{1}{2}$.*

Proof. We begin by defining the classes $\mathcal{F}$ and $\mathcal{G}$. Let $\mathcal{R} = (0, \gamma^p]$, and $\mathcal{X} = [\gamma^{-p}/2, \infty)$. For any $r \in \mathcal{R}$ and $x \in \mathcal{X}$, let $f_r(x) = \frac{1}{2}(1 + \cos(rx))$ and $\mathcal{F} = \{f_r \mid r \in \mathcal{R}\}$. These sinusoidal functions are based on the intuition from Figure 5. As in Figure 5, for any r and x, let $g_r(x) = \frac{1}{2}$ and $\mathcal{G} = \{g_r \mid r \in \mathcal{R}\}$. Since $\mathcal{G}$ consists of identical copies of a single function, $\widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{G}) = 0$ for any set $\mathcal{S} \subseteq \mathcal{X}$. Meanwhile, in Lemma B.9 in Appendix B.3, we prove that for any $N \geq 1$, $\sup_{\mathcal{S}:|\mathcal{S}|=N} \widehat{\mathcal{R}}_{\mathcal{S}}(\mathcal{F}) = \frac{1}{2}$.

In Lemma B.8 in Appendix B.3, we prove that the dual class $\mathcal{G}^*$ (γ, p)-approximates $\mathcal{F}^*$. To prove this, we first show that $\|f_x^* - g_x^*\|_2 \leq \frac{1}{4}\sqrt{2\gamma^p + \frac{1}{x}}$. When $p = 2$, we know $\frac{1}{x} \leq 2\gamma^2$, so $\|f_x^* - g_x^*\|_2 < \gamma$. Otherwise, we use our bound on $\|f_x^* - g_x^*\|_2$, Hölder's inequality, and the log-convexity of the L^p-norm to prove that $\|f_x^* - g_x^*\|_p \leq \gamma$. □

Remark 4.10. Suppose, for example, that $\mathcal{R} = [0, 1]^d$. Theorem 4.9 implies that even if $|f_x^*(\boldsymbol{r}) - g_x^*(\boldsymbol{r})|$ is small for all x in expectation over $\boldsymbol{r} \sim \text{Uniform}(\mathcal{R})$, the function class $\mathcal{F}$ may not have Rademacher complexity close to $\mathcal{G}$.

Statistical learnability. In Appendix B.4, we connect our results to the literature on statistical learnability (Haussler, 1992). At a high level, a function class $\mathcal{F}$ is *statistically learnable* (Definition A.2 in Appendix A) if there exists a learning algorithm that returns a function whose expected value converges—as the size of the training set grows—to the smallest expected value of any function in $\mathcal{F}$. We introduce a relaxation: a function class $\mathcal{F}$ is γ-*statistically learnable* (Definition A.3) if, at a high level, there exists a learning algorithm with error at most γ in the limit as the training set size grows. We prove that if the dual class $\mathcal{F}^*$ is (γ, ∞)-approximated by the dual of a statistically learnable class $\mathcal{G}$, then $\mathcal{F}$ is γ-statistically learnable. On the other hand, Theorem 4.9 implies that there exists a class $\mathcal{F}$ that is not γ-statistically learnable, yet it is (γ, p)-approximated by the dual of a statistically learnable class $\mathcal{G}$.

5. Conclusions

We provided generalization guarantees for algorithm configuration, which bound the difference between a parameterized algorithm's average empirical performance over a set of sample problem instances and its expected performance on future, unseen instances. We did so by exploiting structure exhibited by the *dual functions* which measure the algorithm's performance as a function of its parameters. We analyzed the widely-applicable setting where the dual functions are approximated by "simple" functions. We showed that if this approximation holds under the L^∞-norm, then it is possible to provide strong generalization guarantees. If, however, the approximation only holds under the L^p-norm for $p < \infty$, we showed that it is impossible in the worst-case to provide non-trivial bounds. Via experiments in the context of integer programming algorithm configuration, we demonstrated that our bounds can be significantly stronger than the best-known worst-case guarantees (Balcan et al., 2018a), leading to a sample complexity improvement of 70,000%.

We conclude with a promising direction for future research. Suppose, for some prior $\mathcal{P}$ over parameters, $\mathbb{E}_{x\sim\mathcal{D},\boldsymbol{r}\sim\mathcal{P}}\left[|f_x^*(\boldsymbol{r}) - g_x^*(\boldsymbol{r})|\right]$ is small. From Remark 4.10, we know strong generalization bounds are not possible in the worst case, but what about under some realistic assumptions? This may help us understand, for example, why random forests—which have a simple piecewise-constant structure—are often able to accurately predict the runtime of SAT and MIP solvers (Hutter et al., 2011).

Acknowledgments. We thank Kevin Leyton-Brown for a stimulating discussion that inspired us to pursue this research direction.

This material is based on work supported by the National Science Foundation under grants CCF-1535967, CCF-1733556, CCF-1910321, IIS-1617590, IIS-1618714, IIS-1718457, IIS-1901403, and SES-1919453; the ARO under awards W911NF1710082 and W911NF2010081; a fellowship from Carnegie Mellon University's Center for Machine Learning and Health; the Defense Advanced Research Projects Agency under cooperative agreement HR0011202000; an Amazon Research Award; an AWS Machine Learning Research Award; an Amazon Research Award; a Bloomberg Research Grant; and a Microsoft Research Faculty Fellowship.

References

Tobias Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1): 1–41, 2009.

Alejandro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel. A machine learning-based approximation of strong branching. *INFORMS Journal on Computing*, 29 (1):185–195, 2017.

Martin Anthony and Peter Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 2009.

Patrick Assouad. Densité et dimension. *Annales de l'Institut Fourier*, 33(3):233–282, 1983.

Amine Balafrej, Christian Bessiere, and Anastasia Paparrizou. Multi-armed bandits for adaptive constraint propagation. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. *Conference on Learning Theory (COLT)*, 2017.

Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International Conference on Machine Learning (ICML)*, 2018a.

Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2018b.

Maria-Florina Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized Lloyd's families. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018c.

Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. How much data is sufficient to learn high-performing algorithms? *arXiv preprint arXiv:1908.02894*, 2019.

Maria-Florina Balcan, Travis Dick, and Manuel Lang. Learning to link. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020a.

Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. Learning to optimize computational resources: Frugal training with generalization guarantees. *AAAI Conference on Artificial Intelligence (AAAI)*, 2020b.

Evelyn Beale. Branch and bound methods for mathematical programming systems. *Annals of Discrete Mathematics*, 5:201–219, 1979.

Michel Bénichou, Jean-Michel Gauthier, Paul Girodet, Gerard Hentges, Gerard Ribière, and O Vincent. Experiments in mixed-integer linear programming. *Mathematical Programming*, 1(1):76–94, 1971.

Giovanni Di Liberto, Serdar Kadioglu, Kevin Leo, and Yuri Malitsky. Dash: Dynamic approach for switching heuristics. *European Journal of Operational Research*, 248(3): 943–953, 2016.

J-M Gauthier and Gerard Ribière. Experiments in mixed-integer linear programming using pseudo-costs. *Mathematical Programming*, 12(1):26–47, 1977.

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

David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and computation*, 100(1):78–150, 1992.

He He, Hal Daume III, and Jason M Eisner. Learning to search in branch and bound algorithms. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

Eric Horvitz, Yongshao Ruan, Carla Gomez, Henry Kautz, Bart Selman, and Max Chickering. A Bayesian approach to tackling hard computational problems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

Frank Hutter, Holger Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009. ISSN 1076-9757.

Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization (LION)*, pages 507–523, 2011.

Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC-instance-specific algorithm configuration. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 2010.

Elias Boutros Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

Elias Boutros Khalil, Bistra Dilkina, George Nemhauser, Shabbir Ahmed, and Yufen Shao. Learning to run heuristics in tree search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

Robert Kleinberg, Kevin Leyton-Brown, and Brendan Lucier. Efficiency through procrastination: Approximately optimal algorithm configuration with runtime guarantees. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

Robert Kleinberg, Kevin Leyton-Brown, Brendan Lucier, and Devon Graham. Procrastinating with confidence: Near-optimal, anytime, adaptive algorithm configuration. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001.

Markus Kruber, Marco E Lübbecke, and Axel Parmentier. Learning when to use a decomposition. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 202–210. Springer, 2017.

Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica*, pages 497–520, 1960.

Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 66–76, Minneapolis, MN, 2000.

Jeff Linderoth and Martin Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal of Computing*, 11(2):173–187, 1999.

Andrea Lodi and Giulia Zarpellon. On learning and branching: a survey. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 25(2):207–236, 2017.

Pascal Massart. Some applications of concentration inequalities to statistics. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 9, pages 245–303, 2000.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT Press, 2012.

George Nemhauser and Laurence Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999.

Ashish Sabharwal, Horst Samulowitz, and Chandra Reddy. Guiding combinatorial optimization with UCT. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Springer, 2012.

Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, January 2002.

Tuomas Sandholm. Very-large-scale generalized combinatorial multi-attribute auctions: Lessons from conducting $60 billion of sourcing. In Zvika Neeman, Alvin Roth, and Nir Vulkan, editors, *Handbook of Market Design*. Oxford University Press, 2013.

Karthik Sridharan. *Learning from an optimization viewpoint*. PhD thesis, Toyota Technological Institute at Chicago, 2012.

Gellért Weisz, András György, and Csaba Szepesvári. LEAPSANDBOUNDS: A method for approximately optimal algorithm configuration. In *International Conference on Machine Learning (ICML)*, 2018.

Gellért Weisz, Andrés György, and Csaba Szepesvári. CAPSANDRUNS: An improved method for approximately optimal algorithm configuration. *International Conference on Machine Learning (ICML)*, 2019.