How much data is sufficient to learn high-performing algorithms?

Maria-Florina Balcan,[1,2,*] Dan DeBlasio,[3] Travis Dick,[1]
Carl Kingsford,[3,4,*] Tuomas Sandholm,[1,5,6,7,*] Ellen Vitercik[1]

[1]Computer Science Department, School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
[2]Machine Learning Department, School of Computer Science, Carnegie Mellon University
[3]Computational Biology Department, School of Computer Science, Carnegie Mellon University
[4]Ocean Genomics, Inc., Pittsburgh, PA 15217, USA
[5]Optimized Markets, Inc., Pittsburgh, PA 15213, USA
[6]Strategic Machine, Inc., Pittsburgh, PA 15213, USA
[7]Strategy Robot, Inc., Pittsburgh, PA 15213, USA
Emails: {ninamf, deblasio, tdick, carlk, sandholm, vitercik}@cs.cmu.edu.

October 29, 2019

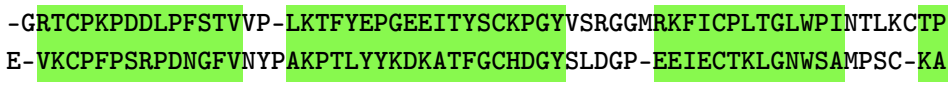
Abstract

Algorithms—for example for scientific analysis—typically have tunable parameters that significantly influence computational efficiency and solution quality. If a parameter setting leads to strong algorithmic performance on average over a set of training instances, that parameter setting—ideally—will perform well on previously unseen future instances. However, if the set of training instances is too small, average performance will not generalize to future performance. This raises the question: how large should this training set be? We answer this question for any algorithm satisfying an easy-to-describe, ubiquitous property: its performance is a piecewise-structured function of its parameters. We provide the first unified sample complexity framework for algorithm parameter configuration; prior research followed case-by-case analyses. We present example applications to diverse domains including biology, political science, economics, integer programming, and clustering.

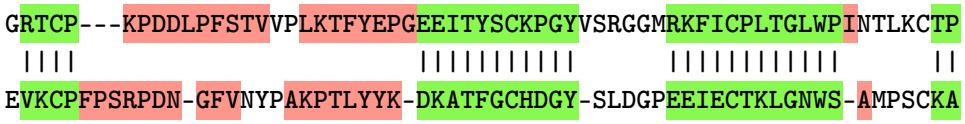
1 Introduction

For decades, algorithmic innovations have led to breakthroughs throughout science. Algorithms are step-by-step procedures for finding solutions to mathematical problems, and typically have many tunable parameters. These adjustable settings can significantly influence an algorithm's requisite computational resources and the quality of the solutions it returns. Poorly-tuned parameters can even mislead scientists into making false claims. Domain experts often fine-tune algorithm parameters by hand. During this time-consuming search, they may overlook many high-performing settings of the parameters.

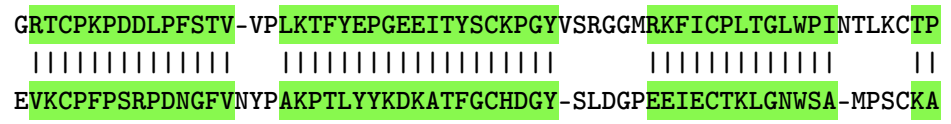
Optimal parameter settings typically depend intimately on the specific application domain. As a concrete example, computational biologists use algorithms to align DNA, RNA, and protein strings. Ideally, the resulting alignments identify regions of similarity that indicate functional, structural, or evolutionary relationships among the sequences. Scientists typically aim to optimize



(a) A ground-truth alignment. The original two sequences consist only of alphabetic characters, and the inserted dash characters align the strings.



(b) An alignment returned by an algorithm with poorly-tuned parameters.



(c) An alignment returned by an algorithm with well-tuned parameters.

Figure 1: A ground-truth alignment of two protein sequences over the amino acid alphabet are shown in Figure (a). Figures (b) and (c) illustrate two alignments the Opal [Wheeler and Kececioglu, 2007] algorithm returns using two different parameter settings. The bar characters in the bottom two figures illustrate where the alignments match the ground-truth alignment. The alignment in Figure (c) matches the ground-truth alignment much better than the alignment in Figure (b): Figure (c) matches 46 columns as opposed to the 27 matched in Figure (b). The only difference between the two computed alignments are the parameter settings the alignment algorithm uses.

alignment features such as the number of matching characters, the length of the alignments, and so on. A string alignment algorithm uses parameters to weight these features, and then solves for the alignment that maximizes the features' weighted sum. In practice, it is rarely clear how to tune these parameters.

We study automated algorithm parameter tuning via machine learning, with broad applications such as sequence alignment in biology, mechanisms for collective decision making in political science and economics, integer programming in optimization, clustering, and so on. This automated approach relieves domain experts of this error-prone, yet essential task. Our analysis applies to settings where the domain expert has a set of typical problem instances from her application domain, also known as a *training set*. The domain expert might also have access to ground-truth solutions to these problem instances — computational biologists, for example, often have access to a small number of *ground-truth* alignments for a handful of sequences, as we illustrate in Figure 1 — though this is not always necessary, as we exemplify in our applications from economics and political science later on. A natural approach is for the domain expert to find a parameter setting with satisfactory *algorithmic performance* on average over the training set, then apply those parameters to solve all future problem instances. Algorithmic performance can mean different things in different application domains, ranging from solution quality (the similarity between the ground-truth solution and the solution the algorithm finds, for example) to computational resource usage

(run time or memory usage, for instance); our approach applies to all of the above. The domain expert must be careful, however, when employing this technique: if her set of training instances is too small, parameters with strong performance on average over the training set may have poor future performance. This phenomenon, known as "overfitting," raises the question:

> *How many samples are sufficient to ensure that any parameter setting's average performance over the training set generalizes to its future performance?*

More succinctly, how many samples are sufficient to ensure generalization? Formally, we assume that both the future problem instances and those in the training set are independently drawn from the same unknown, application-specific distribution.

The main result in this paper is a bound on the number of samples sufficient to ensure generalization for any parameterized algorithm that satisfies an easy-to-describe ubiquitous structural property. Namely, for any problem instance, the algorithm's performance as a function of its parameters is *piecewise structured*: the parameter space decomposes into a small number of equivalence classes, or components, such that within each component, the algorithm's performance is well-behaved. Each component is delineated by boundary functions (such as hyperplanes) and within each component, the function relating the algorithm's performance to its parameters belongs to a function class with small intrinsic complexity (for example, the class of constant, linear, or quadratic functions). We prove that the number of samples sufficient to ensure generalization depends on the intrinsic complexity of both the boundary functions and of the component functions defining algorithm's performance within each component. Moreover, we prove that the sample complexity grows minimally with the number of boundary functions that split the parameter space into components. We precisely quantify the complexity of these function classes.

We instantiate the theorem's sample complexity guarantees in settings ranging from computational biology to algorithmic economics. A strength of our result is that it applies no matter which configuration algorithm the domain expert employs; for any parameters she selects, we guarantee that average performance over a sufficiently large training set generalizes to future performance. We also provide a general application-independent procedure for applying this theorem that domain experts can use for their own tunable algorithms. Finally, we provide experiments from biology and economics that demonstrate that carefully tuning an algorithm's parameters can have a substantial effect on the quality of its output (Section 4.4).

Researchers have studied automated parameter tuning — also called *algorithm configuration*, *automated algorithm design*, and *parameter advising* — for decades, leading to advances in optimization in artificial intelligence [Horvitz et al., 2001, Sandholm, 2013, Xu et al., 2008], computational biology [DeBlasio and Kececioglu, 2018, May et al., 2017, Majoros and Salzberg, 2004], and myriad other fields. This applied research often adopts a model identical to ours: the practitioner has a set of typical problem instances from her domain and uses computational tools to tune parameters based on this training set. In contrast to our framework, the vast majority of this prior research is purely empirical, not providing any guarantees.

We present the most general sample complexity bounds yet for algorithm configuration. A nascent line of research [Gupta and Roughgarden, 2017, Balcan et al., 2017, 2018c,a,b] presents sample complexity guarantees for a selection of tunable algorithms. Unlike the results presented this paper, those papers analyze each algorithm individually, employing case-by-case analyses to derive sample complexity guarantees. We prove that our approach recovers the sample complexity guarantees from prior research, and our results apply broadly across application domains and algorithms.

Computational learning theory has already been used to study generalization in machine learning. A key challenge distinguishes our results from that line of research on sample complexity: the

general volatility of an algorithm's performance as a function of its parameters. For well-understood functions in machine learning, there is a simple connection between a function's parameters and its value; as we vary the parameters, the value changes smoothly. This straightforward connection is typically the key to providing sample complexity guarantees. Meanwhile, for most tunable algorithms, slightly perturbing the parameters can cause a cascade of changes in the algorithm's behavior, thus triggering a jump in the algorithm's performance. Another difference from most (but not all) research on machine learning theory is that here, as we will discuss in detail, we have to study the complexity of a *dual* class. Due to these differences, we must uncover an alternative structure that links the algorithm's parameters and its performance in order to provide sample complexity guarantees. The piecewise structure we present is specific enough to imply strong sample complexity bounds, yet abstract enough that it applies to a diverse array of algorithm configuration problems.

1.1 A theory for parametric algorithm configuration

To prove our general theorem, we build on a long line of research in machine learning theory on sample complexity: given a set of functions together with sample access to an unknown distribution over its domain, classic results from learning theory bound the number of samples sufficient to ensure that for any function in the class, its average value over the samples nearly matches its expected value. At a high level, these bounds are formulated in terms of the function class's *intrinsic complexity*. A standard tool for measuring intrinsic complexity is *pseudo-dimension* [Pollard, 1984], which we formally define in Section 2.3. We denote the intrinsic complexity of a class $\mathcal{F}$ of functions mapping a domain $\mathcal{X}$ to $\mathbb{R}$ using the notation $C_{\mathcal{F}}$. Generally speaking, this real-valued $C_{\mathcal{F}}$ measures the ability of functions in $\mathcal{F}$ to fit complex patterns over many domain elements $x \in \mathcal{X}$.

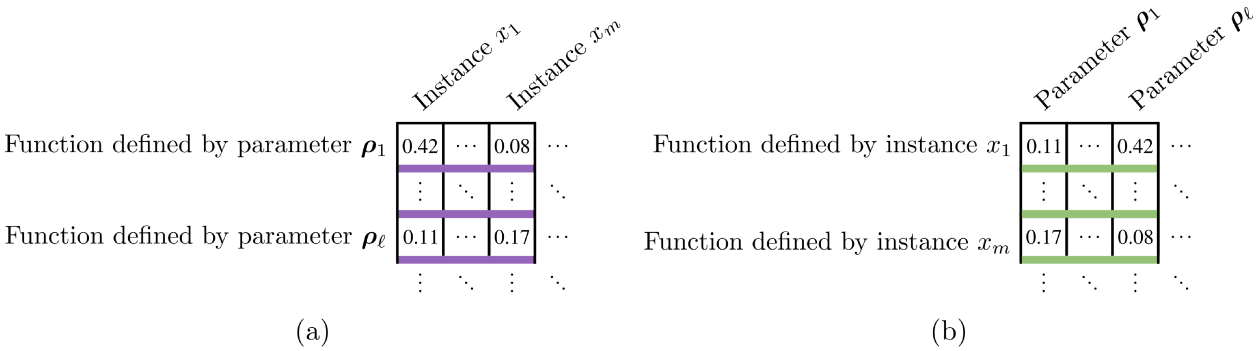
Intuitively, the more complex a function class is, the more samples we require to guarantee that each function's average value over the samples is nearly equal to its expected value. Pollard [1984] formalized this intuition, proving that when the range of the functions f in $\mathcal{F}$ is $[0, 1]$, then with probability $1 - \delta$ over the draw of

$$\frac{8}{\epsilon^2}\left(C_{\mathcal{F}} \ln \frac{8}{\epsilon^2} + \frac{1}{4} \ln \frac{1}{\delta}\right) \tag{1}$$

sample problem instances, the average value of any function over the samples is within ϵ of its expected value. As one shrinks ϵ and δ, the requisite sample size increases while the guarantee becomes stronger.

In algorithm configuration, the function class of interest measures a parameterized algorithm's performance as a function of the algorithm's input problem instance. We denote this function class by $\mathcal{A}$. Thus, every function in $\mathcal{A}$ is characterized by a parameter setting. If we can measure the intrinsic complexity $C_{\mathcal{A}}$ of the class $\mathcal{A}$, then we can use Equation (1) to bound the number of samples sufficient to ensure generalization. We can therefore guarantee that given $\frac{8}{\epsilon^2}\left(C_{\mathcal{A}} \ln \frac{8}{\epsilon^2} + \frac{1}{4} \ln \frac{1}{\delta}\right)$ training instances, the expected performance of the best parameters over those training instances is nearly optimal.

We present a general theorem that bounds the complexity $C_{\mathcal{A}}$ of the function class $\mathcal{A}$ corresponding to a diverse array of algorithms. Our main innovation is to bound the intrinsic complexity $C_{\mathcal{A}}$ using structure exhibited by what we call the *dual* function class $\mathcal{A}^*$ (illustrated in Figure 2). Each function in the original family $\mathcal{A}$ is characterized by a parameter setting and takes as input a problem instance; it measures algorithmic performance as a function of the input. Meanwhile, each function in the dual family $\mathcal{A}^*$ is characterized by a fixed problem instance; it measures algorithmic performance as a function of the parameter when the algorithmic is given that instance as its input.



(a) (b)

Figure 2: Illustration of the dual. We describe the semantic meaning of the dual in the algorithm configuration setting below. Each row i in Figure (a) represents a function in the family $\mathcal{A}$, which is characterized by a parameter setting $\boldsymbol{\rho}_i$. Each column represents a different input. Each entry in the row defined by a parameter vector $\boldsymbol{\rho}_i$ is the parameterized algorithm's performance as a function of the corresponding input x_j. Meanwhile, each row in Figure (b) represents a function in the dual family $\mathcal{A}^*$, which is defined by an input problem instance. Each column represents an algorithm parameter vector. Each entry in the row defined by an input x_j is the algorithm's performance as a function of the corresponding parameter, given x_j as input.

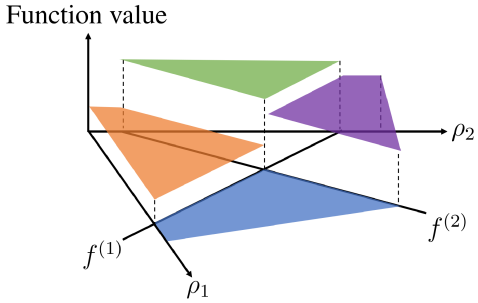


Figure 3: Example of a piecewise *structured* function. Here there are two functions that partition the space into 4 regions, and within each region the function value stays constant.

Here the notion of a dual has the above clear meaning in the context of the algorithm configuration problem. However, the notion of dual here is actually the same as the abstract notion of the dual in mathematics.

Structure exhibited by the dual algorithm family allows us to prove general widely-applicable guarantees. We prove that a large number of dual algorithm families share a useful structural property: for each problem instance, the algorithm's performance as a function of its parameters is piecewise constant, piecewise linear, or—more broadly—piecewise *structured*. As an example, Figure 3 illustrates a piecewise structured function of two parameters, ρ_1 and ρ_2. There are two functions that define a partition of the parameter space (linear separators $g^{(1)}$ and $g^{(2)}$) and four constant functions that define the function value on each subset from this partition.

Motivated by the prevalence of various piecewise structure, we say that the dual algorithm family is $(\mathcal{F}, \mathcal{G}, k)$-*piecewise decomposable* if for every problem instance, there are at most k boundary functions within a set $\mathcal{G}$ (for example, the set of linear separators) that induce a partition of the parameter space such that within any subset from this partition, algorithmic performance is defined by a function within a set $\mathcal{F}$ (for example, the set of constant functions).

We show that bounding the inherent complexity of the dual functions of $\mathcal{F}$ and $\mathcal{G}$, which we

call $\mathcal{F}^*$ and $\mathcal{G}^*$ respectively, can be used to bound the complexity of $\mathcal{A}$ and thereby obtain sample complexity results. (Here these dual functions are defined in the standard abstract mathematical sense from $\mathcal{F}$ and $\mathcal{G}$, and may not have a semantic meaning in terms of algorithm configuration.) Specifically, we prove that $C_{\mathcal{A}}$ is bounded by

$$4\left(C_{\mathcal{G}^*} + C_{\mathcal{F}^*}\right)\ln\left(4ek\left(C_{\mathcal{G}^*} + C_{\mathcal{F}^*}\right)\right).$$

The details of this guarantee and its proof are in Section 3.

Substituting this into Equation (1) then yields the main sample complexity theorem of this paper.

Theorem 1.1. *Assuming the range of the functions in $\mathcal{A}$ is normalized to $[0,1]$, then with probability $1-\delta$ over the draw of*

$$\frac{8}{\epsilon^2}\left(4\left(C_{\mathcal{G}^*} + C_{\mathcal{F}^*}\right)\ln\left(4ek\left(C_{\mathcal{G}^*} + C_{\mathcal{F}^*}\right)\right)\ln\frac{8}{\epsilon^2} + \frac{1}{4}\ln\frac{1}{\delta}\right) \tag{2}$$

training problem instances, for any setting of the algorithm parameters, the average performance of the algorithm over the training instances is within ϵ of its actual expected performance over the unknown real problem instance distribution.

Broad applicability of the main theorem. We find that the function classes $\mathcal{F}$ and $\mathcal{G}$ that occur in most applications are typically structured in ways for which the intrinsic complexity of their dual functions can be analyzed and shown to be low. Those complexity measures can then be substituted into our main theorem to obtain the sample complexity guarantee. Examples of important common structures include the following.

- When $\mathcal{F}$ is the class of constant functions and $\mathcal{G}$ is the class of d-dimensional hyperplanes (as exemplified in Figure 3), $C_{\mathcal{F}^*} = 1$ and $C_{\mathcal{G}^*} = d+1$. Therefore, by our main theorem, the number of samples that suffices grows only linearly with the number of algorithm parameters we are trying to learn. This structure occurs in many applications such as all of the biology applications we discuss in this paper, integer linear programming optimization algorithm configuration (which we discuss in Section 4.3.2), and in welfare maximization in social choice mechanism design settings we discuss in this paper.

- More generally, when $\mathcal{F}$ is the class of *linear* functions and $\mathcal{G}$ is still the class of d-dimensional hyperplanes, $C_{\mathcal{F}^*} = d+1$ and $C_{\mathcal{G}^*} = d+1$. Therefore, by our main theorem, the number of samples that suffices again grows only linearly with the number of algorithm parameters we are trying to learn. This structure occurs in many applications such as revenue maximization mechanism design in auctions and pricing, as we discuss later in this paper.

- Furthermore, when the parameter is one-dimensional, we identify a structure that captures dual functions that are piecewise constant, piecewise linear, piecewise polynomial, or more generally, functions that oscillate a bounded number of times, as we formalize in Section 3.1. This structure implies guarantees for applications such as revenue maximization in mechanism design and nonlinear programming approximation algorithm configuration.

1.2 Applications

In this section, we instantiate our main sample complexity theorem in diverse applications from computational biology and economics.

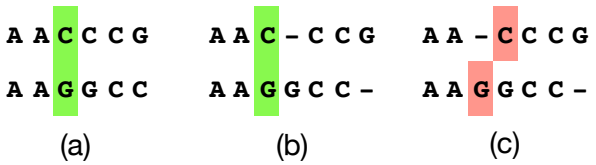


Figure 4: Example alignments of the sequences `AACCCG` and `AAGGCC`. Alignments (b) and (c) are optimal for parameter settings $\alpha = 2, \beta = 1.25$, and $\gamma = 1.0$, and (a) is the reference alignment. Note that while both (b) and (c) are optimal under the alignment objective function, (b) recovers an additional substitution from the reference giving it a higher utility value.

1.2.1 Applications in biology

We study the sample complexity of four common problems from biology: pairwise sequence alignment, multiple sequence alignment, RNA folding, and topologically associated domain finding. In all of these applications, there are two unifying similarities, which we describe below.

First, a solution's quality, which we also refer to as its utility, is measured with respect to a ground-truth solution. This gold-standard solution is constructed in most cases by laboratory experimentation, so it is only available for the problem instances in the training set. Algorithmic performance is then measured in terms of the distance between the solution the algorithm outputs and the ground-truth solution.

Second, the biology algorithms we study all return solutions that maximize some parameterized objective function. Often, there may be multiple solutions that maximize this objective function; we call these solutions *co-optimal*. Although co-optimal solutions have the same objective function value, they may have different utilities. In practice, in any region of the parameter space where the set of co-optimal solutions is invariant, the algorithm's output is invariant as well. We call this type of algorithm *co-optimal constant*. Throughout the remainder of this section, we assume the parameterized algorithms are co-optimal constant. This assumption ensures that within regions of co-optimality in the parameter space, utility is constant, which allows us to apply our main sample complexity theorem.

Global pairwise sequence alignment. Aligning two strings — English sentences, biological sequences, and so on — is a fundamental problem throughout science. The high-level goal is to line up the two strings in order to identify regions of similarity, and there are several classical algorithms that accomplish this task [Needleman and Wunsch, 1970, Smith and Waterman, 1981]. In biology, for example, these similar regions ideally indicate functional, structural, or evolutionary relationships between the sequences.

Given two sequences $S_1, S_2 \in \Sigma^*$ over an alphabet Σ, an alignment is formally a $2 \times k$ grid L with $k \geq \max\{|S_1|, |S_2|\}$, where each row contains the characters from one of the sequences, in order, with inserted gap characters (denoted '-' $\notin \Sigma$). Figure 4 shows several alignments of the same sequences.

There are many features of an alignment that can be used as the optimization criteria. The four most common are the number of columns in the alignment that have the same character (*matches*, denoted $\textsc{mt}(S_1, S_2, L)$), the number of columns that do not have the same character (*mismatches*, denoted $\textsc{ms}(S_1, S_2, L)$), the total number of gap characters (*indels*, short for insertion/deletion, denoted $\textsc{id}(S_1, S_2, L)$), and the number of groups of consecutive gap characters in any one row of the grid (*gaps*, denoted $\textsc{gp}(S_1, S_2, L)$). In Figure 4, the alignment in (a) has 3 matches, 3 mismatches,

0 indels, and 0 gaps; the alignments in (b) and (c) have 4 matches, 1 mismatch, 2 indels, and 2 gaps. This figure exemplifies the reason we consider co-optimal constant algorithms: if (a) is the ground-truth alignment, then alignments (b) and (c) would have the same objective function score but different utility scores. The utility, or distance to the ground truth, of an alignment is the fraction of aligned characters from the ground truth that are recovered in the computed alignment. Since (b) recovers more columns from (a) than (c)—the ones highlighted—it has a higher utility.

There are many variations on this problem. In this section, we study *global pairwise alignment with affine-gap penalties* [Gotoh, 1982], a widely-studied formulation which can be optimized efficiently [Myers and Miller, 1988]. Here the term *affine* refers to the fact that the number of gaps and the number of indels may be weighted differently. In the evolutionary process that the objective function models, it may be easier to extend an existing run of insertions or deletions than to create a new one.

The goal is to find an alignment L that maximizes the objective function

$$\textsc{mt}(S_1, S_2, L) - \alpha \cdot \textsc{ms}(S_1, S_2, L) - \beta \cdot \textsc{id}(S_1, S_2, L) - \gamma \cdot \textsc{gp}(S_1, S_2, L) \tag{3}$$

where $\alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}$ are tunable parameters.

In our model, the domain expert has access to a training set of sequence pairs. For each pair, she is given a ground-truth alignment. Our goal is to learn parameters $\alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}$ so that on a new problem instance, the parameterized sequence-alignment algorithm returns a solution that is close to the unknown, ground-truth alignment. Thus, the performance of the sequence-alignment algorithm is measured in terms of the distance between its output and the ground-truth alignment. Our main theorem provides a bound on the number of samples sufficient to ensure that if we find parameters with strong algorithmic performance on average over the training set, then they will also have strong performance on future problem instances. To instantiate our main theorem, we follow the procedure we provided in the previous section. In our calculations, we assume there is an upper bound, n, on the length of the sequences the algorithm is asked to align.

First, we fix an arbitrary pair of sequences S_1 and S_2. A line of research by Gusfield et al. [1994], Fernández-Baca et al. [2004], and Pachter and Sturmfels [2004a] proved that for some constant $c \in \mathbb{R}$, there are $cn^{3/2}$ distinct optimal solutions over the range of parameters $(\alpha, \beta, \gamma) \in \mathbb{R}^3$. There are at most c^2n^3 hyperplanes that divide the parameter space into at most $\left(c^2n^3 + 1\right)^3$ cells such that any parameter vector (α, β, γ) in that cell, the alignment maximizing Equation (3) is invariant. Within any one region, since the alignment the algorithm returns is invariant, the algorithmic performance—distance to the ground-truth alignment—is constant. Thus, the dual algorithm class is $\left(\mathcal{F}, \mathcal{G}, c^2n^3\right)$-piecewise decomposable, where $\mathcal{G}$ is the set of hyperplanes in $\mathbb{R}^3$ and $\mathcal{F}$ is the set of constant functions. Our general theorem guarantees that with probability $1 - \delta$ over the draw of

$$\frac{8}{\epsilon^2}\left(20 \ln\left(20ec^2n^3\right) \ln\frac{8}{\epsilon^2} + \frac{1}{4}\ln\frac{1}{\delta}\right)$$

sample problem instances, the average algorithmic performance of any parameter setting over the samples is within ϵ of the parameter setting's expected algorithmic performance. So, the number of samples sufficient to ensure generalization increases only proportionally with $\ln n$.

Furthermore, the work of Pachter and Sturmfels [2004a] enables us to provide sample complexity bounds for any alignment objective that can be formulated as a hidden Markov model. This includes most alphabet-dependent alignment schemes. In these cases the number of separating hyperplanes has an exponential dependence on the number of parameters d. Thus we are able to show that these alignment formulations are $\left(\mathcal{F}, \mathcal{G}, cn^{\frac{2d(d-1)}{d+1}}\right)$-piecewise decomposable where $\mathcal{G}$ is

the set of hyperplanes in $\mathbb{R}^d$ and $\mathcal{F}$ is the set of constant functions. Our general theorem then guarantees number of samples sufficient to ensure generalization increases only linearly with d and only logarithmically with n.

Progressive multiple sequence alignment. In many applications, such as phylogenetics and homology search, there are more than two sequences to align. The extension from pairwise to multiple sequence alignment, however, is computationally challenging: all common formulations of the problem are NP-complete [Wang and Jiang, 1994, Kececioglu and Starrett, 2004]. So, every algorithm that solves the multiple sequence alignment problem exactly can take an inordinate amount of time to find a solution. Therefore, there are heuristics to find good, but possibly suboptimal, alignments. The most common heuristic approach is called *progressive multiple sequence alignment* [Feng and Doolittle, 1987]. At its core, this technique uses the family of pairwise alignment algorithms from the previous section. At a high level, the algorithm uses a binary[1] tree to decompose the original alignment problem into a hierarchy of subproblems, each of which it solves using the pairwise alignment algorithm. We formally describe the parameterized algorithm in Section 4.1.2, and prove that the number of samples sufficient for generalization is proportional to $n \ln(n\ell)$, where ℓ is the number of sequences and n is the maximum length of those sequences. Our sample complexity guarantee is thus higher than in the case of pairwise sequence alignment.

RNA secondary structure prediction. RNA molecules have many essential roles, including protein coding and enzymatic functions [Holley et al., 1965]. RNA is assembled as a chain of *bases* denoted using the characters `A`, `U`, `C`, and `G`. It is often found as a single strand folded onto itself: non-adjacent bases physically bound together. Given an unfolded RNA strand, the goal is to infer the way it would naturally fold, which sheds light on its function. This problem is known as *RNA secondary structure prediction*, or simply *RNA folding*.

More formally, given a sequence $S \in \{\texttt{A}, \texttt{U}, \texttt{C}, \texttt{G}\}^n$, we represent a folding by a set of pairs $\phi \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$. If the pair (i, j) is in the folding ϕ, then the i^{th} and j^{th} bases of S physically bind together. Typically, the bases `A` and `U` bind together, as do the bases `C` and `G`. Other matchings may occur, but the resulting structure is likely to be less stable. We assume, as is standard, that the folding does not contain any *pseudoknots*: pairs $(i, j), (i', j')$ that cross with $i < i' < j < j'$.

A well-studied algorithm for the problem returns a folding that maximizes a parameterized objective function [Nussinov and Jacobson, 1980]. At a high level, this objective function trades off between global properties of the folding (the number of binding pairs $|\phi|$) and local properties (the likelihood that bases would appear close together in the folding). Specifically, given a parameter $\alpha \in [0, 1]$, the algorithm returns the folding ϕ that maximizes the objective function

$$\alpha \, |\phi| + (1 - \alpha) \sum_{(i,j) \in \phi} M_{S_i, S_j, S_{i-1}, S_{j+1}} \mathbb{I}_{\{(i-1, j+1) \in \phi\}}, \tag{4}$$

where $M_{w,x,y,z}$ is a score for having neighboring pairs of the letters (w, x) and (y, z) and $\mathbb{I}$ is the indicator function which returns 1 when the pair is in the folding and 0 otherwise. These scores help identify sub-structures that are more stable than others.

In our model, the domain expert has access to a training set of RNA strands together with a ground-truth folding, which she obtains via an expensive computation or laboratory experimentation. Our goal is to learn a parameter $\alpha \in [0, 1]$ so that given a new RNA strand, the algorithm returns a solution that is close to the unknown, ground-truth folding.

[1] In general, guide trees do not have to be binary; for ease of analysis, we impose this limit.

To apply our main theorem we first fix an arbitrary strand S. The total number of foldings with no pseudoknots is bounded by $\binom{n^2}{n/2} \leq n^n$. In Section 4.1.3, we argue that for any pair of foldings, there is a threshold $\alpha_0 \in \mathbb{R}$ where Equation (4) is larger for the first folding when α is on one side of the threshold, and its larger for the second folding when α is on the other side of the threshold. Within any interval induced by these n^{2n} thresholds, the folding that maximizes Equation (4) is invariant, and thus algorithmic performance — distance to the ground-truth folding — is constant. Therefore, the dual algorithm class is $\left(\mathcal{F}, \mathcal{G}, n^{2n}\right)$-piecewise decomposable, where $\mathcal{G}$ is the set of thresholds in $\mathbb{R}$ and $\mathcal{F}$ is the set of constant functions. Our general theorem guarantees that with probability $1 - \delta$ over the draw of

$$\frac{8}{\epsilon^2}\left(4\left(\ln\left(4e\right) + 2n\ln n\right)\ln\frac{8}{\epsilon^2} + \frac{1}{4}\ln\frac{1}{\delta}\right)$$

sample problem instances, the average algorithmic performance of any parameter setting over the samples is within ϵ of the parameter setting's expected algorithmic performance. The number of samples sufficient to ensure generalization increases proportionally with $n \ln n$.

As our bounds demonstrate, sample complexity is not necessarily tied to the computational complexity of a problem. The sample complexity of both the RNA folding and progressive multiple sequence alignment problems grows proportionally with $n \ln n$, whereas the computational complexities are distinct: RNA folding can be solved in polynomial time and multiple sequence alignment is NP-complete.

Prediction of topologically associating domains. Inside a cell, the linear DNA of the genome wraps into three-dimensional structures that influence genome function. Some regions of the genome are closer than others and thereby interact more. An important class of structure is called *topological associating domains (TADs)* that are contiguous segments of the genome that fold into compact regions. Formally, given a DNA sequence of length n, a TAD set T is a set of non-overlapping intervals from the set $\{1, \ldots, n\}$. If an interval $[a, b]$ is in the TAD set T, then the bases within the corresponding substring physically interact more frequently among one another than with bases from the rest of the genome. Disrupting TAD boundaries can affect the expression of nearby genes, which can trigger diseases such as congenital malformations and cancer [Lupiáñez et al., 2016].

Biological experiments facilitate predicting the location of TADs by measuring the contact frequency of any two locations in the genome [Lieberman-Aiden et al., 2009]. TAD finding algorithms use these contact frequency measurements to identify regions along the genome that are frequently in contact. We denote these contact frequencies using a matrix $M \in \mathbb{R}^{n \times n}$. One common parameterized algorithm for finding TADs [Filippova et al., 2014] returns the set of intervals T that maximizes the objective function

$$\sum_{(i,j) \in T} SM_\gamma(i, j) - \mu_\gamma(j - i), \tag{5}$$

where $\gamma \in \mathbb{R}$ is a parameter,

$$SM_\gamma(i, j) = \frac{1}{(j-i)^\gamma} \sum_{i \leq p < q \leq j} M_{pq},$$

and

$$\mu_\gamma(d) = \frac{1}{n-d} \sum_{t=0}^{n-d} SM_\gamma(t, t+d).$$

Unlike in the sequence alignment and RNA folding algorithms, the parameter γ appears in the exponent of the objective function, thus demonstrating the general applicability of our approach.

We assume the domain expert has access to a training set of DNA strands, each of which has a ground-truth TAD set, which she obtains via hand curation. Our goal is to learn a parameter $\gamma \in \mathbb{R}$ so that given a new DNA strand, the algorithm returns a solution that is close to the unknown, ground-truth TAD set.

To apply our main theorem, we first fix an arbitrary strand S. Since each TAD set is a subset of all possible pairs of locations in the string, there are at most 2^{n^2} possible TAD sets the algorithm might return. In Section 4.1.4, we argue that for any pair of TAD sets, there are $t \leq n^2$ thresholds $\gamma_1, \ldots, \gamma_t \in \mathbb{R}$, on either side of which the TAD set maximizing Equation (5) is invariant. We use this fact to instantiate our general theorem, which guarantees that with probability $1 - \delta$ over the draw of

$$\frac{8}{\epsilon^2}\left(4\left(\ln\left(8en^2\right) + n^2 \ln 4\right) \ln \frac{8}{\epsilon^2} + \frac{1}{4} \ln \frac{1}{\delta}\right)$$

sample problem instances, the average algorithmic performance of any parameter setting over the samples is within ϵ of the parameter setting's expected algorithmic performance. The number of samples sufficient to ensure generalization increases proportionally with n^2.

1.2.2 Applications in economics and political science

A fundamental problem in economics is designing protocols that help groups of agents come to collective decisions. For example, the literature on partnership dissolution [Cramton et al., 1987, McAfee, 1992] investigates questions such as: when a jointly-owned company must be dissolved, which partner should buy the others out, and for how much? When a couple divorces or children inherit an estate, how should they divide the property? How should a town decide which public projects to take on? There is no one policy that best answers these questions; the optimal protocol depends on the setting at hand. For example, splitting a family estate equally may seem "fair", but it may be impossible if the estate is not evenly divisible, and it may not be *efficient* if one family member values the estate much more than another.

In this section, we study an infinite, well-studied family of *mechanisms*, each of which takes as input a set of agents' stated values for each possible outcome and returns one of those outcomes. A mechanism can thus be thought of as an algorithm that the agents use to arrive at a single outcome. This family is known as the class of *neutral affine maximizers (NAMs)* [Roberts, 1979, Mishra and Sen, 2012, Nath and Sandholm, 2019]. There several appealing properties that NAMs satisfy. First, each mechanism in this infinite class is *incentive compatible*, which means that each agent is incentivized to report her values truthfully. In other words, she cannot gain by lying. In order to satisfy incentive compatibility, each agent may have to make a payment in addition to the benefit she accrues or loss she suffers from the mechanism's outcome. Otherwise, the agents could wildly misreport their valuations and suffer no consequences. This raises the question: who should receive this payment? Should it be split evenly among the agents? Should it be discarded? These questions motivate the second property that the class of NAMs satisfies: *budget balance*. A mechanism is budget-balanced if the aggregated payments are somehow distributed among the agents. A line of research [Roberts, 1979, Mishra and Sen, 2012, Nath and Sandholm, 2019] has shown that under natural assumptions, every incentive-compatible, budget-balanced mechanism is a NAM, roughly speaking.

We apply our general theorem in the context of *social welfare* maximization, which is the most widely-studied objective in mechanism design. The social welfare of an outcome is the sum of the agents' values for that outcome. To instantiate our main theorem, we assume that the agents'

values for the outcomes are drawn from an unknown distribution, which is a prevalent assumption throughout the mechanism design literature [Myerson, 1981, Nisan et al., 2007]. Our main theorem allows us to answer the question: how many samples are sufficient to ensure that a NAM with high average social welfare over the samples also has high expected social welfare over the unknown distribution? More formally, suppose there are n agents with values normalized to be in $\left(-\frac{1}{n}, \frac{1}{n}\right)$ over m possible outcomes and let $\epsilon > 0$ be an arbitrary accuracy parameter. Our main theorem implies that with probability $1 - \delta$, if $\hat{M}$ is the NAM with maximum average social welfare over $\frac{32}{\epsilon^2}\left(4\left(n+1\right)\ln\left(4em^2\left(n+1\right)\right)\ln\frac{32}{\epsilon^2} + \frac{1}{4}\ln\frac{1}{\delta}\right)$ samples and M^* is the NAM with maximum expected social welfare, then the expected social welfare of $\hat{M}$ is ϵ-close to the expected social welfare of M^*. We thus obtain strong sample complexity guarantees in a completely different setting from computational biology, the focus of the previous section.

1.2.3 Applications to previously studied domains

Our main theorem recovers sample complexity guarantees from existing literature on data-driven algorithm configuration in the following contexts:

1. Clustering (Section 4.3.1), which is used in many application throughout data science.
2. Tree search (Section 4.3.2), which is used to solve combinatorial optimization problems, integer programs, and constraint satisfaction problems.
3. Canonical subset selection problems (Section 4.3.3), such as the knapsack problem.
4. Revenue maximization problems from economics (Section 4.3.4).

In all of these cases, our main theorem implies sample complexity guarantees that match the existing bounds, but in many cases, our approach provides a more succinct proof. Our results also imply sample complexity bounds for other algorithms with performance that is known to be a piecewise-structured function of the parameters, such as SMAC [Hutter et al., 2011].

1.3 General procedure for applying our main theorem

In this section, we provide a guide to applying our main theorem, which we hope practitioners can use to analyze their own parameterized algorithms in their application domains. In many algorithm configuration problems, for any fixed problem instance, there is a partition of the parameter space into regions where the parameterized algorithm's output is invariant. For example, in sequence alignment, as we range parameters over any one subset of this partition, the algorithm will output the same alignment. This typically means that within these regions, the dual algorithm's performance is a well-structured function. Returning to sequence alignment, if the dual algorithm's performance equals the distance between the alignment it returns and some ground-truth alignment, then the performance function will be constant within each region. As a result, understanding the piecewise decomposability of the dual algorithm family comes down to analyzing this partition, as we describe in the following procedure. (A formal version of this high-level guide appears in Section 4.4.1, together with examples of its application.)

1. Fix an arbitrary problem instance. For example, in the case of pairwise sequence alignment, the problem instance is a pair of sequences.
2. Bound the number of different solutions the algorithm could possibly produce on that instance as the algorithm's parameters are varied. Denote this upper bound by κ. For example, in the

case of sequence alignment, prior research [Gusfield et al., 1994, Fernández-Baca et al., 2004, Pachter and Sturmfels, 2004a] guarantees that for some constant $c \in \mathbb{R}$, $\kappa \leq cn^{3/2}$, where n is an upper bound on the length of the sequences the algorithm is asked to align.

3. For any pair of possible solutions that the parameterized algorithm might produce on that problem instance, identify the set of parameter vectors where the algorithm would choose the first solution over the second. Introduce a function that maps any parameter vector within this set to 1 and any parameter outside this set to 0. This class of functions is denoted by $\mathcal{G}$. Prove an upper bound on the inherent complexity $C_{\mathcal{G}^*}$ of the dual function $\mathcal{G}^*$. In the typical case that the functions in $\mathcal{G}$ are d-dimensional hyperplanes, $C_{\mathcal{G}^*} = d + 1$.

4. Focus on an arbitrary region of the parameter space over which the algorithm's output is invariant. What form does the algorithm's performance take in this region as a function of the algorithm's parameters? (For example in auctions, even if the allocation of goods is considered the output and is constant in a parameter region, the performance in that region in terms of revenue can vary—often linearly—as a function of the auction parameters such as reserve prices.) Denote this class of functions by $\mathcal{F}$. Prove an upper bound on the inherent complexity $C_{\mathcal{F}^*}$ of the dual function $\mathcal{F}^*$. In the typical case that the functions in $\mathcal{F}$ are constant, $C_{\mathcal{F}^*} = 1$, and in the typical case that they are linear functions in d dimensions, $C_{\mathcal{F}^*} = d + 1$.

5. Finally, conclude that this algorithm family is $\left(\mathcal{F}, \mathcal{G}, \kappa^2\right)$-piecewise decomposable, so apply our main theorem with $k = \kappa^2$ and the values $C_{\mathcal{F}^*}$ and $C_{\mathcal{G}^*}$.

2 Notation, problem statement, and tools from learning theory

2.1 Notation

Let $\mathcal{A}$ be an infinite set of algorithms parameterized by a set $\mathcal{P} \subseteq \mathbb{R}^d$ of vectors. Let Π be a set of problem instances for $\mathcal{A}$. We measure the algorithmic performance of the parameter vector $\boldsymbol{\rho} \in \mathcal{P}$ via a utility function $u_{\boldsymbol{\rho}} : \Pi \to [0, H]$, where $u_{\boldsymbol{\rho}}(x)$ measures the performance of the algorithm with parameters $\boldsymbol{\rho} \in \mathcal{P}$ on problem instance $x \in \Pi$, and $H \in \mathbb{R}$ is a bound on the utility function's range. For a fixed problem instance x, we will often analyze an algorithm's utility given x as input as a function of $\boldsymbol{\rho}$, which we denote as $u_x(\boldsymbol{\rho})$.

2.2 Problem statement

We assume there is an application-specific distribution $\mathcal{D}$ over problem instances in Π. Our goal is to find a parameter vector in $\mathcal{P}$ with high performance in expectation over the distribution $\mathcal{D}$. As one step in this process, we analyze the number of samples necessary for *uniform convergence*. Specifically, for any $\epsilon, \delta \in (0, 1)$ and any distribution $\mathcal{D}$ over problem instances, we bound the number m of samples sufficient to ensure that with probability at least $1 - \delta$ over the draw of m samples $\mathcal{S} = \{x_1, \ldots, x_m\} \sim \mathcal{D}^m$, for all parameters $\boldsymbol{\rho} \in \mathcal{P}$, the difference between the average utility of $\boldsymbol{\rho}$ and the expected utility of $\boldsymbol{\rho}$ is at most ϵ: $\left|\frac{1}{m}\sum_{i=1}^{m} u_{\boldsymbol{\rho}}(x_i) - \mathbb{E}_{x\sim\mathcal{D}}[u_{\boldsymbol{\rho}}(x)]\right| \leq \epsilon$. Classic results from learning theory guarantee that if uniform convergence holds and $\hat{\boldsymbol{\rho}}$ is a parameter vector that maximizes average utility over the samples $\left(\hat{\boldsymbol{\rho}} \in \operatorname{argmax}\left\{\frac{1}{m}\sum_{i=1}^{m} u_{\boldsymbol{\rho}}(x_i)\right\}\right)$, then $\hat{\boldsymbol{\rho}}$ is nearly optimal in expectation as well. In particular, with probability at least $1 - \delta$ over the draw $\mathcal{S} \sim \mathcal{D}^m$, $\max_{\rho\in\mathcal{P}} \mathbb{E}_{x\sim\mathcal{D}}[u_{\boldsymbol{\rho}}(x)] - \mathbb{E}_{x\sim\mathcal{D}}[u_{\hat{\boldsymbol{\rho}}}(x)] \leq 2\epsilon$.

2.3 Learning theory tools

Sample complexity tools. Pseudo-dimension [Pollard, 1984] is a well-studied learning-theoretic tool used to measure the complexity of a function class. To formally define pseudo-dimension, we first introduce the notion of *shattering*, which is a fundamental concept in machine learning theory.

Definition 2.1 (Shattering)**.** Let $\mathcal{H} \subseteq [0, H]^{\mathcal{Y}}$ be a set of functions mapping an abstract domain $\mathcal{Y}$ to an interval $[0, H]$. Let $\mathcal{S} = \{y_1, \ldots, y_m\}$ be a subset of $\mathcal{Y}$ and let $z_1, \ldots, z_m \in \mathbb{R}$ be a set of *targets*. We say that $z_1, \ldots, z_m$ *witness* the shattering of $\mathcal{S}$ by $\mathcal{H}$ if for all subsets $T \subseteq \mathcal{S}$, there exists some function $h \in \mathcal{H}$ such that for all elements $y_i \in T$, $h(y_i) \leq z_i$ and for all $x_i \notin T$, $h(y_i) > z_i$.

Definition 2.2 (Pseudo-dimension [Pollard, 1984])**.** Let $\mathcal{H} \subseteq [0, H]^{\mathcal{Y}}$ be a set of functions mapping an abstract domain $\mathcal{Y}$ to an interval $[0, H]$. Let $\mathcal{S} \subseteq \mathcal{Y}$ be a largest set that can be shattered by $\mathcal{H}$. Then $\text{Pdim}(\mathcal{H}) = |\mathcal{S}|$.

When $\mathcal{H}$ is a set of binary valued functions mapping $\mathcal{Y}$ to $\{0, 1\}$, the pseudo-dimension of $\mathcal{H}$ is more commonly referred to as the VC-dimension of $\mathcal{H}$, which we denote as $\text{VCdim}(\mathcal{H})$ [Vapnik and Chervonenkis, 1971].

Theorem 2.1 provides generalization bounds in terms of pseudo-dimension.

Theorem 2.1 (Pollard [1984])**.** *Let $\mathcal{H} \subseteq [0, H]^{\mathcal{Y}}$ be a set of functions mapping an abstract domain $\mathcal{Y}$ to an interval $[0, H]$ and let $d_{\mathcal{H}}$ be the pseudo-dimension of $\mathcal{H}$. For any $\delta \in (0, 1)$ and any distribution $\mathcal{D}$ over $\mathcal{Y}$, with probability at least $1-\delta$ over the draw of m samples $\{y_1, \ldots, y_m\} \sim \mathcal{D}^m$, for any function $h \in \mathcal{H}$, the difference between the average value of h over the samples and the expected value of h is bounded as follows:*

$$\left| \frac{1}{m} \sum_{i=1}^{m} h(y_i) - \mathbb{E}_{y \sim \mathcal{D}}[h(y)] \right| \leq H \sqrt{\frac{2d_{\mathcal{H}}}{m} \ln \frac{em}{d_{\mathcal{H}}}} + H \sqrt{\frac{1}{2m} \ln \frac{1}{\delta}}.$$

Said another way, for any $\epsilon > 0$, $m = \frac{8H^2}{\epsilon^2} \left(d_{\mathcal{H}} \ln \frac{8H^2}{\epsilon^2} + \frac{1}{4} \ln \frac{1}{\delta} \right)$ samples are sufficient to ensure that with probability at least $1 - \delta$ over the draw of m samples $\mathcal{S} = \{y_1, \ldots, y_m\} \sim \mathcal{D}^m$, for all functions $h \in \mathcal{H}$, the difference between the average value of h over the samples and the expected value of h is at most ϵ: $\left| \frac{1}{m} \sum_{i=1}^{m} h(y_i) - \mathbb{E}_{y \sim \mathcal{D}} [h(y)] \right| \leq \epsilon$.

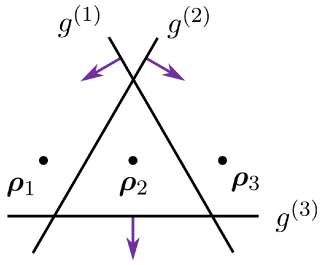
Dual classes. For algorithm configuration problems, there are two closely-related classes of functions. First, for each parameter vector $\boldsymbol{\rho} \in \mathcal{P}$, there is a function $u_{\boldsymbol{\rho}} : \Pi \to [0, H]$ that maps each problem instance x to the utility of the algorithm with parameter $\boldsymbol{\rho}$ given x as input. Similarly, for each problem instance $x \in \Pi$, there is a function $u_x : \mathcal{P} \to [0, H]$ defined as $u_x(\boldsymbol{\rho}) = u_{\boldsymbol{\rho}}(x)$ that fixes the problem instance x and allows the algorithm parameter vector $\boldsymbol{\rho}$ to vary. Our main theorem revolves around the relationship between these two types of functions. In learning theory, the set of functions $\{u_x : \mathcal{P} \to [0, H] \mid x \in \Pi\}$ is equivalent to what is known as the *dual class*, which we define abstractly below.

Definition 2.3 (Dual class [Assouad, 1983])**.** For any domain $\mathcal{Y}$ and set of functions $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{Y}}$, the *dual class* of $\mathcal{H}$ is defined as $\mathcal{H}^* = \left\{ h_y^* : \mathcal{H} \to \mathbb{R} \mid y \in \mathcal{Y} \right\}$ where $h_y^*(h) = h(y)$. Each function $h_y^* \in \mathcal{H}^*$ fixes an input $y \in \mathcal{Y}$ and maps each function $h \in \mathcal{H}$ to $h(y)$. We refer to the class $\mathcal{H}$ as the *primal class.*

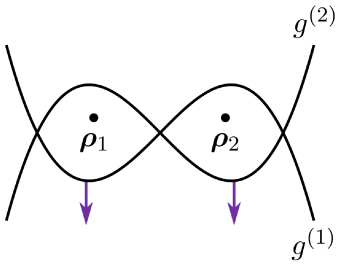
The set of functions $\{u_x : \mathcal{P} \to [0, H] \mid x \in \Pi\}$ is equivalent to the dual class

$$\mathcal{U}^* = \{u_x^* : \mathcal{U} \to [0, H] \mid x \in \Pi\}$$

in the sense that for every parameter vector $\boldsymbol{\rho} \in \mathcal{P}$ and every problem $x \in \Pi$, $u_x(\boldsymbol{\rho}) = u_x^*(u_{\boldsymbol{\rho}})$.



(a) The arrows indicate on which side of each linear separator $g^{(i)}$ we have that $g^{(i)}(\boldsymbol{\rho}) = 0$ and on which side $g^{(i)}(\boldsymbol{\rho}) = 1$. For example, $g^{(1)}(\boldsymbol{\rho}_1) = 1$, $g^{(1)}(\boldsymbol{\rho}_2) = 1$, and $g^{(1)}(\boldsymbol{\rho}_3) = 0$.



(b) The arrows indicate on which side of each polynomial separator $g^{(i)}$ we have that $g^{(i)}(\boldsymbol{\rho}) = 0$ and on which side $g^{(i)}(\boldsymbol{\rho}) = 1$. For example, $g^{(1)}(\boldsymbol{\rho}_1) = 1$ and $g^{(1)}(\boldsymbol{\rho}_2) = 0$.

Figure 5: Figures 5a and 5b illustrate boundary functions partitioning $\mathbb{R}^2$.

3 General theorem

As we have shown, a large number of algorithm configuration problems share a clear-cut, useful structure: for each problem instance $x \in \Pi$, the function u_x—which measures the algorithm's utility given the input x as a function of the parameters $\boldsymbol{\rho}$—is a piecewise structured. For example, each function u_x might be piecewise constant with a small number of pieces. Given the equivalence of the functions $\{u_x \mid x \in \Pi\}$ and the dual class $\mathcal{U}^*$, the dual class exhibits this piecewise structure as well. We use this piecewise structure of the dual class to bound the pseudo-dimension of the primal class $\mathcal{U}$. We then apply Theorem 2.1 to bound the number of samples sufficient to ensure that uniform convergence holds.

Before stating our main result, we introduce notation that we will use to more formally define the notion of piecewise-structured functions. Let $h : \mathcal{C} \to \mathbb{R}$ be a function mapping an abstract domain $\mathcal{C}$ to the real line. Intuitively, the function h is piecewise structured if we can partition the domain $\mathcal{C}$ into subsets $\mathcal{C}_1, \ldots, \mathcal{C}_N$ such that when we restrict h to a single piece $\mathcal{C}_i$, h equals some function $f : \mathcal{C} \to \mathbb{R}$. In other words, for all $c \in \mathcal{C}_i$, $h(c) = f(c)$. We describe the partition $\mathcal{C}_1, \ldots, \mathcal{C}_N$ using a collection of *boundary functions* $g^{(1)}, \ldots, g^{(k)} : \mathcal{C} \to \{0, 1\}$. Each boundary function $g^{(i)}$ divides the domain $\mathcal{C}$ into two sets: the points it labels 0 and the points it labels 1. Figure 5 illustrates two partitions of $\mathbb{R}^2$ by boundary functions. Together, the k boundary functions partition the domain $\mathcal{C}$ into at most 2^k regions, each one corresponding to a bit vector $\boldsymbol{b} \in \{0, 1\}^k$ describing on which side of each boundary the region belongs. For each region, we specify a *piece function* $f_{\boldsymbol{b}} : \mathcal{C} \to \mathbb{R}$ defining the function values of h restricted to that region, where $\boldsymbol{b} \in \{0, 1\}^k$ is the bit vector describing the region. More formally, the function h can be written as $h(c) = f_{\boldsymbol{b}_c}(c)$, where $\boldsymbol{b}_{\boldsymbol{c}} = \left(g^{(1)}(c), \ldots, g^{(k)}(c)\right) \in \{0, 1\}^k$ is the bit vector identifying the region in the partition containing the element c. Figure 1 shows an example of a piecewise-structured function with two boundary functions and four piece functions.

In many algorithm configuration problems, every function in the dual class is piecewise structured. Moreover, across dual functions, the corresponding boundary functions come from a single, fixed class, as do the piece functions. For example, the boundary functions might always be halfspace indicator functions, while the piece functions might always be linear functions. The following definition formalizes this structure.

Definition 3.1 ($(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable)**.** A class of functions $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{C}}$ mapping some domain $\mathcal{C}$ to $\mathbb{R}$ is *$(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable* for a class $\mathcal{G} \subseteq \{0, 1\}^{\mathcal{C}}$ of boundary functions and a class $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{C}}$ of piece functions if the following holds: for every function $h \in \mathcal{H}$, there exist

k boundary functions $g^{(1)}, \ldots, g^{(k)} \in \mathcal{G}$ and a piece function $f_{\boldsymbol{b}} \in \mathcal{F}$ for each bit vector $\boldsymbol{b} \in \{0,1\}^k$ such that for all $c \in \mathcal{C}$

$$h(c) = f_{\boldsymbol{b}_c}(c) \qquad \text{where} \qquad \boldsymbol{b}_c = \left(g^{(1)}(c), \ldots, g^{(k)}(c)\right) \in \{0,1\}^k.$$

Our main theorem shows that whenever a class $\mathcal{Q}$ of functions has a $(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable dual function class $\mathcal{Q}^*$, we can bound the pseudo-dimension of $\mathcal{Q}$ in terms of the VC-dimension of $\mathcal{G}^*$ and the pseudo-dimension of $\mathcal{F}^*$. In other words, if the boundary and piece functions both have dual classes with low complexity, then the pseudo-dimension of $\mathcal{Q}$ is small. In Section 3.1, we show that for many common boundary and piece classes $\mathcal{F}$ and $\mathcal{G}$, we can easily bound the complexity of their dual classes.

Throughout the proof, it may be useful to relate the concepts we discuss to the algorithm configuration setting, in which case $\mathcal{Q}$ equals the function class $\{u_{\boldsymbol{\rho}} \mid \boldsymbol{\rho} \in \mathcal{P}\}$. In that setting, every function in $\mathcal{Q}$ is defined by a set of parameters $\boldsymbol{\rho}$ and maps problem instances x to real-valued utilities $u_{\boldsymbol{\rho}}(x)$. Moreover, the dual class $\mathcal{Q}^*$ is equivalent to the function class $\{u_x \mid x \in \Pi\}$. Every function in $\mathcal{Q}^*$ is defined by a problem instance x and maps parameters $\boldsymbol{\rho}$ to utilities $u_x(\boldsymbol{\rho}) = u_{\boldsymbol{\rho}}(x)$.

Theorem 3.1 (Main sample complexity theorem)**.** *Let $\mathcal{Q} \subseteq \mathbb{R}^{\mathcal{Y}}$ be a class of functions mapping an abstract domain $\mathcal{Y}$ to the real line. Suppose that the dual function class $\mathcal{Q}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-decomposable with boundary functions $\mathcal{G} \subseteq \{0,1\}^{\mathcal{Q}}$ and piece functions $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{Q}}$. Denote the VC-dimension of $\mathcal{G}^*$ as $d_{\mathcal{G}^*}$ and the pseudo-dimension of $\mathcal{F}^*$ as $d_{\mathcal{F}^*}$. The pseudo-dimension of $\mathcal{Q}$ is bounded as follows:*

$$\operatorname{Pdim}(\mathcal{Q}) \leq 4\left(d_{\mathcal{F}^*} + d_{\mathcal{G}^*}\right) \ln\left(4ek\left(d_{\mathcal{F}^*} + d_{\mathcal{G}^*}\right)\right).$$

Proof. Fix any set of points $y_1, \ldots, y_D \in \mathcal{Y}$ and targets $z_1, \ldots, z_D \in \mathbb{R}$. We will bound the number of ways that $\mathcal{Q}$ can label the points $y_1, \ldots, y_D$ with respect to the target thresholds $z_1, \ldots, z_D$ by $(ekD)^{d_{\mathcal{G}^*}}(eD)^{d_{\mathcal{F}^*}}$. Then solving for the largest D such that $2^D \leq (ekD)^{d_{\mathcal{G}^*}}(eD)^{d_{\mathcal{F}^*}}$ gives a bound on the pseudo-dimension of $\mathcal{Q}$. Our bound on the number of ways that $\mathcal{Q}$ can label $y_1, \ldots, y_D$ has two main steps:

1. In Claim 3.2, we show that there are $M < (ekD)^{d_{\mathcal{G}^*}}$ subsets $\mathcal{Q}_1, \ldots, \mathcal{Q}_M$ partitioning the function class $\mathcal{Q}$ such that within any one subset, the dual functions $q^*_{y_1}, \ldots, q^*_{y_D}$ are simultaneously structured[2]. In particular, for each subset $\mathcal{Q}_j$, there exist piece functions $f_1, \ldots, f_D \in \mathcal{F}$ such that $q^*_{y_i}(q) = f_i(q)$ for all $q \in \mathcal{Q}_j$ and $i \in [D]$. This is the partition of $\mathcal{Q}$ induced by aggregating all of the boundary functions corresponding to the dual functions $q^*_{y_1}, \ldots, q^*_{y_D}$.

2. In Claim 3.3, we show that the functions belonging to any single subset $\mathcal{Q}_i$ in the partition constructed in Claim 3.2 can label the points $y_1, \ldots, y_D$ in at most $(eD)^{d_{\mathcal{F}^*}}$ ways. It follows that the total number of ways that $\mathcal{Q}$ can label the points $y_1, \ldots, y_D$ is bounded by $(ekD)^{d_{\mathcal{G}^*}}(eD)^{d_{\mathcal{F}^*}}$.

Assuming the dual function class $\mathcal{Q}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-decomposable with boundary functions $\mathcal{G} \subseteq \{0,1\}^{\mathcal{Q}}$ and piece functions $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{Q}}$, we now prove our first claim.

Claim 3.2. *There are $M < (ekD)^{d_{\mathcal{G}^*}}$ subsets $\mathcal{Q}_1, \ldots, \mathcal{Q}_M$ partitioning the function class $\mathcal{Q}$ such that within any one subset, the dual functions $q^*_{y_1}, \ldots, q^*_{y_D}$ are simultaneously structured. In particular, for each subset $\mathcal{Q}_j$, there exist piece functions $f_1, \ldots, f_D \in \mathcal{F}$ such that $q^*_{y_i}(q) = f_i(q)$ for all $q \in \mathcal{Q}_j$ and $i \in [D]$.*

[2]We can relate this step to the algorithm configuration setting as follows. Since every function in $\mathcal{Q}$ is defined by a parameter vector, partitioning $\mathcal{Q}$ is equivalent to partitioning the parameter space. In this step, we show that we can partition the parameter space into regions where the dual functions are piecewise structured functions of the parameters.

Proof of Claim 3.2. Let $q^*_{y_1}, \ldots, q^*_{y_D} \in \mathcal{Q}^*$ be the dual functions corresponding to the points $y_1, \ldots, y_D$. Since $\mathcal{Q}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable, we know that for each function $q^*_{y_i}$, there are k boundary functions $g_i^{(1)}, \ldots, g_i^{(k)} \in \mathcal{G} \subseteq \{0,1\}^{\mathcal{Q}}$ that define its piecewise decomposition. Let $\hat{\mathcal{G}} = \bigcup_{i=1}^{D} \left\{ g_i^{(1)}, \ldots, g_i^{(k)} \right\}$ be the union of these boundary functions across all $i \in [D]$. For ease of notation, we relabel the functions in $\hat{\mathcal{G}}$, calling them $g_1, \ldots, g_{kD}$. Let M be the total number of kD-dimensional vectors we can obtain by applying the functions in $\hat{\mathcal{G}} \subseteq \{0,1\}^{\mathcal{Q}}$ to elements of $\mathcal{Q}$:

$$M := \left| \left\{ \begin{pmatrix} g_1(q) \\ \vdots \\ g_{kD}(q) \end{pmatrix} : q \in \mathcal{Q} \right\} \right|. \tag{6}$$

By definition of the dual class $\mathcal{G}^*$, we know that $g_i(q) = g^*_q(g_i)$ for every function $g_i \in \hat{\mathcal{G}}$ and element $q \in \mathcal{Q}$, which means that

$$M = \left| \left\{ \begin{pmatrix} g^*_q(g_1) \\ \vdots \\ g^*_q(g_{kD}) \end{pmatrix} : q \in \mathcal{Q} \right\} \right|.$$

Therefore, M equals the number of distinct ways that functions in $\mathcal{G}^*$ can label the functions $g_1, \ldots, g_{kD}$. Sauer's Lemma[3] guarantees that $\mathcal{G}^*$ cannot label kD points in $\mathcal{G}$ in too many ways, leading to a bound on M [Sauer, 1972]. Specifically, $M \leq \left(\frac{ekD}{d_{\mathcal{G}^*}}\right)^{d_{\mathcal{G}^*}} < (ekD)^{d_{\mathcal{G}^*}}$.

Finally, let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ be the binary vectors in the set from Equation (6). For each $i \in [M]$, let $\mathcal{Q}_i = \{q \in \mathcal{Q} \mid (g_1(q), \ldots, g_{kD}(q)) = \boldsymbol{b}_i\}$. For each set $\mathcal{Q}_i$, the value of all the boundary functions $g_1, \ldots, g_{kD}$ is constant, so there is a fixed set of piece functions $f_1, \ldots, f_D \in \mathcal{F}$ so that $q^*_{y_i}(q) = f_i(q)$ for all elements $q \in \mathcal{Q}_i$ and indices $i \in [D]$. Therefore, the lemma statement holds. □

We now show that each subset $\mathcal{Q}_i$ can label the points $y_1, \ldots, y_D$ in at most $(eD)^{d_{\mathcal{F}^*}}$ ways.

Claim 3.3. *For each subset $\mathcal{Q}_i$ in the partition defined by Claim 3.2, we have*

$$\left| \left\{ \begin{pmatrix} \operatorname{sign}(q(y_1) - z_1) \\ \vdots \\ \operatorname{sign}(q(y_D) - z_D) \end{pmatrix} \middle| \, q \in \mathcal{Q}_i \right\} \right| \leq (eD)^{d_{\mathcal{F}^*}}.$$

In other words, functions from $\mathcal{Q}_i$ can label the points $y_1, \ldots, y_D$ in at most $(eD)^{d_{\mathcal{F}^*}}$ distinct ways relative to the targets $z_1, \ldots, z_D$.

Proof of Claim 3.3. Let $q^*_{y_1}, \ldots, q^*_{y_D} \in \mathcal{Q}^*$ be the dual functions corresponding to the points $y_1, \ldots, y_D$. From Claim 3.2, there exist piece functions $f_1, \ldots, f_D \in \mathcal{F}$ such that for all $q \in \mathcal{Q}_i$ and $j \in [D]$, we have $q^*_{y_j}(q) = f_j(q)$. For all $q \in \mathcal{Q}_i$ and $j \in [D]$, using the definition of the dual of q and f_j, we have

$$q(y_j) = q^*_{y_j}(q) = f_j(q) = f^*_q(f_j).$$

[3]Sauer's lemma applies to the following generic setting: There is a set of n datapoints $t_1, ..., t_n$ from the domain of some function class $\mathcal{R}$. For each function $r \in \mathcal{R}$, we define a vector $(r(t_1), \cdots, r(t_n))$. We then define a set of vectors by unioning over all $r \in \mathcal{R}$. Sauer's lemma says this set cannot be too big. Sauer's lemma does not immediately imply M (in Equation (6)) is bounded: Each component is defined by a different function (rather than a different datapoint), and we obtain the set of vectors by unioning over all datapoints (rather than unioning over all functions). This is why we need to transition to the dual class $\mathcal{G}^*$ in order to bound M.

Therefore, we can rewrite the set of labelings of $y_1, \ldots, y_D$ by functions in $\mathcal{Q}_i$ as follows:

$$\left\{ \begin{pmatrix} \operatorname{sign}\left(q\left(y_1\right) - z_1\right) \\ \vdots \\ \operatorname{sign}\left(q\left(y_D\right) - z_D\right) \end{pmatrix} \middle| \, q \in \mathcal{Q}_i \right\} = \left\{ \begin{pmatrix} \operatorname{sign}\left(f_q^*(f_1) - z_1\right) \\ \vdots \\ \operatorname{sign}\left(f_q^*(f_D) - z_D\right) \end{pmatrix} \middle| \, q \in \mathcal{Q}_i \right\}. \tag{7}$$

The right hand side of Equation (3) is the set of labelings of the functions $f_1, \ldots, f_D$ by the dual functions $\{f_q^* \mid q \in \mathcal{Q}_i\} \subset \mathcal{F}^*$ with respect to the targets $z_1, \ldots, z_D$. Finally, from Sauer's Lemma, we know that the number of possible labelings of $f_1, \ldots, f_D$ possible by $\mathcal{F}^*$ is at most $\left(\frac{eD}{d_{\mathcal{F}^*}}\right)^{d_{\mathcal{F}^*}} \leq (eD)^{d_{\mathcal{F}^*}}$, which proves the claim. □

As a consequence of the above claims, we know that $\mathcal{Q}$ can label the points $y_1, \ldots, y_D$ in at most $(ekD)^{d_{\mathcal{G}^*}}(eD)^{d_{\mathcal{F}^*}}$ distinct ways relative to the targets $z_1, \ldots, z_D$. On the other hand, if $\mathcal{Q}$ shatters the points $y_1, \ldots, y_D$, then the number of distinct labelings must be 2^D. Therefore, the pseduo-dimension of $\mathcal{Q}$ is at most the largest value of D such that $2^D \leq (ekD)^{d_{\mathcal{G}^*}}(eD)^{d_{\mathcal{F}^*}}$. Taking the log of both sides and rearranging gives $D < \left(d_{\mathcal{F}^*} + d_{\mathcal{G}^*}\right) \ln D + d_{\mathcal{F}^*} + d_{\mathcal{G}^*} \ln(ek)$. By Lemma 3.4, this implies that $D < 2d_{\mathcal{F}^*} \ln\left(4e\left(d_{\mathcal{F}^*} + d_{\mathcal{G}^*}\right)^2\right) + 2d_{\mathcal{G}^*} \ln\left(4ek\left(d_{\mathcal{F}^*} + d_{\mathcal{G}^*}\right)^2\right)$. Therefore, the pseudo-dimension of $\mathcal{Q}$ is at most $2d_{\mathcal{F}^*} \ln\left(4e\left(d_{\mathcal{F}^*} + d_{\mathcal{G}^*}\right)^2\right) + 2d_{\mathcal{G}^*} \ln\left(4ek\left(d_{\mathcal{F}^*} + d_{\mathcal{G}^*}\right)^2\right) \leq 4\left(d_{\mathcal{F}^*} + d_{\mathcal{G}^*}\right) \ln\left(4ek\left(d_{\mathcal{F}^*} + d_{\mathcal{G}^*}\right)\right)$. □

Lemma 3.4 (Shalev-Shwartz and Ben-David [2014])**.** *Let $a \geq 1$ and $b > 0$. Then $y < a \ln y + b$ implies that $y < 4a \ln(2a) + 2b$.*

3.1 Applications of our main theorem to representative function classes

In this section, we instantiate our main result, Theorem 3.1, in settings inspired by algorithm configuration problems.

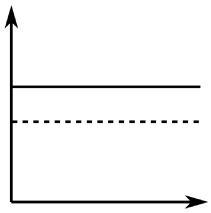
3.1.1 One-dimensional functions with a bounded number of oscillations

Let $\mathcal{U} = \{u_\rho \mid \rho \in \mathcal{P} \subseteq \mathbb{R}\}$ be a class of utility functions defined over a single-dimensional parameter space. We often find that the dual class contains functions that are piecewise constant, linear, or polynomial in the parameter. More generally, the functions in the dual class are piecewise-structured, and we can guarantee that the structured functions oscillate a fixed number of times. In the language of decomposability, this means that the dual function $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-decomposable, where the boundary functions $\mathcal{G} \subseteq \{0,1\}^{\mathcal{U}}$ are thresholds and the piece functions $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{U}}$ oscillate a bounded number of times, as we formalize below.

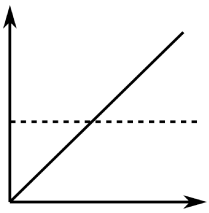
Definition 3.2. We say that a function $h : \mathbb{R} \to \mathbb{R}$ has at most B oscillations if for every $z \in \mathbb{R}$, the function $\rho \mapsto \mathbb{I}_{\{h(\rho) \geq z\}}$ is piecewise constant with at most B discontinuities.

For example, constant functions have zero oscillations (see Figure 6a), linear functions have one oscillation (see Figure 6b), and inverse-quadratic functions (of the form $h(x) = \frac{a}{x^2} + bx + c$) have at most two oscillations (see Figure 6c). Throughout our applications, we analyze piecewise-structured functions whose the piece functions come from these three families (see Section 4). In the following lemma, we bound the pseudo-dimension of classes with bounded oscillations.

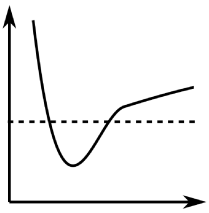
Lemma 3.5. *Let $\mathcal{H}$ be a class of functions mapping $\mathbb{R}$ to $\mathbb{R}$, each of which has at most B oscillations. Then* $\operatorname{Pdim}(\mathcal{H}^*) < 4 \ln(256(B+1))$.



(a) Constant function (zero oscillations).



(b) Linear function (one oscillation).



(c) Inverse-quadratic function (two oscillations).

Figure 6: Examples of functions with bounded oscillations. Each solid line is a function with bounded oscillations and each dotted line represents an arbitrary threshold.

Proof. Suppose that $\text{Pdim}(\mathcal{H}^*) = D$. Then there exist functions $h_1, \ldots, h_D \in \mathcal{H}$ and witnesses $z_1, \ldots, z_D \in \mathbb{R}$ such that for every subset $T \subseteq [D]$, there exists a parameter $\rho \in \mathbb{R}$ such that $h^*_\rho(h_i) \geq z_i$ if and only if $i \in T$. We can simplify notation as follows: since $h(\rho) = h^*_\rho(h)$ for every function $h \in \mathcal{H}$, we have that for every subset $T \subseteq [D]$, there exists a parameter $\rho \in \mathbb{R}$ such that $h_i(\rho) \geq z_i$ if and only if $i \in T$. Let $\mathcal{P}^*$ be the set of 2^D parameters corresponding to each subset $T \subseteq [D]$. By definition, these parameters induce 2^D distinct binary vectors as follows:

$$\left| \left\{ \begin{pmatrix} \mathbb{I}_{\{h_1(\rho) \geq z_1\}} \\ \vdots \\ \mathbb{I}_{\{h_D(\rho) \geq z_D\}} \end{pmatrix} : \rho \in \mathcal{P}^* \right\} \right| = 2^D.$$

On the other hand, since each function h_i has at most B oscillations, we can partition $\mathbb{R}$ into $M \leq BD + 1$ intervals $I_1, \ldots, I_M$ such that for every interval I_j and every $i \in [D]$, the function $\rho \mapsto \mathbb{I}_{\{h_i(\rho) \geq z_i\}}$ is constant across the interval I_j. Therefore, at most one parameter $\rho \in \mathcal{P}^*$ can fall within a single interval I_j. Otherwise, if $\rho, \rho' \in I_j \cap \mathcal{P}^*$, then

$$\begin{pmatrix} \mathbb{I}_{\{h_1(\rho) \geq z_1\}} \\ \vdots \\ \mathbb{I}_{\{h_D(\rho) \geq z_D\}} \end{pmatrix} = \begin{pmatrix} \mathbb{I}_{\{h_1(\rho') \geq z_1\}} \\ \vdots \\ \mathbb{I}_{\{h_D(\rho') \geq z_D\}} \end{pmatrix},$$

which is a contradiction. As a result, $2^D \leq BD + 1 \leq BD + D$. Therefore, $D \leq \frac{1}{\ln 2}(\ln D + \ln(B+1)) < 2(\ln D + \ln(B+1))$. By Lemma 3.4, we conclude that $D < 8 \ln 16 + 4 \ln(B+1) = 4 \ln(256(B+1))$. □

Lemma 3.5 implies the following pseudo-dimension bound for the case where the dual function class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-decomposable, where the boundary functions $\mathcal{G} \subseteq \{0,1\}^{\mathcal{U}}$ are thresholds and the piece functions $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{U}}$ oscillate a bounded number of times.

Corollary 3.6. *Let $\mathcal{U} = \{u_\rho \mid \rho \in \mathcal{P} \subseteq \mathbb{R}\}$ be a class of utility functions defined over a single-dimensional parameter space. Suppose the dual function $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-decomposable, where the boundary functions $\mathcal{G} = \{f_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ are thresholds $g_a : u_\rho \mapsto \mathbb{I}_{\{a \leq \rho\}}$. Moreover, suppose that for each function $f \in \mathcal{F}$, the function $\rho \mapsto f(u_\rho)$ has at most B oscillations. Then* $\text{Pdim}(\mathcal{U}) < 4(4 \ln(256(B+1)) + 1) \ln(4ek(4 \ln(256(B+1)) + 1))$.

Proof. First, we claim that $\text{VCdim}(\mathcal{G}^*) = 1$. For a contradiction, suppose $\mathcal{G}^*$ can shatter two functions $g_a, g_b \in \mathcal{G}^*$, where $a < b$. There must exist a parameter $\rho \in \mathbb{R}$ such that $g^*_{u_\rho}(g_a) =$

$g_a(u_\rho) = \mathbb{I}_{\{a \le \rho\}} = 0$ and $g^*_{u_\rho}(g_b) = g_b(u_\rho) = \mathbb{I}_{\{b \le \rho\}} = 1$. Therefore, $b \le \rho < a$, which is a contradiction, so $\text{VCdim}(\mathcal{G}^*) = 1$.

Next, we claim that $\text{Pdim}(\mathcal{F}^*) < 4\ln(256(B+1))$. For each function $f \in \mathcal{F}$, let $h_f : \mathbb{R} \to \mathbb{R}$ be defined as $h_f(\rho) = f(u_\rho)$. By assumption, each function h_f has at most B oscillations. Let $\mathcal{H} = \{h_f \mid f \in \mathcal{F}\}$ and let $D = \text{Pdim}(\mathcal{H}^*)$. By Lemma 3.5, we know that $D < 4\ln(256(B+1))$. We claim that $\text{Pdim}(\mathcal{H}^*) \ge \text{Pdim}(\mathcal{F}^*)$. For a contradiction, suppose the class $\mathcal{F}^*$ can shatter $D+1$ points $f_1, \ldots, f_{D+1}$ using witnesses $z_1, \ldots, z_{D+1} \in \mathbb{R}$. By definition, this means that

$$\left| \left\{ \begin{pmatrix} \mathbb{I}_{\left\{f^*_{u_\rho}(f_1) \ge z_1\right\}} \\ \vdots \\ \mathbb{I}_{\left\{f^*_{u_\rho}(f_{D+1}) \ge z_{D+1}\right\}} \end{pmatrix} : \rho \in \mathcal{P} \right\} \right| = 2^{D+1}.$$

For any function $f \in \mathcal{F}$ and any parameter $\rho \in \mathbb{R}$, $f^*_{u_\rho}(f) = f(u_\rho) = h_f(\rho) = h^*_\rho(h_f)$. Therefore,

$$\left| \left\{ \begin{pmatrix} \mathbb{I}_{\left\{h^*_\rho(h_{f_1}) \ge z_1\right\}} \\ \vdots \\ \mathbb{I}_{\left\{h^*_\rho\left(h_{f_{D+1}}\right) \ge z_{D+1}\right\}} \end{pmatrix} : \rho \in \mathcal{P} \right\} \right| = \left| \left\{ \begin{pmatrix} \mathbb{I}_{\left\{f^*_{u_\rho}(f_1) \ge z_1\right\}} \\ \vdots \\ \mathbb{I}_{\left\{f^*_{u_\rho}(f_{D+1}) \ge z_{D+1}\right\}} \end{pmatrix} : \rho \in \mathcal{P} \right\} \right| = 2^{D+1},$$

which contradicts the fact that $\text{Pdim}(\mathcal{H}^*) = D$. Therefore, $\text{Pdim}(\mathcal{F}^*) \le D < 4\ln(256(B+1))$. The corollary then follows from Theorem 3.1. □

3.1.2 Multi-dimensional piecewise linear functions

Generalizing to multi-dimensional parameter spaces, we often find that the boundary functions correspond to halfspace thresholds and the piece functions correspond to constant or linear functions. We handle this case in the following lemma.

Lemma 3.7. *Let $\mathcal{U} = \left\{u_{\boldsymbol{\rho}} \mid \boldsymbol{\rho} \in \mathcal{P} \subseteq \mathbb{R}^d\right\}$ be a class of utility functions defined over a d-dimensional parameter space. Suppose the dual function $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-decomposable, where the boundary functions $\mathcal{G} = \left\{f_{\boldsymbol{a},\theta} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^d, \theta \in \mathbb{R}\right\}$ are halfspace thresholds $g_{\boldsymbol{a},\theta} : u_{\boldsymbol{\rho}} \mapsto \mathbb{I}_{\{\boldsymbol{a}\cdot\boldsymbol{\rho} \le \theta\}}$ and the piece functions $\mathcal{F} = \left\{f_{\boldsymbol{a},\theta} : \mathcal{U} \to \mathbb{R} \mid \boldsymbol{a} \in \mathbb{R}^d, \theta \in \mathbb{R}\right\}$ are linear functions $f_{\boldsymbol{a},\theta} : u_{\boldsymbol{\rho}} \mapsto \boldsymbol{a}\cdot\boldsymbol{\rho} + \theta$. Then $\text{Pdim}(\mathcal{U}) \le 8(d+1)\ln(8ek(d+1))$.*

Proof. First, we prove that the VC-dimension of the dual class $\mathcal{G}^*$ is at most $d+1$. The dual class $\mathcal{G}^*$ consists of functions $g^*_{u_{\boldsymbol{\rho}}}$ for all $\boldsymbol{\rho} \in \mathcal{P}$ where $g^*_{u_{\boldsymbol{\rho}}}(g_{\boldsymbol{a},\theta}) = \mathbb{I}_{\{\boldsymbol{a}\cdot\boldsymbol{\rho} \le \theta\}}$. Let $\hat{\mathcal{G}} = \left\{\hat{g}_{\boldsymbol{\rho}} : \mathbb{R}^{d+1} \to \{0,1\}\right\}$ be the class of halfspace thresholds $\hat{g}_{\boldsymbol{\rho}} : (\boldsymbol{a}, \theta) \mapsto \mathbb{I}_{\{\boldsymbol{a}\cdot\boldsymbol{\rho} \le \theta\}}$. It is well-known that $\text{VCdim}\left(\hat{\mathcal{G}}\right) \le d+1$, which we prove means that $\text{VCdim}(\mathcal{G}^*) \le d+1$. For a contradiction, suppose $\mathcal{G}^*$ can shatter $d+2$ functions $g_{\boldsymbol{a}_1,\theta_1}, \ldots, g_{\boldsymbol{a}_{d+2},\theta_{d+2}} \in \mathcal{G}$. Then for every subset $T \subseteq [d+2]$, there exists a parameter vector $\boldsymbol{\rho}_T$ such that $\boldsymbol{a}_i \cdot \boldsymbol{\rho}_T \le \theta_i$ if and only if $i \in T$. This means that $\hat{\mathcal{G}}$ can shatter the tuples $(\boldsymbol{a}_1, \theta_1), \ldots, (\boldsymbol{a}_{d+2}, \theta_{d+2})$ as well, which contradicts the fact that $\text{VCdim}\left(\hat{\mathcal{G}}\right) \le d+1$. Therefore, $\text{VCdim}(\mathcal{G}^*) \le d+1$.

By a similar argument, we prove that the pseudo-dimension of the dual class $\mathcal{F}^*$ is at most $d+1$. The dual class $\mathcal{F}^*$ consists of functions $f^*_{u_{\boldsymbol{\rho}}}$ for all $\boldsymbol{\rho} \in \mathcal{P}$ where $f^*_{u_{\boldsymbol{\rho}}}(f_{\boldsymbol{a},\theta}) = \boldsymbol{a}\cdot\boldsymbol{\rho} + \theta$. Let $\hat{\mathcal{F}} = \left\{\hat{f}_{\boldsymbol{\rho}} : \mathbb{R}^{d+1} \to \mathbb{R}\right\}$ be the class of linear functions $\hat{f}_{\boldsymbol{\rho}} : (\boldsymbol{a}, \theta) \mapsto \boldsymbol{a}\cdot\boldsymbol{\rho} + \theta$. It is well-known that $\text{Pdim}\left(\hat{\mathcal{F}}\right) \le d+1$, which we prove means that $\text{Pdim}(\mathcal{F}^*) \le d+1$. For a contradiction, suppose

$\mathcal{F}^*$ can shatter $d+2$ functions $f_{\boldsymbol{a}_1,\theta_1},\ldots,f_{\boldsymbol{a}_{d+2},\theta_{d+2}} \in \mathcal{F}$. Then there exist witnesses $z_1,\ldots,z_{d+2}$ such that for every subset $T \subseteq [d+2]$, there exists a parameter vector $\boldsymbol{\rho}_T$ such that $\boldsymbol{a}_i \cdot \boldsymbol{\rho}_T + \theta_i \leq z_i$ if and only if $i \in T$. This means that $\hat{\mathcal{F}}$ can shatter the tuples $(\boldsymbol{a}_1,\theta_1),\ldots,(\boldsymbol{a}_{d+2},\theta_{d+2})$ as well, which contradicts the fact that $\text{Pdim}\left(\hat{\mathcal{F}}\right) \leq d+1$. Therefore, $\text{Pdim}\left(\mathcal{F}^*\right) \leq d+1$.

The lemma statement now follows from Theorem 3.1. □

4 Applications

In this section, we apply our main sample complexity guarantee to parameterized algorithms ranging from computational biology to algorithmic economics. In Section 4.4.1, we also present a generic recipe for applying this theorem that we hope practitioners can use for their own tunable algorithms (see Remark 4.1).

4.1 Applications in biology

In this section, we instantiate Theorem 3.1 in three diverse applications from computational biology: sequence alignment, RNA folding, and finding Topologically Associated Domains (TADs).

4.1.1 Global pairwise sequence alignment

Pairwise sequence alignment is a fundamental problem in biological sequence analysis; database search, where the goal is to find a given query in a larger text [Altschul et al., 1990]; homology detection, where given two sequences the goal is to find the locations that are analogous [Patthy, 1987]; and many other scientific domains. Pairwise alignment is also a basic operation in many tools for multiple sequence alignment, where the objective is to find correlation between at least three strings [Sankoff and Cedergren, 1983], which we analyze in Section 4.1.2. Depending on the application, the goal may be to find a complete alignment of the two sequences, called *global alignment*, or to find the best alignment of any subsequences of the two input sequences, called *local alignment*. In either case, the high level goal is the same: given two sequences, find a two-dimensional grid, where each row of the grid corresponds to one of the two sequences with inserted gap characters, that optimizes a given parameterized objective function. While the pairwise sequence alignment problem in general is well studied, most common problem formulations have the same issue: it is a challenge to best select the objective function's parameters. Depending on the application domain, the best parameter values differ between instances greatly.

More formally, let Σ be an alphabet and let S_1 and S_2 be two sequences in Σ of length n. A *sequence alignment* is a pair of sequences $\tau_1,\tau_2 \in (\Sigma \cup \{-\})^*$ such that $|\tau_1| = |\tau_2|$, $\texttt{del}(\tau_1) = S_1$, and $\texttt{del}(\tau_2) = S_2$, where $\texttt{del}$ is a function that deletes every $-$, or *gap character*, in the input sequence. We require that a gap character is never paired with a gap character: for all $i \in [|\tau_1|]$, if $\tau_1[i] = -$, then $\tau_2[i] \neq -$ and vice versa. There are many features of a sequence alignment that effect its quality, such as the number of *matches* (indices i where $\tau_1[i] = \tau_2[i]$), *mismatches* (indices i where $\tau_1[i] \neq \tau_2[i]$), *indels* (indices i where $\tau_1[i] = -$ or $\tau_2[i] = -$), and *gaps* (ranges $[i...j]$ where $\tau[\ell] = -$ for all $\ell \in [i,j]$ and $\tau[i-1] \neq -$ or $i = 0$ and $\tau[j+1] \neq -$ or $j = n-1$ for $\tau = \tau_1$ or τ_2). We denote these features by functions $\ell_1,\ldots,\ell_d$, where each maps pairs of sequences (S_1,S_2) and alignments L to real values $\ell_j(S_1,S_2,L) \in \mathbb{R}$.

The affine-gap scoring model [Gotoh, 1982] for aligning two input sequences S_1 and S_2 computes the alignment that maximizes the objective function

$$\alpha_1 \cdot \ell_1(S_1,S_2,L) + \cdots + \alpha_d \cdot \ell_d(S_1,S_2,L), \tag{8}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^d$ is a parameter vector and the set ℓ are the numbers of matches, mismatches, indels and gaps as defined earlier. We use the notation $\mathcal{L}_{\boldsymbol{\alpha}}(S_1, S_2)$ to denote the set of alignments maximizing Equation (8). For each parameter vector $\boldsymbol{\alpha}$, we can run a dynamic programming algorithm $A_{\boldsymbol{\alpha}}$ which returns an alignment $A_{\boldsymbol{\alpha}}(S_1, S_2)$ in $\mathcal{L}_{\boldsymbol{\alpha}}(S_1, S_2)$. As we vary the weights, this gives rise to a family of algorithms. Since there is no consensus about what the best weights are, our goal is to automatically learn the best weights for a specific application domain. We assume that the domain expert has a utility function that characterizes an alignment's quality, denoted $u(S_1, S_2, L) \in [0, 1]$. We are agnostic to the specific definition of u. As a concrete example, $u(S_1, S_2, L)$ might measure the distance between L and a "ground truth" alignment of S_1 and S_2, also known as the *developer's accuracy* [Sauder et al., 2000]. In this case, the learning algorithm would require access to the ground truth alignment for every problem instance (S_1, S_2) in the training set. Ground truth is difficult to measure, so these reference alignments are never available for all sequence pairs (otherwise, we need not compute alignments).

In order to avoid tie-breaking complications, we assume that if any two parameter vectors lead to the same set of co-optimal solutions to Equation (8), the algorithm outputs the same alignment (such as the lexicographically first alignment). Formally, we say that the algorithm family $\left\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ consists of *co-optimal-constant algorithms*, defined as follows:

Definition 4.1 (Co-optimal-constant sequence alignment algorithms)**.** For each parameter vector $\boldsymbol{\alpha} \in \mathbb{R}^d$, let $A_{\boldsymbol{\alpha}}$ be an algorithm that takes as input a sequence pair $S_1, S_2 \in \Sigma^n$ and returns an alignment from the set $\mathcal{L}_{\boldsymbol{\alpha}}(S_1, S_2)$. We say that the set $\left\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ consists of *co-optimal-constant algorithms* if for any pair $\boldsymbol{\alpha}, \boldsymbol{\alpha}' \in \mathbb{R}^d$ of parameter vectors and any sequence pair $S_1, S_2 \in \Sigma^n$, $\mathcal{L}_{\boldsymbol{\alpha}}(S_1, S_2) = \mathcal{L}_{\boldsymbol{\alpha}'}(S_1, S_2)$ implies that $A_{\boldsymbol{\alpha}}(S_1, S_2) = A_{\boldsymbol{\alpha}'}(S_1, S_2)$.

In the following theorem, we prove that the utility function u, when applied to the output of the algorithm $A_{\boldsymbol{\alpha}}$, has a piecewise-structured dual function. Therefore, we can apply our main theorem to derive sample complexity guarantees.

Lemma 4.1. *Let $\left\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ be a set of co-optimal-constant algorithms and let u be a utility function mapping tuples (S_1, S_2, L) of sequence pairs and alignments to the interval $[0, 1]$. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \left\{u_{\boldsymbol{\alpha}} : (S_1, S_2) \mapsto u\left(S_1, S_2, A_{\boldsymbol{\alpha}}(S_1, S_2)\right) \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ mapping sequence pairs $S_1, S_2 \in \Sigma^n$ to $[0, 1]$. The dual class $\mathcal{U}^*$ is $\left(\mathcal{W}, \mathcal{G}, 4^n n^{4n+2}\right)$-piecewise decomposable, where $\mathcal{G} = \left\{g_{\boldsymbol{a}} : \mathcal{U} \to \{0, 1\} \mid \boldsymbol{a} \in \mathbb{R}^d\right\}$ consists of halfspace indicator functions $g_{\boldsymbol{a}} : u_{\boldsymbol{\alpha}} \mapsto \mathbb{I}_{\{\boldsymbol{a} \cdot \boldsymbol{\alpha} < 0\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_{\boldsymbol{\alpha}} \mapsto c$.*

Proof. Fix a sequence pair S_1 and S_2 and consider the function $u^*_{S_1,S_2} : \mathcal{U} \to \mathbb{R}$ from the dual class $\mathcal{U}^*$, where $u^*_{S_1,S_2}(u_{\boldsymbol{\alpha}}) = u_{\boldsymbol{\alpha}}(S_1, S_2)$. Consider the set of alignments $\mathcal{L}_{S_1,S_2} = \{A_{\boldsymbol{\alpha}}(S_1, S_2) \mid \boldsymbol{\alpha} \in \mathbb{R}^d\}$. By Lemma 4.2, we know that there are at most $2^n n^{2n+1}$ sets of co-optimal solutions as we range $\boldsymbol{\alpha}$ over $\mathbb{R}^d$. In other words, $\left|\left\{\mathcal{L}_{\boldsymbol{\alpha}}(S_1, S_2) \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}\right| \leq 2^n n^{2n+1}$. Since $\left\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ consists of co-optimal-constant algorithms, we know that $|\mathcal{L}_{S_1,S_2}| \leq 2^n n^{2n+1}$ as well. Consider an arbitrary alignment $L \in \mathcal{L}_{S_1,S_2}$. We know that L will be the alignment returned by the algorithm $A_{\boldsymbol{\alpha}}$ if and only if

$$\alpha_1 \cdot \ell_1(S_1, S_2, L) + \cdots + \alpha_d \cdot \ell_d(S_1, S_2, L) > \alpha_1 \cdot \ell_1\left(S_1, S_2, L'\right) + \cdots + \alpha_d \cdot \ell_d\left(S_1, S_2, L'\right) \tag{9}$$

for all $L' \in \mathcal{L}_{S_1,S_2} \setminus \{L\}$. Therefore, there is a set $\mathcal{H}$ of at most $\binom{2^n n^{2n+1}}{2} \leq 4^n n^{4n+2}$ hyperplanes such that across all parameter vectors $\boldsymbol{\alpha}$ in a single connected component of $\mathbb{R}^d \setminus \mathcal{H}$, the output of the algorithm parameterized by $\boldsymbol{\alpha}$, $A_{\boldsymbol{\alpha}}(S_1, S_2)$, is invariant. This means that for single connected component R of $\mathbb{R}^d \setminus \mathcal{H}$, there exists a real value c_R such that $u_{\boldsymbol{\alpha}}(S_1, S_2) = c_R$ for all $\boldsymbol{\alpha} \in R$. By definition of the dual, this means that $u^*_{S_1,S_2}(u_{\boldsymbol{\alpha}}) = c_R$ as well.

Recall that $\mathcal{G} = \left\{g_{\boldsymbol{a}} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^d\right\}$ consists of halfspace indicator functions $g_{\boldsymbol{a}} : u_{\boldsymbol{\alpha}} \mapsto \mathbb{I}_{\{\boldsymbol{a}\cdot\boldsymbol{\alpha}<0\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_{\boldsymbol{\alpha}} \mapsto c$. For each pair $L, L' \in \mathcal{L}_{S_1,S_2}$, let $g^{(L,L')} \in \mathcal{G}$ correspond to the halfspace represented in Equation (9). Order these $k := \binom{|\mathcal{L}_{S_1,S_2}|}{2}$ functions arbitrarily as $g^{(1)}, \ldots, g^{(k)}$. Every connected component R of $\mathbb{R}^d \setminus \mathcal{H}$ corresponds to a sign pattern of the k hyperplanes. For a given region R, let $\boldsymbol{b}_R \in \{0,1\}^k$ be the corresponding sign pattern. Define $f_{\boldsymbol{b}_R} = w_{c_R}$, and for every vector $\boldsymbol{b}$ not corresponding to a sign pattern of the k hyperplanes, let $f_{\boldsymbol{b}} = w_0$. In this way, for every $\boldsymbol{\alpha} \in \mathbb{R}^d$,

$$u^*_{S_1,S_2}(u_{\boldsymbol{\alpha}}) = \sum_{\boldsymbol{b}\in\{0,1\}^k} \mathbb{I}_{\left\{g^{(i)}(u_{\boldsymbol{\alpha}})=b[i],\forall i\in[k]\right\}} f_{\boldsymbol{b}}(u_{\boldsymbol{\alpha}}),$$

as desired. □

Lemma 4.2. *Fix a pair of sequences $S_1, S_2 \in \Sigma^n$. There are at most $2^n n^{2n+1}$ alignments of S_1 and S_2.*

Proof. For any alignment (τ_1, τ_2), we know that $|\tau_1| = |\tau_2|$ and for all $i \in [|\tau_1|]$, if $\tau_1[i] = -$, then $\tau_2[i] \neq -$ and vice versa. This means that τ_1 and τ_2 have the same number of gaps. To prove the upper bound, we count the number of alignments (τ_1, τ_2) where τ_1 and τ_2 each have exactly i gaps. There are $\binom{n+i}{i}$ choices for the sequence τ_1. Given a sequence τ_1, we can only pair a gap in τ_2 with a non-gap in τ_1. Since there are i gaps in τ_2 and n non-gaps in τ_1, there are $\binom{n}{i}$ choices for the sequence τ_2 once τ_1 is fixed. This means that there are $\binom{n+i}{i}\binom{n}{i} \leq 2^n n^{2n}$ alignments (τ_1, τ_2) where τ_1 and τ_2 each have exactly i gaps. Summing over $i \in [n]$, the total number of alignments is at most $2^n n^{2n+1}$. □

We now prove that the function classes $\mathcal{F}$ and $\mathcal{W}$ as defined in Lemma 4.1 have low pseudo- and VC-dimension.

Lemma 4.3. *Let $\mathcal{G} = \left\{g_{\boldsymbol{a}} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^d\right\}$ be the class of halfspace indicator functions $g_{\boldsymbol{a}} : u_{\boldsymbol{\alpha}} \mapsto \mathbb{I}_{\{\boldsymbol{a}\cdot\boldsymbol{\alpha}<0\}}$ and let $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ be the class of constant functions $w_c : u_{\boldsymbol{\alpha}} \mapsto c$. Then $\mathrm{VCdim}(\mathcal{G}^*) = d + 1$ and $\mathrm{Pdim}(\mathcal{W}^*) = 0$.*

Proof. First, consider an arbitrary function $g^*_{u_{\boldsymbol{\alpha}}} \in \mathcal{G}^*$. We know that for any $g_{\boldsymbol{a}} \in \mathcal{G}$, $g^*_{u_{\boldsymbol{\alpha}}}(f_{\boldsymbol{a}}) = \mathbb{I}_{\{\boldsymbol{a}\cdot\boldsymbol{\alpha}<0\}}$. Therefore, $\mathcal{G}^*$ is equivalent to the class of d-dimensional threshold functions, which has a VC dimension of $d + 1$. Next, consider an arbitrary function $w^*_{u_{\boldsymbol{\alpha}}} \in \mathcal{W}^*$. We know that for any $w_c \in \mathcal{W}$, $w^*_{u_{\boldsymbol{\alpha}}}(w_c) = c$. Therefore, $\mathcal{W}^*$ consists of a single function, so its pseudo-dimension is 0. □

Our main theorem together with Lemmas 4.1 and 4.3 imply the following pseudo-dimension bound.

Corollary 4.4. *Let $\left\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ be a set of co-optimal-constant algorithms and let u be a utility function mapping tuples (S_1, S_2, L) to the interval $[0,1]$. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \left\{u_{\boldsymbol{\alpha}} : (S_1, S_2) \mapsto u(S_1, S_2, A_{\boldsymbol{\alpha}}(S_1, S_2)) \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ mapping sequence pairs $S_1, S_2 \in \Sigma^n$ to $[0,1]$. Then $\mathrm{Pdim}(\mathcal{U}) \leq 4(d+1)\ln\left(e4^{n+1}n^{4n+2}(d+1)\right)$.*

Corollary 4.4 implies that for any $\epsilon > 0$, $\frac{8}{\epsilon^2}\left(\mathrm{Pdim}(\mathcal{U})\ln\frac{8}{\epsilon^2} + \frac{1}{4}\ln\frac{1}{\delta}\right)$ samples are sufficient to ensure uniform convergence. The proof of Lemma 4.1 follows a general recipe in every application throughout this paper. We summarize the recipe in the following remark in the hopes that practitioners can apply it to their own tunable algorithms.

Remark 4.1. In general, given a class of algorithms $\{A_{\boldsymbol{\rho}} \mid \boldsymbol{\rho} \in \mathbb{R}^d\}$ and corresponding utility functions $\mathcal{U} = \{u_{\boldsymbol{\rho}} : \Pi \to \mathbb{R} \mid \boldsymbol{\rho} \in \mathbb{R}^d\}$, there is a simple recipe we can typically follow to determine how the dual class $\mathcal{U}^*$ is piecewise decomposable, and thus apply our main theorem, Theorem 3.1. In particular, this recipe helps us characterize two function classes $\mathcal{F}$ and $\mathcal{G}$ and an integer k such that $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable. We describe this recipe below.

1. Fix a problem instance $x \in \Pi$. For example, in the case of sequence alignment, the problem instance x corresponds to the sequence pair (S_1, S_2) we fixed at the beginning of the proof.

2. Consider all the possible solutions the algorithm A_ρ might produce given x as input as we range over parameters $\boldsymbol{\rho} \in \mathbb{R}^d$, and call this set Ψ_x. In other words, $\Psi_x \supseteq \{A_{\boldsymbol{\rho}}(x) \mid \boldsymbol{\rho} \in \mathbb{R}^d\}$. In the proof of Lemma 4.1, the set Ψ_x corresponds to the set of alignments $\mathcal{L}_{S_1,S_2}$. We assume that there is a cap κ on the size of Ψ_x — that is, $\max_{x' \in \Pi} |\Psi_{x'}| \leq \kappa$. In the case of sequence alignment, $\kappa \leq 4^n n^{4n+2}$.

3. For any pair of possible solutions ψ and ψ' from the set Ψ_x, identify the set of parameters $\boldsymbol{\rho}$ where the algorithm $A_{\boldsymbol{\rho}}$ would choose ψ over ψ' given x as input. Let $f_{\psi,\psi',x}$ be the indicator function corresponding to this set of parameters:

$$g_{\psi,\psi',x}(u_{\boldsymbol{\rho}}) = \begin{cases} 1 & \text{if } A_{\boldsymbol{\rho}} \text{ would choose } \psi \text{ over } \psi' \text{ given } x \text{ as input} \\ 0 & \text{otherwise} \end{cases}.$$

   What form do these functions have? For example, are they hyperplanes? Are they more complex polynomial hypersurfaces? Let $\mathcal{G}$ be the corresponding class of functions: $\mathcal{G} \supseteq \{g_{\psi,\psi',x} \mid x \in \Pi, \psi, \psi' \in \Psi_x\}$. In the case of sequence alignment, the class $\mathcal{G}$ consists of halfspace indicator functions.

4. Consider an arbitrary region $R \subseteq \mathbb{R}^d$ of the parameter space where the parameterized algorithm's output is invariant. In other words, $A_{\boldsymbol{\rho}}(x)$ is fixed across all $\boldsymbol{\rho} \in R$. How does the utility $u_{\boldsymbol{\rho}}(x)$ behave as a function of $\boldsymbol{\rho} \in R$? In other words, what is the form of the dual utility function $u_x^*(u_{\boldsymbol{\rho}})$ when $\boldsymbol{\rho}$ is restricted to R? Is it constant or linear, for example, or is it some other type of function? Let $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{U}}$ be the corresponding class of functions. In the case of sequence alignment, the utility function is a constant function of the parameters (α, β, γ).

5. Finally, we conclude that the dual class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, \kappa^2)$-piecewise decomposable[4].

Tighter guarantees for a structured algorithm subclass: the affine-gap model. A line of prior work [Gusfield et al., 1994, Fernández-Baca et al., 2004, Pachter and Sturmfels, 2004b,a] analyzed the specific instantiation of the objective function (8) where $d = 3$. The goal is to find the alignment L maximizing the objective function

$$\textsc{mt}(S_1, S_2, L) - \alpha \cdot \textsc{ms}(S_1, S_2, L) - \beta \cdot \textsc{id}(S_1, S_2, L) - \gamma \cdot \textsc{gp}(S_1, S_2, L),$$

[4]To see why, consider a function $u_x^* \in \mathcal{U}^*$ and the $k = \binom{|\Psi_x|}{2} \leq \kappa^2$ binary functions $g_{\psi,\psi',x} \in \mathcal{G}$ for all $\psi, \psi' \in \Psi_x$. Order them arbitrarily as $g^{(1)}, \ldots, g^{(k)}$. Consider any region $R \subseteq \mathbb{R}^d$ where for all $\psi, \psi' \in \Psi_x$, $g_{\psi,\psi',x}(u_{\boldsymbol{\rho}})$ is invariant across all $\boldsymbol{\rho} \in R$ and let $\boldsymbol{b} = \left(g^{(1)}(u_{\boldsymbol{\rho}}), \ldots, g^{(k)}(u_{\boldsymbol{\rho}})\right)$ for an arbitrary $\boldsymbol{\rho} \in \mathbb{R}^d$. In this region, the output $A_{\boldsymbol{\rho}}(x)$ is fixed. Thus, there exists a function $f_{\boldsymbol{b}} \in \mathcal{F}$ such that $u_x^*(u_{\boldsymbol{\rho}}) = f_{\boldsymbol{b}}(u_{\boldsymbol{\rho}})$ for all $\boldsymbol{\rho} \in R$, so we conclude that the dual class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, \kappa^2)$-piecewise decomposable.

where $\textsc{mt}(S_1, S_2, L)$ is the number of columns in the alignment that have the same character (*matches*), $\textsc{ms}(S_1, S_2, L)$ is the number of columns that do not have the same character (*mismatches*), $\textsc{id}(S_1, S_2, L)$ is the total number of gap characters (*indels*, short for insertion/deletion), and $\textsc{gp}(S_1, S_2, L)$ is the number of groups of consecutive gap characters in any one row of the grid (*gaps*). This is known as the *the affine-gap scoring model*. Note that while there are four tunable parameters in this definition, without loss of flexibility we can set one to 1 (say the weight on the number of matches) and find optimal parameters only on the other three. We exploit specific structure exhibited by this algorithm family to obtain an exponential improvement in the sample complexity. This useful structure guarantees that for some constant $c_0 > 0$ and any pair of sequences (S_1, S_2), there are only $c_0 n^{3/2}$ different alignments the algorithm family $\left\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^4\right\}$ might produce as we range over parameter vectors [Gusfield et al., 1994, Fernández-Baca et al., 2004, Pachter and Sturmfels, 2004a]. This bound is exponentially smaller than our generic bound of $4^n n^{4n+2}$ from Lemma 4.2. We thereby tighten our bound in Step 2 of the generic recipe, implying a stronger sample complexity bound.

Lemma 4.5. *Let $\{A_{\alpha,\beta,\gamma} \mid \alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}\}$ be a set of co-optimal-constant algorithms and let u be a utility function mapping tuples (S_1, S_2, L) of sequence pairs and alignments to the interval $[0, 1]$. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_{\alpha,\beta,\gamma} : (S_1, S_2) \mapsto u\left(S_1, S_2, A_{\alpha,\beta,\gamma}\left(S_1, S_2\right)\right) \mid \alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}\}$ mapping sequence pairs $S_1, S_2 \in \Sigma^n$ to $[0, 1]$. For some constant $c_0 > 0$, the dual class $\mathcal{U}^*$ is $\left(\mathcal{W}, \mathcal{G}, c_0^2 n^3\right)$-piecewise decomposable, where $\mathcal{G} = \{g_{a_1,a_2,a_3,a_4} : \mathcal{U} \to \{0, 1\} \mid a_1, a_2, a_3, a_4 \in \mathbb{R}\}$ consists of halfspace indicator functions $g_{a_1,a_2,a_3,a_4} : u_{\alpha,\beta,\gamma} \mapsto \mathbb{I}_{\{a_1\alpha + a_2\beta + a_3\gamma < a_4\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_{\alpha,\beta,\gamma} \mapsto c$.*

Proof. Fix a sequence pair S_1 and S_2 and consider the function $u^*_{S_1,S_2} : \mathcal{U} \to \mathbb{R}$ from the dual class $\mathcal{U}^*$, where $u^*_{S_1,S_2}(u_{\alpha,\beta,\gamma}) = u_{\alpha,\beta,\gamma}(S_1, S_2)$. Consider the set of alignments $\mathcal{L}_{S_1,S_2} = \{A_{\alpha,\beta,\gamma}(S_1, S_2) \mid \alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}\}$. From work by Pachter and Sturmfels [2004a], we know that for some constant $c_0 > 0$ there are at most $c_0 n^{3/2}$ sets of co-optimal solutions as we range α, β, and γ over $\mathbb{R}^3_{\geq 0}$. In other words,

$$\left|\{\mathcal{L}_{\alpha,\beta,\gamma}\left(S_1, S_2\right) \mid \alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}\}\right| \leq c_0 n^{3/2}.$$

Since $\{A_{\alpha,\beta,\gamma} \mid \alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}\}$ consists of co-optimal-constant algorithms, we know that $|\mathcal{L}_{S_1,S_2}| \leq c_0 n^{3/2}$ as well. Consider an arbitrary alignment $L \in \mathcal{L}_{S_1,S_2}$. We know that L will be the alignment returned by the algorithm $A_{\alpha,\beta,\gamma}$ if and only if

$$\begin{aligned} & \textsc{mt}(S_1, S_2, L) - \alpha \cdot \textsc{ms}(S_1, S_2, L) - \beta \cdot \textsc{id}(S_1, S_2, L) - \gamma \cdot \textsc{gp}(S_1, S_2, L) \\ \geq\ & \textsc{mt}(S_1, S_2, L') - \alpha \cdot \textsc{ms}(S_1, S_2, L') - \beta \cdot \textsc{id}(S_1, S_2, L') - \gamma \cdot \textsc{gp}(S_1, S_2, L') \end{aligned} \tag{10}$$

for all $L' \in \mathcal{L}_{S_1,S_2} \setminus \{L\}$. Therefore, there is a set $\mathcal{H}$ of

$$\binom{c_0 n^{3/2}}{2} \leq c_0^2 n^3$$

hyperplanes such that across all (α, β, γ) in a single connected component of $\mathbb{R}^3_{\geq 0} \setminus \mathcal{H}$, the output of the algorithm parameterized by α, β, γ, $A_{\alpha,\beta,\gamma}(S_1, S_2)$, is invariant. This means that for single connected component R of $\mathbb{R}^3 \setminus \mathcal{H}$, there exists a real value c_R such that $u_{\alpha,\beta,\gamma}(S_1, S_2) = c_R$ for all $(\alpha, \beta, \gamma) \in R$. By definition of the dual, this means that $u^*_{S_1,S_2}(u_{\alpha,\beta,\gamma}) = c_R$ as well.

Recall that $\mathcal{G} = \{g_{a_1,a_2,a_3,a_4} : \mathcal{U} \to \{0, 1\} \mid a_1, a_2, a_3, a_4 \in \mathbb{R}\}$ consists of halfspace indicator functions $g_{a_1,a_2,a_3,a_4} : u_{\alpha,\beta,\gamma} \mapsto \mathbb{I}_{\{a_1\alpha + a_2\beta + a_3\gamma < a_4\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_{\alpha,\beta,\gamma} \mapsto c$. For each pair $L, L' \in \mathcal{L}_{S_1,S_2}$, let $g^{(L,L')} \in \mathcal{G}$ correspond to the halfspace represented in Equation (10). Order these $k := \binom{|\mathcal{L}_{S_1,S_2}|}{2}$ functions arbitrarily as

$g^{(1)}, \ldots, g^{(k)}$. Every connected component R of $\mathbb{R}^3_{\geq 0} \setminus \mathcal{H}$ corresponds to a sign pattern of the k hyperplanes. For a given region R, let $\boldsymbol{b}_R \in \{0,1\}^k$ be the corresponding sign pattern. Define $f_{\boldsymbol{b}_R} = w_{c_R}$, and for every vector $\boldsymbol{b}$ not corresponding to a sign pattern of the k hyperplanes, let $f_{\boldsymbol{b}} = w_0$. In this way, for every $\alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}$,

$$u^*_{S_1,S_2}(u_{\alpha,\beta,\gamma}) = \sum_{\boldsymbol{b} \in \{0,1\}^k} \mathbb{I}_{\left\{g^{(i)}(u_{\alpha,\beta,\gamma}) = b[i], \forall i \in [k]\right\}} f_{\boldsymbol{b}}(u_{\alpha,\beta,\gamma}),$$

as desired. □

Our main theorem together with Lemmas 4.5 and 4.3 imply the following pseudo-dimension bound.

Corollary 4.6. *Let $\{A_{\alpha,\beta,\gamma} \mid \alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}\}$ be a set of co-optimal-constant algorithms and let u be a utility function mapping tuples (S_1, S_2, L) to the interval $[0,1]$. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_{\alpha,\beta,\gamma} : (S_1, S_2) \mapsto u\left(S_1, S_2, A_{\alpha,\beta,\gamma}\left(S_1, S_2\right)\right) \mid \alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}\}$ mapping sequence pairs $S_1, S_2 \in \Sigma^n$ to $[0,1]$. For some constant $c_0 > 0$, $\operatorname{Pdim}(\mathcal{U}) \leq 20 \ln\left(20 c_0^2 e n^3\right)$.*

Corollary 4.6 implies that for any $\epsilon > 0$, $\frac{8}{\epsilon^2}\left(20 \ln\left(20 e c_0^2 n^3\right) \ln \frac{8}{\epsilon^2} + \frac{1}{4} \ln \frac{1}{\delta}\right)$ samples are sufficient to ensure uniform convergence. By taking advantage of structure uncovered by prior research [Pachter and Sturmfels, 2004a], we thus obtain an exponentially-better dependence on the sequence length n than the sample complexity guarantee from Corollary 4.4 for one of the most common pairwise sequence alignment formulations.

Tighter guarantees for a structured algorithm subclass: sequence alignment using hidden Markov models. While we focused on the affine gap model in the previous section, which was inspired by the results by Gusfield et al. [1994], the result by Pachter and Sturmfels [2004a] helps to provide uniform convergence guarantees for any alignment scoring function that can be modeled as a *hidden Markov model (HMM)*. A bound on the number of parameter choices that emit distinct sets of co-optimal alignments in that work is found by taking an algebraic view of the alignment HMM with d tunable parameters. In fact, the bounds provided can be used to provide guarantees for many types of HMMs.

Lemma 4.7. *Let $\{A_{\alpha_1,\ldots,\alpha_d} \mid \alpha_1, \ldots, \alpha_d \in \mathbb{R}\}$ be a set of co-optimal-constant algorithms and let u be a utility function mapping tuples (S_1, S_2, L) of sequence pairs and alignments to the interval $[0,1]$. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_{\alpha_1,\ldots,\alpha_d} : (S_1, S_2) \mapsto u\left(S_1, S_2, A_{\alpha_1,\ldots,\alpha_d}\left(S_1, S_2\right)\right) \mid \alpha_1, \ldots, \alpha_d \in \mathbb{R}\}$ mapping sequence pairs $S_1, S_2 \in \Sigma^n$ to $[0,1]$. For some constant $c_1 > 0$, the dual class $\mathcal{U}^*$ is $\left(\mathcal{W}, \mathcal{G}, c_1^2 n^{2d(d-1)/(d+1)}\right)$-piecewise decomposable, where $\mathcal{G} = \{g_{a_1,\ldots,a_{d+1}} : \mathcal{U} \to \{0,1\} \mid a_1, \ldots, a_{d+1} \in \mathbb{R}\}$ consists of halfspace indicator functions $g_{a_1,\ldots,a_{d+1}} : u_{\alpha_1,\ldots,\alpha_d} \mapsto \mathbb{I}_{\{a_1\alpha_1 + \ldots + a_d\alpha_d < a_{d+1}\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consist of constant functions $w_c : u_{\alpha_1,\ldots,\alpha_d} \mapsto c$.*

Proof. Fix a sequence pair S_1 and S_2 and consider the function $u^*_{S_1,S_2} : \mathcal{U} \to \mathbb{R}$ from the dual class $\mathcal{U}^*$, where $u^*_{S_1,S_2}(u_{\alpha_1,\ldots,\alpha_d}) = u_{\alpha_1,\ldots,\alpha_d}(S_1, S_2)$. Consider the set of alignments $\mathcal{L}_{S_1,S_2} = \{A_{\alpha_1,\ldots,\alpha_d}(S_1, S_2) \mid \alpha_1, \ldots, \alpha_d \in \mathbb{R}\}$. There are at most $c_1 n^{d(d-1)/(d+1)}$ sets of co-optimal solutions as we range $\alpha_1, \ldots, \alpha_d$ over $\mathbb{R}^d$ [Pachter and Sturmfels, 2004a]. The remainder of the proof is analogous to that for Lemma 4.5. □

Finally the results of Lemma 4.7 imply the following pseudo-dimension bound.

Corollary 4.8. *Let* $\{A_{\alpha_1,\ldots,\alpha_d} \mid \alpha_1,\ldots,\alpha_d \in \mathbb{R}\}$ *be a set of co-optimal-constant algorithms and let* u *be a utility function mapping tuples* (S_1, S_2, L) *to the interval* $[0,1]$*. Let* $\mathcal{U}$ *be the set of functions* $\mathcal{U} = \{u_{\alpha_1,\ldots,\alpha_d} : (S_1,S_2) \mapsto u\left(S_1, S_2, A_{\alpha_1,\ldots,\alpha_d}(S_1,S_2)\right) \mid \alpha_1,\ldots,\alpha_d \in \mathbb{R}\}$ *mapping sequence pairs* $S_1, S_2 \in \Sigma^n$ *to* $[0,1]$*. For some* $c_1 > 0$*,* $\mathrm{Pdim}(\mathcal{U}) \leq 4\,(d+1)\ln\left(4ec_1^2 n^{2d(d-1)/(d+1)}\,(d+1)\right)$*.*

Corollary 4.8 implies that for any $\epsilon > 0$, $\frac{8}{\epsilon^2}\left(\mathrm{Pdim}(\mathcal{U})\ln\frac{8}{\epsilon^2} + \frac{1}{4}\ln\frac{1}{\delta}\right)$ samples are sufficient to ensure uniform convergence even with more complicated optimization functions than considered in by the initial work on inverse parametric alignment.

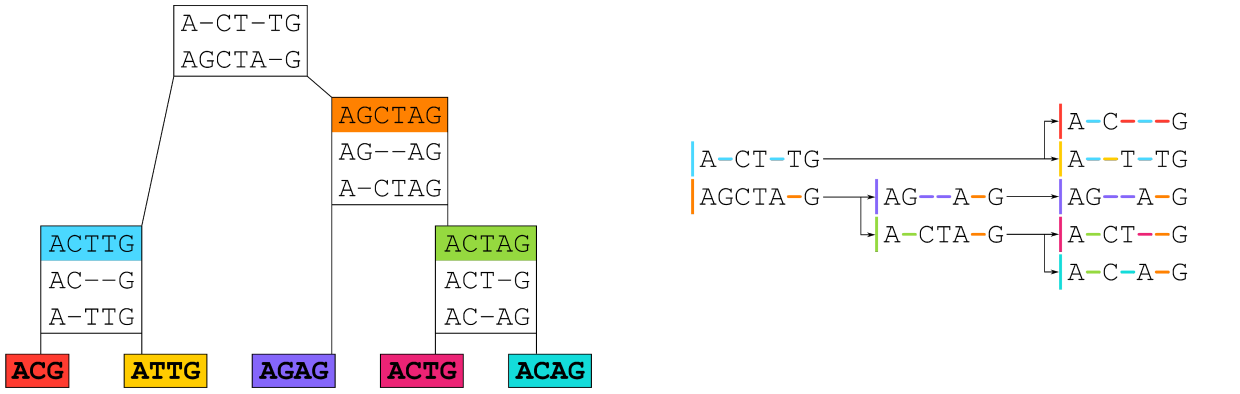
4.1.2 Progressive multiple sequence alignment

The multiple sequence alignment problem is a generalization of the pairwise alignment problem introduced in Section 4.4.1. Let Σ be an abstract alphabet and let $S_1,\ldots,S_\kappa \in \Sigma^n$ be a collection of sequences in Σ of length n. A *multiple sequence alignment* is a collection of sequences $\tau_1,\ldots,\tau_\kappa \in (\Sigma \cup \{-\})^*$ such that the following hold:

1. The aligned sequences are the same length: $\forall i,j$ we have $|\tau_i| = |\tau_j|$.
2. Removing the gap characters from τ_i gives S_i: $\forall i$ we have $\texttt{del}(\tau_i) = S_i$.
3. For every position i in the final alignment, at least one of the aligned sequences has a non-gap character: $\forall i \in [|\tau_1|]$, $\exists j \in [\kappa]$ such that $\tau_j[i] \neq -$.

The extension from pairwise to multiple sequence alignment, however, is computationally challenging: all common formulations of the problem are NP-complete [Wang and Jiang, 1994, Kececioglu and Starrett, 2004]. Therefore, every algorithm that solves the multiple sequence alignment problem likely takes an inordinate amount of time to find a solution. As a result, scientists have developed heuristics to find good, but possibly sub-optimal, alignments. The most common heuristic approach is called *progressive multiple sequence alignment*. It leverages efficient pairwise alignment algorithms to heuristically align multiple sequences [Feng and Doolittle, 1987]. Progressive alignment algorithms have two phases. First, they construct a binary *guide tree* that decomposes the the original alignment problem into a hierarchy of subproblems, each of which can be approximately solved using pairwise alignment. The leaves of the guide tree correspond to the input sequences $S_1,\ldots,S_\kappa$. Each internal node represents the subproblem of aligning the sequences at the leaves of its subtree. We assume this guide tree is provided to the algorithm as input.

At a high level, the second phase recursively constructs an alignment and a *consensus sequence* for each node of the guide tree. That is, for each node v in the tree, we construct an alignment L_v of the leaves in the subtree rooted at v, as well as a consensus sequence $\rho_v \in \Sigma^*$. Since the leaves correspond to single input sequences, they have a trivial alignment and the consensus sequence is just the corresponding input sequence. For an internal node v with children c_1 and c_2, we use a pairwise alignment algorithm to construct an alignment of the consensus strings ρ_{c_1} and ρ_{c_2}. The consensus sequences are defined so that when we combine this alignment with the alignments for the children L_{c_1} and L_{c_2}, we obtain an alignment L_v for the subproblem at node v. Finally, we define the consensus sequence of the node v to be the string $\rho_v \in \Sigma^*$ such that $\rho_v[i]$ is the most-frequent non-gap character in the i^{th} position in the alignment L_v. This is an adaptation of the "partial consensus" generalization described by Higgins and Sharp [1988]. We obtain a full multiple sequence alignment by iteratively replacing each consensus sequence by the pairwise alignment it represents, adding gap columns to the sub-alignments when necessary. Once we add a gap to a sequence, we never remove it: "once a gap, always a gap." Figure 7 illustrates an example of this algorithm in action.



(a) Guide tree.

(b) Constructing the alignment using the guide tree

Figure 7: This figure illustrates an example of the progressive sequence alignment algorithm in action. Figure 7a depicts a completed guide tree. The five input sequences are represented by the leaves. Each internal leaf, depicts an alignment of the (consensus) sequences contained in the leaf's children. Each internal leaf other than the root also contains the consensus sequence corresponding to that alignment. Figure 7b illustrates how to extract an alignment of the five input strings (as well as the consensus strings) from Figure 7a.

The family $\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\}$ of parameterized pairwise alignment algorithms introduced in Section 4.4.1 induces a parameterized family of progressive multiple sequence alignment algorithms $\{M_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\}$. In particular, the algorithm $M_{\boldsymbol{\alpha}}$ takes as input a collection of input sequences $S_1, \ldots, S_\kappa \in \Sigma^n$, a guide tree G, and outputs a multiple-sequence alignment L by applying the pairwise alignment algorithm $A_{\boldsymbol{\alpha}}$ at each node of the guide tree.

Lemma 4.9. *Let $\{M_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\}$ be the family of multiple sequence alignment algorithms derived from a family $\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\}$ of co-optimal-constant pairwise alignment algorithms. Let u be a utility function mapping tuples $(S_1, \ldots, S_\kappa, G, L)$ of sequences, a guide graph G with height at most η, and an alignment L to the interval $[0, 1]$. Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \left\{ u_{\boldsymbol{\alpha}} : (S_1, \ldots, S_\kappa, G) \mapsto u\big(S_1, \ldots, S_\kappa, G, M_{\boldsymbol{\alpha}}(S_1, \ldots, S_\kappa, G)\big) \mid \boldsymbol{\alpha} \in \mathbb{R}^d \right\}$$

mapping problem instances to utilities. The dual class $\mathcal{U}^*$ is

$$\left(\mathcal{W}, \mathcal{G}, \left(4^{n\kappa}\,(n\kappa)^{4n\kappa+2}\right)^{2d^\eta}\, 4^{d^{\eta+1}}\right)\text{-piecewise decomposable,}$$

where $\mathcal{G} = \{g_{\boldsymbol{a},\theta} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^d, \theta \in \mathbb{R}\}$ consists of halfspace indicator functions $g_{\boldsymbol{a},\theta} : u_{\boldsymbol{\rho}} \mapsto \mathbb{I}_{\{\boldsymbol{a}\cdot\boldsymbol{\rho} \le \theta\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_{\boldsymbol{\alpha}} \mapsto c$.

The doubly-exponential bound on the number of hyperplanes may seem ominous at first glance, but the pseudo-dimension grows only linearly in n and quadratically in κ in the affine-gap model ($d = 3$) when the guide tree is balanced ($\eta \le \log \kappa$).

Proof. A key step in the proof of Lemma 4.1 shows that for any pair of sequences $S_1, S_2 \in \Sigma^n$, we can find a set $\mathcal{H}$ of $4^n n^{4n+2}$ hyperplanes such that for any pair $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha'}$ belonging to the same connected component of $\mathbb{R}^d \setminus \mathcal{H}$, we have $A_{\boldsymbol{\alpha}}(S_1, S_2) = A_{\boldsymbol{\alpha'}}(S_1, S_2)$. We use this result to prove the following claim.

Claim 4.10. *For each node v in the guide tree, there is a set $\mathcal{H}_v$ of hyperplanes where for any connected component C of $\mathbb{R}^d \setminus \mathcal{H}_v$, the alignment and consensus sequence computed by $M_{\boldsymbol{\alpha}}$ is invariant across $\boldsymbol{\alpha} \in C$. Moreover, the size of $\mathcal{H}_v$ is bounded as follows:*

$$|\mathcal{H}_v| \leq \ell^{d^{\text{height}(v)}} \left(\ell 4^d\right)^{\left(d^{\text{height}(v)}-1\right)/(d-1)},$$

where $\tilde{n} := n\kappa$ and $\ell := 4^{\tilde{n}}\tilde{n}^{4\tilde{n}+2}$.

Before we prove Claim 4.10, we remark that the longest consensus sequence computed for any node v of the guide tree has length at most $\tilde{n} = n\kappa$, which is a bound on the sum of the lengths of the input sequences.

Proof of Claim 4.10. We prove this claim by induction on the guide tree G. The base case corresponds to the leaves of G. On each leaf, the alignment and consensus sequence constructed by $M_{\boldsymbol{\alpha}}$ is constant for all $\boldsymbol{\alpha} \in \mathbb{R}^d$, since there is only one string to align (i.e., the input string placed at that leaf). Therefore, the claim holds for the leaves of G. Moving to an internal node v, suppose that the inductive hypothesis holds for its children c_1 and c_2. Assume without loss of generality that $\text{height}(c_1) \geq \text{height}(c_2)$, so that $\text{height}(v) = \text{height}(c_1) + 1$. Let $\mathcal{H}_{c_1}$ and $\mathcal{H}_{c_2}$ be the sets of hyperplanes corresponding to the children c_1 and c_2. By the inductive hypothesis, these sets are each of size at most

$$s := \ell^{d^{\text{height}(c_1)}} \left(\ell 4^d\right)^{\left(d^{\text{height}(c_1)}-1\right)/(d-1)}$$

Letting $\mathcal{H} = \mathcal{H}_{c_1} \cup \mathcal{H}_{c_2}$, we are guaranteed that for every connected component of $\mathbb{R}^d \setminus \mathcal{H}$, the alignment and consensus string computed by $M_{\boldsymbol{\alpha}}$ for both children c_1 and c_2 is constant. Based on work by Buck [1943], we know that there are at most $(2s+1)^d \leq (3s)^d$ connected components of $\mathbb{R}^d \setminus \mathcal{H}$. For each region, by the same argument as in the proof of Lemma 4.1, there are an additional ℓ hyperplanes that partition the region into subregions where the outcome of the pairwise merge at node v is constant. Therefore, there is a set $\mathcal{H}_v$ of at most

$$\begin{aligned}
\ell(3s)^d + 2s &\leq \ell(4s)^d \\
&= \ell\left(4\ell^{d^{\text{height}(c_1)}} \left(\ell 4^d\right)^{\left(d^{\text{height}(c_1)}-1\right)/(d-1)}\right)^d \\
&= \ell^{d^{\text{height}(c_1)+1}} \left(\ell 4^d\right)^{\left(d^{\text{height}(c_1)+1}-d\right)/(d-1)+1} \\
&= \ell^{d^{\text{height}(c_1)+1}} \left(\ell 4^d\right)^{\left(d^{\text{height}(c_1)+1}-1\right)/(d-1)} \\
&= \ell^{d^{\text{height}(v)}} \left(\ell 4^d\right)^{\left(d^{\text{height}(v)}-1\right)/(d-1)}
\end{aligned}$$

hyperplanes where for every connected component of $\mathbb{R}^d \setminus \mathcal{H}$, the alignment and consensus string computed by $M_{\boldsymbol{\alpha}}$ at v is invariant. □

Applying Claim 4.10 to the root of the guide tree, the function $\boldsymbol{\alpha} \mapsto M_{\boldsymbol{\alpha}}(S_1, \ldots, S_\kappa, G)$ is piecewise constant with

$$\ell^{d^{\text{height}(G)}} \left(\ell 4^d\right)^{\left(d^{\text{height}(G)}-1\right)/(d-1)}$$

linear boundary functions. The lemma then follows from the following chain of inequalities:

$$\ell^{d^{\text{height}(G)}} \left(\ell 4^d\right)^{\left(d^{\text{height}(G)}-1\right)/(d-1)} \leq \ell^{d^{\text{height}(G)}} \left(\ell 4^d\right)^{d^{\text{height}(G)}}$$

$$
\begin{aligned}
&= \ell^{2d^{\text{height}(G)}} 4^{d^{\text{height}(G)+1}} \\
&= \left(4^{\tilde{n}} \tilde{n}^{4\tilde{n}+2}\right)^{2d^{\text{height}(G)}} 4^{d^{\text{height}(G)+1}} \\
&= \left(4^{n\kappa} (n\kappa)^{4n\kappa+2}\right)^{2d^{\text{height}(G)}} 4^{d^{\text{height}(G)+1}} \\
&\leq \left(4^{n\kappa} (n\kappa)^{4n\kappa+2}\right)^{2d^{\eta}} 4^{d^{\eta+1}}.
\end{aligned}
$$

□

Our main theorem together with Lemma 4.9 implies the following pseudo-dimension bound.

Corollary 4.11. *Let $\left\{M_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ be the family of progressive multiple sequence alignment algorithms derived from a family $\left\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ of co-optimal-constant pairwise alignment algorithms. Let u be a utility function mapping tuples $(S_1, \ldots, S_\kappa, G, L)$ of sequences, a guide graph G with height at most η, and an alignment L to the interval $[0, 1]$. Let $\mathcal{U}$ be the set of functions*

$$
\mathcal{U} = \left\{u_{\boldsymbol{\alpha}} : (S_1, \ldots, S_\kappa, G) \mapsto u\big(S_1, \ldots, S_\kappa, G, M_{\boldsymbol{\alpha}}(S_1, \ldots, S_\kappa, G)\big) \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}
$$

mapping problem instances to utilities. Then

$$
\text{Pdim}(\mathcal{U}) \leq 4\,(d+1) \ln\left(4e\left(4^{n\kappa}\,(n\kappa)^{4n\kappa+2}\right)^{2d^{\eta}} 4^{d^{\eta+1}}\,(d+1)\right).
$$

This pseudo-dimension bound is small in the affine-gap model when the guide tree G is balanced because $\eta \leq \log \kappa$. In that case, the pseudo-dimension grows only linearly in n and quadratically in κ.

4.1.3 RNA folding

To perform functions within the cell, some RNA form 3-dimensional structures by folding the single length of RNA, binding non-adjacent pairs of bases together physically. Formally, given a sequence $S \in \Sigma^n$ from an alphabet Σ, a *folding* is a set of a pairs (i, j) such that $0 \leq i < j \leq n$, each base is involved in only one pair, and the folding does not contain any pseudoknots (a pair of pairs $(i, j), (i', j')$ such that $i < i' < j < j'$). A commonly-used procedure for finding a folding $\phi \subset \{(i, j) \mid 0 \leq i < j < n\}$ of an input sequence $S \subseteq \Sigma^n$ computes the folding that maximizes the objective function

$$
\alpha\,|\phi| + (1-\alpha) \sum_{(i,j)\in\phi} M_{\binom{S[i],S[j]}{S[i-1],S[j+1]}}^{\mathbb{I}_{\{(i,j),(i-1,j+1)\in\phi\}}} \tag{11}
$$

where $\alpha \in [0, 1]$ is a tunable parameter and M is a fixed, arbitrary, non-negative weight matrix. The model described here is subset of that described in Nussinov and Jacobson [1980]. We only consider the energies of single and adjacent base-pairs whereas Nussinov and Jacobson [1980] consider many more possible patterns. We use the notation $\phi_\alpha(S)$ to denote the set of foldings maximizing Equation (11). For each parameter α, we can run a dynamic programming algorithm A_α which returns a folding $A_\alpha(S)$ in $\phi_\alpha(S)$. As we vary the weight, this gives rise to a family of algorithms. Since there is no consensus about what the best weight is, our goal is to automatically learn the best weight for a specific application domain. As in Section 4.4.1, we assume that the domain expert has a utility function that characterizes a folding's quality, denoted $u(S, \phi) \in [0, 1]$. We are again agnostic to the specific definition of u, but as a concrete example, $u(S, \phi)$ might measure

the fraction of pairs shared between ϕ and a "ground truth" folding ϕ^*. In this case, the learning algorithm would require access to the ground truth folding for every sequence S in the training set.

As in Section 4.4.1, we assume the algorithm family $\left\{A_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{R}^d\right\}$ consists of *co-optimal-constant algorithms* in order to avoid tie-breaking complications. Namely, we assume that if any two parameters lead to the same set of co-optimal solutions to Equation (11), the algorithm outputs the same folding (such as the lexicographically first folding).

Definition 4.2 (Co-optimal-constant folding algorithms)**.** For each parameter $\alpha \in [0,1]$, let A_α be an algorithm that takes as input a sequence S and returns a folding from the set $\phi_\alpha(S)$. We say that the set $\{A_\alpha \mid \alpha \in [0,1]\}$ consists of *co-optimal-constant algorithms* if for any pair $\alpha, \alpha' \in [0,1]$ of parameters and any matrix M, $\phi_\alpha(S) = \phi_{\alpha'}(S)$ implies that $A_\alpha(S) = A_{\alpha'}(S)$.

In the following theorem, we prove that the utility function u, when applied to the output of the algorithm A_α, has a piecewise-structured dual function. Therefore, we can apply our main theorem to derive sample complexity guarantees.

Lemma 4.12. *Let $\{A_\alpha \mid \alpha \in [0,1]\}$ be a set of co-optimal-constant algorithms and let u be a utility function mapping pairs (S,ϕ) of sequences and foldings to the interval $[0,1]$. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_\alpha : S \mapsto u(S, A_\alpha(S)) \mid \alpha \in [0,1]\}$ mapping sequences S to $[0,1]$. The dual class $\mathcal{U}^*$ is $\left(\mathcal{W}, \mathcal{G}, n^{2n}\right)$-piecewise decomposable, where $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\alpha \mapsto \mathbb{I}_{\{\alpha<a\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_\alpha \mapsto c$.*

Proof. Fix a sequence S and consider the function $u_S^* : \mathcal{U} \to \mathbb{R}$ from the dual class $\mathcal{U}^*$, where $u_S^*(u_\alpha) = u_\alpha(S)$. Consider the set of foldings $\Phi^* = \{A_\alpha(S) \mid \alpha \in [0,1]\}$. We know that for any folding ϕ, $\phi \subset [n] \times [n]$ and $|\phi| \in \{0, \ldots, n/2\}$, which means that $|\Phi^*| \leq \binom{n^2}{n/2} \leq n^n$. Consider an arbitrary folding $\phi \in \Phi^*$. We know that ϕ will be the folding returned by the algorithm $A_\alpha(S)$ if and only if

$$\begin{aligned} &\alpha\,|\phi| + (1-\alpha) \sum_{(i,j)\in\phi} M_{\binom{S[i],S[j]}{S[i-1],S[j+1]}}^{\mathbb{I}_{\{(i,j),(i-1,j+1)\in\phi\}}} \\ \geq\ &\alpha\,|\phi'| + (1-\alpha) \sum_{(i,j)\in\phi'} M_{\binom{S[i],S[j]}{S[i-1],S[j+1]}}^{\mathbb{I}_{\{(i,j),(i-1,j+1)\in\phi'\}}} \end{aligned} \tag{12}$$

for all $\phi' \in \Phi^* \setminus \{\phi\}$. Since these functions are linear in α, this means there is a set of $T \leq \binom{n^n}{2} \leq n^{2n}$ intervals $[\alpha_0, \alpha_1), [\alpha_1, \alpha_2), \ldots, [\alpha_{T-1}, \alpha_T]$ with $0 := \alpha_0 < \alpha_1 < \cdots < \alpha_{T-1} < 1 := \alpha_T$ such that for any one interval I, across all $\alpha \in I$, $A_\alpha(S)$ is invariant. This means that for any one interval $[\alpha_i, \alpha_{i+1})$, there exists a real value c_i such that $u_\alpha(S) = c_i$ for all $\alpha \in [\alpha_i, \alpha_{i+1})$. By definition of the dual, this means that $u_S^*(u_\alpha) = c_i$ as well.

Recall that $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\alpha \mapsto \mathbb{I}_{\{\alpha<a\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_\alpha \mapsto c$. Consider the functions $g^{(1)} := g_{\alpha_0}, \ldots, g^{(T+1)} := g_{\alpha_T} \in \mathcal{G}$. We claim that there exists a function $f_{\boldsymbol{b}}$ for every vector $\boldsymbol{b} \in \{0,1\}^{T+1}$ such that for every $\alpha \in [0,1]$,

$$u_S^*(u_\alpha) = \sum_{\boldsymbol{b}\in\{0,1\}^{T+1}} \mathbb{I}_{\left\{g^{(i)}(u_\alpha)=b[i],\forall i\in[T+1]\right\}} f_{\boldsymbol{b}}(u_\alpha). \tag{13}$$

To see why, suppose $\alpha \in [\alpha_i, \alpha_{i+1})$ for some $i \leq T$. Then $g_{\alpha_j}(u_\alpha) = g^{(j+1)}(u_\alpha) = \mathbb{I}_{\{\alpha\leq\alpha_j\}} = 1$ for all $j \geq i+1$ and $g_{\alpha_j}(u_\alpha) = g^{(j+1)}(u_\alpha) = \mathbb{I}_{\{\alpha\leq\alpha_j\}} = 0$ for all $j \leq i$. Let $\boldsymbol{b} \in \{0,1\}^{T+1}$ be the vector

that has only 0's in its first i coordinates and all 1's in its remaining $n-i$ coordinates. We define $f_{\boldsymbol{b}} = w_{c_i}$. For any other $\boldsymbol{b}$, we set $f_{\boldsymbol{b}} = w_0$ (note that it will never be the case that $g^{(i)}(u_\alpha) = b[i]$ for all $i \in [T+1]$). Therefore, Equation (13) holds. □

We now prove that the function classes $\mathcal{G}$ and $\mathcal{W}$ as defined in Lemma 4.12 have low pseudo- and VC-dimension.

Lemma 4.13. *Let $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ be the class of threshold functions $g_a : u_\gamma \mapsto \mathbb{I}_{\{\gamma<a\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ be the class of constant functions $w_c : u_\alpha \mapsto c$. Then $\mathrm{VCdim}(\mathcal{G}^*) = 1$ and $\mathrm{Pdim}(\mathcal{W}^*) = 0$.*

Proof. First, consider an arbitrary function $g^*_{u_\alpha} \in \mathcal{G}^*$. We know that for any $g_a \in \mathcal{G}$, $g^*_{u_\alpha}(g_a) = \mathbb{I}_{\{\alpha<a\}}$. Therefore, $\mathcal{G}^*$ is equivalent to the class of threshold functions, which has a VC dimension of 1. Next, consider an arbitrary function $w^*_{u_\alpha} \in \mathcal{W}^*$. We know that for any $w_c \in \mathcal{W}$, $w^*_{u_\alpha}(w_c) = c$. Therefore, $\mathcal{W}^*$ consists of a single function, so its pseudo-dimension is 0. □

Our main theorem together with Lemmas 4.12 and 4.13 imply the following pseudo-dimension bound.

Corollary 4.14. *Let $\{A_\alpha \mid \alpha \in [0,1]\}$ be a set of co-optimal-constant algorithms and let u be a utility function mapping pairs (S,ϕ) of sequences and foldings to the interval $[0,1]$. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_\alpha : S \mapsto u(S, A_\alpha(S)) \mid \alpha \in [0,1]\}$ mapping sequences S to $[0,1]$. Then $\mathrm{Pdim}(\mathcal{U}) \le 4\ln\left(4en^{2n}\right)$.*

Corollary 4.14 implies that for any $\epsilon > 0$, $\frac{8}{\epsilon^2}\left(4\left(\ln(4e) + 2n\ln n\right)\ln\frac{8}{\epsilon^2} + \frac{1}{4}\ln\frac{1}{\delta}\right)$ samples are sufficient to ensure uniform convergence.

4.1.4 Topologically Associated Domain (TAD) finding

Inside a cell, the linear DNA of the genome takes on a 3-dimensional shape that has been shown to be important for various cellular functions. Because of this, certain regions of the genome are closer to each other more often, and are thought to interact more. We call these regions *topologically associating domains (TADs)*. Measuring TADs directly is not currently possible. However, the Hi-C [Lieberman-Aiden et al., 2009] protocol permits the measurement of the contact frequency for all pairs of locations in the genome. Using this contact frequency, the TAD-prediction problem is to identify the regions along the genome that are frequently in contact in certain conditions and label them as TADs.

More formally, given the sequence or genome length $n \in \mathbb{N}$, let $\binom{[n]}{2}$ be the set of all ordered pairs $\binom{[n]}{2} = \{(i,j) : 1 \le i < j \le n\}$. We use the objective function from Filippova et al. [2014]. Given a weighted adjacency matrix $M \in \mathbb{R}^{n\times n}$ and a parameter $\gamma \ge 0$, the goal of TAD-finding is to compute the TAD set $T \subseteq \binom{[n]}{2}$ that maximizes the objective function

$$\sum_{(i,j)\in T} SM_\gamma(i,j) - \mu_\gamma(j-i), \tag{14}$$

where

$$SM_\gamma(i,j) = \frac{1}{(j-i)^\gamma}\sum_{i\le p<q\le j} M_{pq} \text{ and } \mu_\gamma(d) = \frac{1}{n-d}\sum_{t=0}^{n-d} SM_\gamma(t,t+d).$$

We use the notation $\mathcal{T}_\gamma(M)$ to denote the set of TAD sets maximizing Equation (14). For each parameter γ, we can run a dynamic programming algorithm A_γ which returns a labeling $A_\gamma(M)$ in

$\mathcal{T}_\gamma(M)$. As we vary the weight, this gives rise to a family of algorithms. Since there is no consensus about what the best parameter is, our goal is to automatically learn the best parameter. As before, we assume that the domain expert has a utility function that characterizes the quality of a TAD set T, denoted $u(M,T) \in [0,1]$. We are again agnostic to the specific definition of u, but as a concrete example, $u(M,T)$ might measure the fraction of TADs in T that are in the correct location given a "ground truth" TAD set T^*. In this case, the learning algorithm would require access to the ground truth TAD set—which may be hand curated—for every matrix M in the training set.

Definition 4.3 (Co-optimal-constant TAD-finding algorithms)**.** For each parameter $\gamma \in \mathbb{R}_{\geq 0}$, let A_γ be an algorithm that takes as input a matrix $M \in \mathbb{R}^{n \times n}$ and returns an alignment from the set $\mathcal{T}_\gamma(M)$. We say that the set $\{A_\gamma \mid \gamma \in \mathbb{R}_{\geq 0}\}$ consists of *co-optimal-constant algorithms* if for any pair $\gamma, \gamma' \in \mathbb{R}_{\geq 0}$ of parameters and any matrix $M \in \mathbb{R}^{n \times n}$, $\mathcal{T}_\gamma(M) = \mathcal{T}_{\gamma'}(M)$ implies that $A_\gamma(M) = A_{\gamma'}(M)$.

Lemma 4.15. *Let $\{A_\gamma \mid \gamma \in \mathbb{R}_{\geq 0}\}$ be a set of co-optimal-constant algorithms and let u be a utility function mapping pairs (M,T) of matrices and TAD sets to the interval $[0,1]$. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_\gamma : M \mapsto u(M, A_\gamma(M)) \mid \gamma \in \mathbb{R}_{\geq 0}\}$ mapping matrices $M \in \mathbb{R}^{n \times n}$ to $[0,1]$. The dual class $\mathcal{U}^*$ is $\left(\mathcal{W}, \mathcal{G}, 2n^2 4^{n^2}\right)$-piecewise decomposable, where $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\gamma \mapsto \mathbb{I}_{\{\gamma < a\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_\gamma \mapsto c$.*

Proof. We begin by rewriting Equation (14) as follows:

$$\begin{aligned}
T_\gamma &= \operatorname{argmax}_{T \subseteq \binom{[n]}{2}} \sum_{(i,j) \in T} \left(\frac{1}{(j-i)^\gamma} \left(\sum_{i \leq u < v \leq j} M_{uv} \right) - \frac{1}{n-j+i} \sum_{t=0}^{n-j+i} \frac{1}{(j-i)^\gamma} \sum_{t \leq p < q \leq t+j-i} M_{pq} \right) \\
&= \operatorname{argmax}_{T \subseteq \binom{[n]}{2}} \sum_{(i,j) \in T} \frac{1}{(j-i)^\gamma} \left(\left(\sum_{i \leq u < v \leq j} M_{uv} \right) - \frac{1}{n-j+i} \sum_{t=0}^{n-j+i} \sum_{t \leq p < q \leq t+j-i} M_{pq} \right) \\
&= \operatorname{argmax}_{T \subseteq \binom{[n]}{2}} \sum_{(i,j) \in T} \frac{c_{ij}}{(j-i)^\gamma},
\end{aligned}$$

where

$$c_{ij} = \left(\sum_{i \leq u < v \leq j} M_{uv} \right) - \frac{1}{n-j+i} \sum_{t=0}^{n-j+i} \sum_{t \leq p < q \leq t+j-i} M_{pq}.$$

Note that c_{ij} is a constant that does not depend on γ.

Fix a matrix M and consider the function $u_M^* : \mathcal{U} \to \mathbb{R}$ from the dual class $\mathcal{U}^*$, where $u_M^*(u_\gamma) = u_\gamma(M)$. Consider the set of TAD sets $\mathcal{T}^* = \{A_\gamma(M) \mid \gamma \in \mathbb{R}_{\geq 0}\}$. Since each TAD set is a subset of $\binom{[n]}{2}$, $|\mathcal{T}^*| \leq 2^{n^2}$. Moreover, since $\{A_\gamma \mid \gamma \in \mathbb{R}_{\geq 0}\}$ consists of co-optimal-constant algorithms, we know that $|\mathcal{T}^*| \leq 2^{n^2}$ as well. Consider an arbitrary TAD set $T \in \mathcal{T}^*$. We know that T will be the set returned by the algorithm A_γ if and only if

$$\sum_{(i,j) \in T} \frac{c_{ij}}{(j-i)^\gamma} > \sum_{(i',j') \in T'} \frac{c_{i'j'}}{(j'-i')^\gamma}$$

for all $T' \in \mathcal{T}^* \setminus \{T\}$. This means that as we range γ over the positive reals, the TAD set returned by algorithm $A_\gamma(M)$ will only change when

$$\sum_{(i,j) \in T} \frac{c_{ij}}{(j-i)^\gamma} - \sum_{(i',j') \in T'} \frac{c_{i'j'}}{(j'-i')^\gamma} = 0 \tag{15}$$

for some $T, T' \in \mathcal{T}^*$. By Rolle's Theorem (see Corollary 4.17), we know that Equation (15) has at most $|T| + |T'| \leq 2n^2$ solutions, this means there are at most $2n^2\binom{|\mathcal{T}^*|}{2} \leq 2n^2 4^{n^2}$ intervals partitioning $\mathbb{R}_{\geq 0}$ such that across all γ within any one interval I, the TAD set returned by algorithm $A_\gamma(M)$ is invariant. This means that there exists a real value c_I such that $u_\gamma(M) = c_I$ for all $\gamma \in I$. By definition of the dual, this means that $u^*_M(u_\gamma) = c_I$ as well.

Recall that $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of thresholds $g_a : u_\gamma \mapsto \mathbb{I}_{\{\gamma < a\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_\gamma \mapsto c$. Let $a_1, \ldots, a_t \in \mathbb{R}$ be the t boundaries of the $t - 1 = O\left(n^2 4^{n^2}\right)$ intervals partitioning $\mathbb{R}_{\geq 0}$ and let $g^{(1)}, \ldots, g^{(t)}$ be the corresponding threshold functions. For a given interval I, let $\boldsymbol{b}_I \in \{0,1\}^t$ be the corresponding sign pattern of the t threshold functions. Define $f_{\boldsymbol{b}_I} = w_{c_I}$, and for every vector $\boldsymbol{b}$ not corresponding to a sign pattern of the t thresholds, let $f_{\boldsymbol{b}} = w_0$. In this way, for every $\gamma \in \mathbb{R}_{\geq 0}$,

$$u^*_M(u_\gamma) = \sum_{\boldsymbol{b} \in \{0,1\}^t} \mathbb{I}_{\left\{g^{(i)}(u_\gamma) = b[i], \forall i \in [t]\right\}} f_{\boldsymbol{b}}(u_\gamma),$$

as desired. □

The following is a corollary of Rolle's theorem that we use in the proof of Lemma 4.15.

Lemma 4.16 (Tossavainen [2006])**.** *Let h be a polynomial-exponential sum of the form $h(x) = \sum_{i=1}^t a_i b_i^x$, where $b_i > 0$ and $a_i \in \mathbb{R}$. The number of roots of h is upper bounded by t.*

Corollary 4.17. *Let h be a polynomial-exponential sum of the form*

$$h(x) = \sum_{i=1}^t \frac{a_i}{b_i^x},$$

where $b_i > 0$ and $a_i \in \mathbb{R}$. The number of roots of h is upper bounded by t.

Proof. Note that $\sum_{i=1}^t \frac{a_i}{b_i^x} = 0$ if and only if

$$\left(\prod_{j=1}^n b_i^x\right) \sum_{i=1}^t \frac{a_i}{b_i^x} = \sum_{i=1}^n a_i \left(\prod_{j \neq i} b_i\right)^x = 0.$$

Therefore, the corollary follows from Lemma 4.16. □

The following is a corollary of Lemma 4.15.

Corollary 4.18. *Let $\{A_\gamma \mid \gamma \in \mathbb{R}_{\geq 0}\}$ be a set of co-optimal-constant algorithms and let u be a utility function mapping pairs (M, T) of matrices and TAD sets to the interval $[0,1]$. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_\gamma : M \mapsto u(M, A_\gamma(M)) \mid \gamma \in \mathbb{R}_{\geq 0}\}$ mapping matrices $M \in \mathbb{R}^{n \times n}$ to $[0,1]$. Then $\operatorname{Pdim}(\mathcal{U}) \leq 4 \ln\left(8en^2 4^{n^2}\right)$.*

4.2 Applications in economics and political science

We study a setting where there is a set $\{1, \ldots, m\}$ of m alternatives and a set of n agents. Each agent i has a value $v_i(j) \in \left(-\frac{1}{n}, \frac{1}{n}\right)$ for each alternative $j \in [m]$. We denote all m of his values as $\boldsymbol{v}_i \in \left(-\frac{1}{n}, \frac{1}{n}\right)^m$ and all n agents' values as $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \in \left(-\frac{1}{n}, \frac{1}{n}\right)^{nm}$.

A mechanism takes as input a set of bids $\boldsymbol{b}_i \in \left(-\frac{1}{n}, \frac{1}{n}\right)^m$ from each agent i. We denote all n agents' bids as $\boldsymbol{b} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \left(-\frac{1}{n}, \frac{1}{n}\right)^{nm}$. Every mechanism is defined by a *social choice*

function and a set of *payment functions*. A social choice function $f : \left(-\frac{1}{n}, \frac{1}{n}\right)^{nm} \to [m]$ uses the bids $\boldsymbol{b} \in \left(-\frac{1}{n}, \frac{1}{n}\right)^{nm}$ to choose an alternative $f(\boldsymbol{b}) \in [m]$. Moreover, for each agent $i \in [n]$, there is a payment function $p_i : \left(-\frac{1}{n}, \frac{1}{n}\right)^{nm} \to \mathbb{R}$ which maps the bids $\boldsymbol{b}$ to a value $p_i(\boldsymbol{b}) \in \mathbb{R}$ that agent i either pays or receives (if $p_i(\boldsymbol{b}) > 0$, then the agent pays that value, and if $p_i(\boldsymbol{b}) < 0$, then the agent receives that value).

We focus on mechanisms that are incentive compatible and budget balanced. A mechanism is incentive compatible if each agent is incentivized to report her values truthfully. In other words, she cannot gain by reporting strategically. We formally define incentive compatibility below.

Definition 4.4 (Incentive compatibility)**.** Fix an arbitrary agent $i \in [n]$ with values $\boldsymbol{v}_i \in \left(-\frac{1}{n}, \frac{1}{n}\right)^m$. Let $\boldsymbol{b}_{-i} \in \left(-\frac{1}{n}, \frac{1}{n}\right)^{m(n-1)}$ denote an arbitrary set of bids for all agents except agent i. Given a social choice function f, let $f\left(\boldsymbol{b}_i, \boldsymbol{b}_{-i}\right)$ be the outcome when agent i bids $\boldsymbol{b}_i \in \left(-\frac{1}{n}, \frac{1}{n}\right)^m$ and the other agents bid $\boldsymbol{b}_{-i}$. Similarly, given a payment function p_i, let $p_i\left(\boldsymbol{b}_i, \boldsymbol{b}_{-i}\right)$ be the value agent i either pays or receives when agent i bids $\boldsymbol{b}_i \in \left(-\frac{1}{n}, \frac{1}{n}\right)^m$ and the other agents bid $\boldsymbol{b}_{-i}$. We say the mechanism defined by f and p_i is *incentive compatible* if agent i cannot gain by bidding anything other than her true value. In other words, for all $\boldsymbol{b}_i \in \left(-\frac{1}{n}, \frac{1}{n}\right)^n$, $v_i\left(f\left(\boldsymbol{v}_i, \boldsymbol{b}_{-i}\right)\right) - p_i\left(\boldsymbol{v}_i, \boldsymbol{b}_{-i}\right) \geq v_i\left(f\left(\boldsymbol{b}_i, \boldsymbol{b}_{-i}\right)\right) - p_i\left(\boldsymbol{b}_i, \boldsymbol{b}_{-i}\right)$.

Moreover, a mechanism defined by payment functions $p_1, \ldots, p_n$ is budget balanced if the sum of the agents' payments equals zero: $\sum_{i=1}^n p_i(\boldsymbol{v}) = 0$.

A *neutral affine maximizer* mechanism [Roberts, 1979, Mishra and Sen, 2012, Nath and Sandholm, 2019], defined as follows, is incentive compatible and budget balanced.

Definition 4.5 (Neutral affine maximizer with sink agents)**.** A *neutral affine maximizer* (NAM) mechanism is defined by n parameters (one per agent) $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n) \in \mathbb{R}^n_{\geq 0}$ such that at least one agent is assigned a weight of zero ($\{i : \mu_i = 0\} \neq \emptyset$). The social choice function is defined as $f_{\boldsymbol{\mu}}(\boldsymbol{v}) = \operatorname{argmax}_{j \in [m]} \sum_{i=1}^n \mu_i v_i(j)$. Let $j^* = f_{\boldsymbol{\mu}}(\boldsymbol{v})$ and for each agent i, let $j_{-i} = \operatorname{argmax}_{j \in [m]} \sum_{i' \neq i} \mu_{i'} v_{i'}(j)$. The payment function is defined as

$$p_i(\boldsymbol{v}) = \begin{cases} \frac{1}{\mu_i}\left(\sum_{i' \neq i} \mu_{i'} v_{i'}\left(j^*\right) - \sum_{i' \neq i}^n \mu_{i'} v_{i'}\left(j_{-i}\right)\right) & \text{if } \mu_i \neq 0 \\ -\sum_{i' \neq i} p_{i'}(\boldsymbol{v}) & \text{if } i = \min\left\{i' : \mu_{i'} = 0\right\} \\ 0 & \text{otherwise.} \end{cases}$$

Each agent i such that $\mu_i = 0$ is known as *sink agents* because his values do not influence the outcome.

Instantiating our general theorem. Our high-level goal is to find a NAM that nearly maximizes the expected social welfare ($\sum_{i=1}^n v_i\left(j^*\right)$). The expectation is over the draw of a valuation vector $\boldsymbol{v} \sim \mathcal{D}$. Thus we define

$$u_{\boldsymbol{\mu}}(\boldsymbol{v}) = \sum_{i=1}^n v_i\left(j^*\right) \tag{16}$$

where $j^* = \operatorname{argmax}_{j \in [m]} \sum_{i=1}^n \mu_i v_i(j)$. We now prove that the set of utility functions $u_{\boldsymbol{\mu}}$ is piecewise decomposable, and thus we can apply our main theorem. This theorem allows us to prove that with high probability, if $\hat{M}$ is the NAM with maximum average social welfare over $\tilde{O}\left(n^3 \log(nm)/\epsilon^2\right)$ samples[5] and M^* is the NAM with maximum expected social welfare, then the expected social welfare of $\hat{M}$ is ϵ-close to the expected social welfare of M^*.

[5]This is assuming that the agents' values are scaled such that the social welfare of any alternative is between zero and one $\left(\forall j \in [m], \sum_{i=1}^n v_i(j) \in \left(-\frac{1}{n}, \frac{1}{n}\right)\right)$.

Theorem 4.19. *Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_{\boldsymbol{\mu}} \mid \boldsymbol{\mu} \in \mathbb{R}_{\geq 0}, \{\mu_i \mid i = 0\} \neq \emptyset\}$ where $u_{\boldsymbol{\mu}}$ is defined by Equation* (16)*. The dual class $\mathcal{U}^*$ is $\left(\mathcal{W}, \mathcal{G}, m^2\right)$-piecewise decomposable, where $\mathcal{G} = \{g_{\boldsymbol{a}} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^n\}$ consists of halfspace indicator functions $g_{\boldsymbol{a}} : u_{\boldsymbol{\mu}} \mapsto \mathbb{I}_{\{\boldsymbol{\mu}\cdot\boldsymbol{a} \leq 0\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_{\boldsymbol{\mu}} \mapsto c$.*

Proof. Fix a valuation vector $\boldsymbol{v} \in \left(-\frac{1}{n}, \frac{1}{n}\right)^{nm}$. We know that for any two alternatives $j, j' \in [m]$, the alternative j would be selected over j' so long as

$$\sum_{i=1}^{n} \mu_i v_i(j) > \sum_{i=1}^{n} \mu_i v_i\left(j'\right). \tag{17}$$

Therefore, there is a set $\mathcal{H}$ of $\binom{m}{2}$ hyperplanes such that across all parameter vectors $\boldsymbol{\mu}$ in a single connected component of $\mathbb{R}^n \setminus \mathcal{H}$, the outcome of the NAM defined by $\boldsymbol{\mu}$ is invariant. When the outcome of the NAM is invariant, the social welfare is invariant as well. This means that for a single connected component R of $\mathbb{R}^n \setminus \mathcal{H}$, there exists a real value c_R such that $u_{\boldsymbol{\mu}}(\boldsymbol{v}) = c_R$ for all $\boldsymbol{\mu} \in R$. By definition of the dual, this means that $u^*_{\boldsymbol{v}}\left(u_{\boldsymbol{\mu}}\right) = c_R$ as well.

For each pair $j, j' \in [m]$, let $g^{(j,j')} \in \mathcal{G}$ correspond to the halfspace represented in Equation (17). Order these $k := \binom{m}{2}$ functions arbitrarily as $g^{(1)}, \ldots, g^{(k)}$. Every connected component R of $\mathbb{R}^n \setminus \mathcal{H}$ corresponds to a sign pattern of the k hyperplanes. For a given region R, let $\boldsymbol{b}_R \in \{0,1\}^k$ be the corresonding sign pattern. Define $\boldsymbol{f}_{\boldsymbol{b}_R} = w_{c_R}$ and for every vector $\boldsymbol{b}$ not corresponding to a sign pattern of the k hyperplanes, let $f_{\boldsymbol{b}} = w_0$. In this way, for every $\boldsymbol{\mu} \in \mathbb{R}^n$, $u^*_{\boldsymbol{v}}\left(u_{\boldsymbol{\mu}}\right) = \sum_{\boldsymbol{b} \in \{0,1\}^k} \mathbb{I}_{\left\{g^{(i)}(u_{\boldsymbol{\mu}}) = b[i], \forall i \in [k]\right\}} f_{\boldsymbol{b}}\left(u_{\boldsymbol{\mu}}\right)$. □

Theorems 3.1 and 4.19 immediately imply the following corollary.

Corollary 4.20. *Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_{\boldsymbol{\mu}} \mid \boldsymbol{\mu} \in \mathbb{R}_{\geq 0}, \{\mu_i \mid i = 0\} \neq \emptyset\}$ where $u_{\boldsymbol{\mu}}$ is defined by Equation* (16)*. The pseudo-dimension of $\mathcal{U}$ is bounded by $4\left(n+1\right)\ln\left(4em^2\left(n+1\right)\right)$.*

Next, we prove that the pseudo-dimension of $\mathcal{U}$ is $\frac{n-1}{2}$, which means that the pseudo-dimension upper bound implied by Theorem 3.1 is tight up to log factors.

Theorem 4.21. *Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_{\boldsymbol{\mu}} \mid \boldsymbol{\mu} \in \mathbb{R}_{\geq 0}, \{\mu_i \mid i = 0\} \neq \emptyset\}$ where $u_{\boldsymbol{\mu}}$ is defined by Equation* (16)*. The pseudo-dimension of $\mathcal{U}$ is at least $\frac{n-1}{2}$.*

Proof. To prove this theorem, we present a set of $\frac{n-1}{2}$ valuation vectors over $m = 2$ alternatives $\{1, 2\}$ that are shattered by the set $\mathcal{U}$. Without loss of generality, suppose n is odd. We define a set of $(n-1)/2$ valuation vectors $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{((n-1)/2)}$ as follows:

$$v_i^{(j)}(1) = \begin{cases} \frac{1}{n} & \text{if } i \in \{2j-1, n\} \\ 0 & \text{otherwise} \end{cases} \text{ and } v_i^{(j)}(2) = \begin{cases} \frac{1}{n} & \text{if } i = 2j \\ 0 & \text{otherwise.} \end{cases}$$

Next, fix an arbitrary vector $\boldsymbol{b} \in \{0,1\}^{(n-1)/2}$. We claim that there exists a set of buyer weights $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ such that for all $\ell \in [(n-1)/2]$, if $b[\ell] = 0$, then $\operatorname{argmax}_{j \in \{1,2\}} \sum_{i=1}^{n} \mu_i v_i^{(\ell)}(j) = \frac{1}{n}$ and if $b[\ell] = 1$, then $\operatorname{argmax}_{j \in \{1,2\}} \sum_{i=1}^{n} \mu_i v_i^{(\ell)}(j) = \frac{2}{n}$. Therefore, if $b[\ell] = 0$, the social welfare of the outcome given bids $\boldsymbol{v}^{(\ell)}$ is $\frac{2}{n}$, which means that $u_{\boldsymbol{\mu}}\left(\boldsymbol{v}^{(\ell)}\right) = \frac{2}{n}$. Meanwhile, if $b[\ell] = 1$, the social welfare of the outcome given bids $\boldsymbol{v}^{(\ell)}$ is $\frac{1}{n}$, which means that $u_{\boldsymbol{\mu}}\left(\boldsymbol{v}^{(\ell)}\right) = \frac{1}{n}$. Thus, the set $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{((n-1)/2)}$ is shatterable with witnesses $\left(\frac{3n}{2}, \ldots, \frac{3n}{2}\right)$.

Claim 4.22. *For an arbitrary vector $\boldsymbol{b} \in \{0,1\}^{(n-1)/2}$, there exists a set of buyer weights $\mu_1, \ldots, \mu_n$ such that for all $\ell \in [(n-1)/2]$, if $b[\ell] = 0$, then $\operatorname{argmax}_{j \in \{1,2\}} \sum_{i=1}^{n} \mu_i v_i^{(\ell)}(j) = \frac{1}{n}$ and if $b[\ell] = 1$, then $\operatorname{argmax}_{j \in \{1,2\}} \sum_{i=1}^{n} \mu_i v_i^{(\ell)}(j) = \frac{2}{n}$.*

Proof. Let $\mu_1, \ldots, \mu_n$ be defined as follows:

$$\mu_i = \begin{cases} 1 & \text{if } i \text{ is odd and } i < n \\ 0 & \text{if } i \text{ is even and } b[i/2] = 0 \\ 2 & \text{if } i \text{ is even and } b[i/2] = 1 \\ 0 & \text{if } i = n. \end{cases}$$

Suppose $b[\ell] = 0$. By definition of $\boldsymbol{v}^{(\ell)}$, $\sum_{i=1}^n \mu_i v_i^{(\ell)}(1) = \mu_{2\ell-1} v_{2\ell-1}^{(\ell)}(1) + \mu_n v_n^{(\ell)}(1) = v_{2\ell-1}^{(\ell)}(1) = \frac{1}{n}$ and $\sum_{i=1}^n \mu_i v_i^{(\ell)}(2) = \mu_{2\ell} v_{2\ell}^{(\ell)}(2) = 0$. Therefore, $\operatorname{argmax}_{j\in\{1,2\}} \sum_{i=1}^n \mu_i v_i^{(\ell)}(j) = \frac{1}{n}$. Meanwhile, suppose $b[\ell] = 1$. Then $\sum_{i=1}^n \mu_i v_i^{(\ell)}(1) = \mu_{2\ell-1} v_{2\ell-1}^{(\ell)}(1) + \mu_n v_n^{(\ell)}(1) = v_{2\ell-1}^{(\ell)}(1) = \frac{1}{n}$ and

$$\sum_{i=1}^n \mu_i v_i^{(\ell)}(2) = \mu_{2\ell} v_{2\ell}^{(\ell)}(2) = \frac{2}{n}.$$

Therefore, $\operatorname{argmax}_{j\in\{1,2\}} \sum_{i=1}^n \mu_i v_i^{(\ell)}(j) = \frac{2}{n}$. □

□

4.3 Connections to prior research on generalization guarantees

In this section, we connect the piecewise decomposability of dual functions to prior research on generalization guarantees for algorithm configuration. The majority of these papers employed a case-by-case analysis, without developing the type of high-level structural insights we present. Using our main theorem, we match these existing generalization guarantees.

4.3.1 Clustering algorithms

A clustering instance (V, d) is made up of a set points and a distance metric V and $d : V \times V \to \mathbb{R}_{\geq 0}$. The goal is to split up the points into groups, or "clusters," so that within each group, distances are minimized and between each group, distances are maximized. Typically, a clustering's quality is quantified by some objective function. Classic choices include the k-means, k-median, or k-center objective functions. Unfortunately, finding the clustering that minimizes any one of these objectives is NP-hard. Clustering algorithms have uses in data science, computational biology [Navlakha et al., 2009], and many other fields.

Balcan et al. [2017] analyze *agglomerative clustering algorithms.* This type of algorithm requires a merge function $c(A, B) \to \mathbb{R}_{\geq 0}$, defining the distances between point sets $A, B \subseteq V$. The algorithm constructs a *cluster tree* $\mathcal{T}$. This tree starts with n leaf nodes, each containing a point from V. Over a series of rounds, the algorithm merges the sets with minimum distance according to c. The tree is complete when there is one node remaining, which consists of the set V. The children of each internal node T consist of the two sets merged to create the node. There are several common merge function c: $\min_{a\in A, b\in B} d(a, b)$ (single-linkage), $\frac{1}{|A|\cdot|B|} \sum_{a\in A, b\in B} d(a, b)$ (average-linkage), and $\max_{a\in A, b\in B} d(a, b)$ (complete-linkage). Following the linkage procedure, there is a dynamic programming step. This steps finds the tree pruning that minimizes an objective function, such as the k-means, -median, or -center objectives.

Balcan et al. [2017] study several families of merge functions:

$$\mathcal{C}_1 = \left\{ c_{1,\rho} : (A, B) \mapsto \left(\min_{u\in A, v\in B} (d(u, v))^\rho + \max_{u\in A, v\in B} (d(u, v))^\rho \right)^{1/\rho} \;\middle|\; \rho \in \mathbb{R} \cup \{\infty, -\infty\} \right\},$$

$$\mathcal{C}_2 = \left\{ c_{2,\rho} : (A,B) \mapsto \rho \min_{u\in A, v\in B} d(u,v) + (1-\rho) \max_{u\in A, v\in B} d(u,v) \;\middle|\; \rho \in [0,1] \right\},$$

$$\mathcal{C}_3 = \left\{ c_{3,\rho} : (A,B) \mapsto \left(\frac{1}{|A||B|} \sum_{u\in A, v\in B} (d(u,v))^\rho \right)^{1/\rho} \;\middle|\; \rho \in \mathbb{R} \cup \{\infty, -\infty\} \right\}.$$

The classes $\mathcal{C}_1$ and $\mathcal{C}_2$ interpolate between single- ($c_{1,-\infty}$ and $c_{2,1}$) and complete-linkage ($c_{1,\infty}$ and $c_{2,0}$). The class $\mathcal{C}_3$ includes as special cases average-, complete-, and single-linkage.

For each class $i \in \{1,2,3\}$ and each parameter ρ, let $A_{i,\rho}$ be the algorithm that takes as input a clustering instance (V,d) and returns the sequence ψ of merges the linkage algorithm makes using the merge function $c_{i,\rho}$. Let Ψ be the set of all possible *merge sequences* ($A_{i,\rho}(V,d) \in \Psi$). To evaluate the quality of a clustering, we assume access to a utility function $u : \Psi \to [-1,1]$. For example, $u\left(A_{i,\rho}(V,d)\right)$ might measure the distance between the ground truth clustering and the optimal k-means pruning of the cluster tree corresponding to $A_{i,\rho}(V,d)$.

Balcan et al. [2017] prove the following useful structure about the classes $\mathcal{C}_1$ and $\mathcal{C}_2$:

Lemma 4.23 (Balcan et al. [2017])**.** *Let (V,d) be an arbitrary clustering instance over n points. There is a partition of $\mathbb{R}$ into $k \le n^8$ intervals $I_1, \ldots, I_k$ such that for any interval I_j and any two parameters $\rho, \rho' \in I_j$, the sequences of merges the agglomerative clustering algorithm makes using the merge functions $c_{1,\rho}$ and $c_{1,\rho'}$ are identical. The same holds for the set of merge functions $\mathcal{C}_2$.*

This structure immediately implies that the corresponding class of utility functions has a piecewise-structured dual class.

Corollary 4.24. *Let u be a utility function mapping tuples (V,d,ψ) of clustering instances and merge sequences to the interval $[-1,1]$. Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \{u_\rho : (V,d) \mapsto u\left(V,d,A_{1,\rho}(V,d)\right) \mid \rho \in \mathbb{R} \cup \{-\infty,\infty\}\}$$

mapping clustering instances (V,d) to $[-1,1]$. The dual class $\mathcal{U}^*$ is $(\mathcal{W},\mathcal{G},n^8)$-piecewise decomposable, where $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\gamma \mapsto \mathbb{I}_{\{\gamma<a\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_\gamma \mapsto c$. The same holds when $\mathcal{U}$ is defined according to merge functions in $\mathcal{C}_2$ as $\mathcal{U} = \{u_\rho : (V,d) \mapsto u\left(V,d,A_{2,\rho}(V,d)\right) \mid \rho \in [0,1]\}$.

Corollaries 3.6 and 4.24 imply the following pseudo-dimension bound.

Corollary 4.25. *Let u be a utility function mapping tuples (V,d,ψ) of clustering instances and merge sequences to the interval $[-1,1]$. Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \{u_\rho : (V,d) \mapsto u\left(V,d,A_{1,\rho}(V,d)\right) \mid \rho \in \mathbb{R} \cup \{-\infty,\infty\}\}$$

mapping clustering instances (V,d) to $[-1,1]$. The pseudo-dimension of $\mathcal{U}$ is at most $4\ln\left(4en^8\right)$. The same holds when $\mathcal{U}$ is defined according to merge functions in $\mathcal{C}_2$ as

$$\mathcal{U} = \{u_\rho : (V,d) \mapsto u\left(V,d,A_{2,\rho}(V,d)\right) \mid \rho \in [0,1]\}.$$

Balcan et al. [2017] prove a similar guarantee for the more complicated class $\mathcal{C}_3$.

Lemma 4.26 (Balcan et al. [2017])**.** *Let (V,d) be an arbitrary clustering instance over n points. There is a partition of $\mathbb{R}$ into $k \le n^2 3^{2n}$ intervals $I_1, \ldots, I_k$ such that for any interval I_j and any two parameters $\rho, \rho' \in I_j$, the sequences of merges the agglomerative clustering algorithm makes using the merge functions $c_{3,\rho}$ and $c_{3,\rho'}$ are identical.*

Again, this structure immediately implies that the corresponding class of utility functions has a piecewise-structured dual class.

Corollary 4.27. *Let u be a utility function mapping tuples (V, d, ψ) of clustering instances and merge sequences to the interval $[-1, 1]$. Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \{u_\rho : (V, d) \mapsto u\left(V, d, A_{1,\rho}(V, d)\right) \mid \rho \in \mathbb{R} \cup \{-\infty, \infty\}\}$$

mapping clustering instances (V, d) to $[-1, 1]$. The dual class $\mathcal{U}^*$ is $\left(\mathcal{W}, \mathcal{G}, n^2 3^{2n}\right)$-piecewise decomposable, where $\mathcal{G} = \{g_a : \mathcal{U} \to \{0, 1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\gamma \mapsto \mathbb{I}_{\{\gamma < a\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_\gamma \mapsto c$.

Corollaries 3.6 and 4.27 imply the following pseudo-dimension bound.

Corollary 4.28. *Let u be a utility function mapping tuples (V, d, ψ) of clustering instances and merge sequences to the interval $[-1, 1]$. Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \{u_\rho : (V, d) \mapsto u\left(V, d, A_{3,\rho}(V, d)\right) \mid \rho \in \mathbb{R} \cup \{-\infty, \infty\}\}$$

mapping clustering instances (V, d) to $[-1, 1]$. The pseudo-dimension of $\mathcal{U}$ is $4 \ln\left(4en^2 3^{2n}\right)$.

Corollaries 4.25 and 4.28 match the pseudo-dimension guarantees that Balcan et al. [2017] prove.

4.3.2 Integer programming

Balcan et al. [2017, 2018a] study algorithm configuration for both integer linear and integer quadratic programming, as we describe below.

Integer linear programming. In the context of integer linear programming, Balcan et al. [2018a] focus on branch-and-bound (B&B) [Land and Doig, 1960], an algorithm for solving mixed integer linear programs (MILPs). A MILP is defined by a matrix $A \in \mathbb{R}^{m \times n}$, a vector $\boldsymbol{b} \in \mathbb{R}^m$, a vector $\boldsymbol{c} \in \mathbb{R}^n$, and a set of indices $I \subseteq [n]$. The goal is to find a vector $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{c} \cdot \boldsymbol{x}$ is maximized, $A\boldsymbol{x} \leq \boldsymbol{b}$, and for every index $i \in I$, x_i is constrained to be binary: $x_i \in \{0, 1\}$.

Branch-and-bound builds a search tree to solve an input MILP Q. At the root of the search tree is the original MILP Q. At each round, the algorithm chooses a leaf of the search tree, which represents an MILP Q'. 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 x_i: it sets the left child of Q' to be that same integer program, but with the additional constraint that $x_i = 0$, and it sets the right child of Q' to be that same integer program, but with the additional constraint that $x_i = 1$. The algorithm *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. The algorithm terminates when it has fathomed every leaf. At that point, we can guarantee that the best solution to Q found so far is optimal. See the paper by Balcan et al. [2018a] for more details.

Balcan et al. [2018a] study *mixed integer linear programs* (MILPs) where the goal is to maximize an objective function $\boldsymbol{c}^\top \boldsymbol{x}$ subject to the constraints that $A\boldsymbol{x} \leq \boldsymbol{b}$ and that some of the components of $\boldsymbol{x}$ are contained in $\{0, 1\}$. Given a MILP Q, we use the notation $\breve{\boldsymbol{x}}_Q = (\breve{x}_Q[1], \ldots \breve{x}_Q[n])$ to denote an optimal solution to the MILP's LP relaxation. We denote the optimal objective value to the MILP's LP relaxation as $\breve{c}_Q$, which means that $\breve{c}_Q = \boldsymbol{c}^\top \breve{\boldsymbol{x}}_Q$.

Branch-and-bound systematically partitions the feasible set in order to find an optimal solution, organizing the partition as a tree. At the root of this tree is the original integer program. Each child

represents the simplified integer program obtained by partitioning the feasible set of the problem contained in the parent node. The algorithm prunes a branch if the corresponding subproblem is infeasible or its optimal solution cannot be better than the best one discovered so far. Oftentimes, branch-and-bound partitions the feasible set by adding a constraint. For example, if the feasible set is characterized by the constraints $A\boldsymbol{x} \leq \boldsymbol{b}$ and $\boldsymbol{x} \in \{0,1\}^n$, the algorithm partition the feasible set into one subset where $A\boldsymbol{x} \leq \boldsymbol{b}$, $x_1 = 0$, and $x_2, \ldots, x_n \in \{0,1\}$, and another where $A\boldsymbol{x} \leq \boldsymbol{b}$, $x_1 = 1$, and $x_2, \ldots, x_n \in \{0,1\}$. In this case, we say that the algorithm *branches* on x_1.

Balcan et al. [2018a] show how to learn variable selection policies. Specifically, they study *score-based variable selection policies*, defined below.

Definition 4.6 (Score-based variable selection policy [Balcan et al., 2018a])**.** Let $\texttt{score}$ be a deterministic function that takes as input a partial search tree $\mathcal{T}$, a leaf Q of that tree, and an index i, and returns a real value $\texttt{score}(\mathcal{T}, Q, i) \in \mathbb{R}$. For a leaf Q of a tree $\mathcal{T}$, let $N_{\mathcal{T},Q}$ be the set of variables that have not yet been branched on along the path from the root of $\mathcal{T}$ to Q. A score-based variable selection policy selects the variable $\operatorname{argmax}_{x_i \in N_{\mathcal{T},Q}}\{\texttt{score}(\mathcal{T}, Q, i)\}$ to branch on at the node Q.

This type of variable selection policy is widely used [Linderoth and Savelsbergh, 1999, Achterberg, 2009, Gilpin and Sandholm, 2011]. See the paper by Balcan et al. [2018a] for examples.

Given d arbitrary scoring rules $\texttt{score}_1, \ldots, \texttt{score}_d$, Balcan et al. [2018a] provide guidance for learning a linear combination $\rho_1\texttt{score}_1 + \cdots + \rho_d\texttt{score}_d$ that leads to small expected tree sizes. They assume that all aspects of the tree search algorithm except the variable selection policy, such as the node selection policy, are fixed. In their analysis, they prove the following lemma.

Lemma 4.29 (Balcan et al. [2018a])**.** *Let $\texttt{score}_1, \ldots, \texttt{score}_d$ be d arbitrary scoring rules and let Q be an arbitrary MILP over n binary variables. Suppose we limit B&B to producing search trees of size τ. There is a set $\mathcal{H}$ of at most $n^{2(\tau+1)}$ hyperplanes such that for any connected component R of $[0,1]^d \setminus \mathcal{H}$, the search tree B&B builds using the scoring rule $\rho_1\texttt{score}_1 + \cdots + \rho_d\texttt{score}_d$ is invariant across all $(\rho_1, \ldots, \rho_d) \in R$.*

This piecewise structure immediately implies the following guarantee.

Corollary 4.30. *Let $\texttt{score}_1, \ldots, \texttt{score}_d$ be d arbitrary scoring rules and let Q be an arbitrary MILP over n binary variables. Suppose we limit B&B to producing search trees of size τ. For each parameter vector $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_d) \in [0,1]^d$, let $u_{\boldsymbol{\rho}}(Q)$ be the size of the tree, divided by τ, that B&B builds using the scoring rule $\rho_1\texttt{score}_1 + \cdots + \rho_d\texttt{score}_d$ given Q as input. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \left\{u_\rho \mid \boldsymbol{\rho} \in [0,1]^d\right\}$ mapping MILPs to $[0,1]$. The dual class $\mathcal{U}^*$ is $\left(\mathcal{W}, \mathcal{G}, n^{2(\tau+1)}\right)$-piecewise decomposable, where $\mathcal{G} = \{g_{\boldsymbol{a},\theta} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^d, \theta \in \mathbb{R}\}$ consists of halfspace indicator functions $g_{\boldsymbol{a},\theta} : u_{\boldsymbol{\rho}} \mapsto \mathbb{I}_{\{\boldsymbol{\rho}\cdot\boldsymbol{a} \leq \theta\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_\gamma \mapsto c$.*

Corollaries 3.6 and 4.30 imply the following pseudo-dimension bound.

Corollary 4.31. *Let $\texttt{score}_1, \ldots, \texttt{score}_d$ be d arbitrary scoring rules and let Q be an arbitrary MILP over n binary variables. Suppose we limit B&B to producing search trees of size τ. For each parameter vector $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_d) \in [0,1]^d$, let $u_{\boldsymbol{\rho}}(Q)$ be the size of the tree, divided by τ, that B&B builds using the scoring rule $\rho_1\texttt{score}_1 + \cdots + \rho_d\texttt{score}_d$ given Q as input. Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \left\{u_\rho \mid \boldsymbol{\rho} \in [0,1]^d\right\}$ mapping MILPs to $[0,1]$. The pseudo-dimension of $\mathcal{U}$ is at most $4\,(d+1)\ln\left(4en^{2(\tau+1)}\,(d+1)\right)$.*

Corollary 4.31 matches the pseudo-dimension guarantee that Balcan et al. [2018a] prove.

Algorithm 1 SDP rounding algorithm with rounding function r

Input: Matrix $A \in \mathbb{R}^{n \times n}$.

1: Draw a random vector $\boldsymbol{Z}$ from $\mathcal{Z}$, the n-dimensional Gaussian distribution.
2: Solve the SDP (18) for the optimal embedding $U = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$.
3: Compute set of fractional assignments $r(\langle \boldsymbol{Z}, \boldsymbol{u}_1 \rangle), \ldots, r(\langle \boldsymbol{Z}, \boldsymbol{u}_n \rangle)$.
4: For all $i \in [n]$, set x_i to 1 with probability $\frac{1}{2} + \frac{1}{2} \cdot r\left(\langle \boldsymbol{Z}, \boldsymbol{u}_i \rangle\right)$ and -1 with probability $\frac{1}{2} - \frac{1}{2} \cdot r\left(\langle \boldsymbol{Z}, \boldsymbol{u}_i \rangle\right)$.

Output: $x_1, \ldots, x_n$.

Integer quadratic programming. A diverse array of NP-hard problems, including max-2SAT, max-cut, and correlation clustering, can be characterized as integer quadratic programs (IQPs). An IQP is represented by a matrix $A \in \mathbb{R}^{n \times n}$. The goal is to find a set $X = \{x_1, \ldots, x_n\} \in \{-1, 1\}^n$ maximizing $\sum_{i,j \in [n]} a_{ij} x_i x_j$. The most-studied IQP approximation algorithms operate via an SDP relaxation:

$$\text{maximize} \sum_{i,j \in [n]} a_{ij} \langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle \qquad \text{subject to } \boldsymbol{u}_i \in S^{n-1}. \tag{18}$$

The approximation algorithm must transform, or "round," the unit vectors into a binary assignment of the variables $x_1, \ldots, x_n$. In the seminal *GW algorithm* [Goemans and Williamson, 1995], the algorithm projects the unit vectors onto a random vector $\boldsymbol{Z}$, which it draws from the n-dimensional Gaussian distribution, which we denote using $\boldsymbol{Z}$. If $\langle \boldsymbol{u}_i, \boldsymbol{Z} \rangle > 0$, it sets $x_i = 1$. Otherwise, it sets $x_i = -1$.

The GW algorithm's approximation ratio can sometimes be improved if the algorithm probabilistically assigns the binary variables. In the final step, the algorithm can use any rounding function $r : \mathbb{R} \to [-1, 1]$ to set $x_i = 1$ with probability $\frac{1}{2} + \frac{1}{2} \cdot r\left(\langle \boldsymbol{Z}, \boldsymbol{u}_i \rangle\right)$ and $x_i = -1$ with probability $\frac{1}{2} - \frac{1}{2} \cdot r\left(\langle \boldsymbol{Z}, \boldsymbol{u}_i \rangle\right)$. See Algorithm 1 for the pseudocode. Algorithm 1 is known as a *Random Projection, Randomized Rounding* (RPR^2) algorithm, so named by the seminal work of Feige and Langberg [2006].

Balcan et al. [2017] analyze s-linear rounding functions [Feige and Langberg, 2006] $\phi_s : \mathbb{R} \to [-1, 1]$, parameterized by $s > 0$, defined as follows:

$$\phi_s(y) = \begin{cases} -1 & \text{if } y < -s \\ y/s & \text{if } -s \leq y \leq s \\ 1 & \text{if } y > s. \end{cases}$$

The goal is to learn a parameter s such that in expectation, $\sum_{i,j \in [n]} a_{ij} x_i x_j$ is maximized. The expectation is over several sources of randomness: first, the distribution $\mathcal{D}$ over matrices A; second, the vector $\boldsymbol{Z}$; and third, the assignment of $x_1, \ldots, x_n$. This final assignment depends on the parameter s, the matrix A, and the vector $\boldsymbol{Z}$. Balcan et al. [2017] refer to this value as the *true utility of the parameter* s. Note that the distribution over matrices, which defines the algorithm's input, is unknown and external to the algorithm, whereas the Gaussian distribution over vectors as well as the distribution defining the variable assignment are internal to the algorithm.

The distribution over matrices is unknown, so we cannot know any parameter's true utility. Therefore, to learn a good parameter s, we must use samples. Balcan et al. [2017] suggest drawing samples from two sources of randomness: the distributions over vectors and matrices. In other words, they suggest drawing a set of samples $\mathcal{S} = \left\{\left(A^{(1)}, \boldsymbol{Z}^{(1)}\right), \ldots, \left(A^{(m)}, \boldsymbol{Z}^{(m)}\right)\right\} \sim (\mathcal{D} \times \mathcal{Z})^m$. Given these samples, Balcan et al. [2017] define a parameter's *empirical utility* to be the expected

objective value of the solution Algorithm 1 returns given input A, using the vector $\boldsymbol{Z}$ and ϕ_s in Step 3, on average over all $(A, \boldsymbol{Z}) \in \mathcal{S}$. Generally speaking, Balcan et al. [2017] suggest sampling the first two randomness sources in order to isolate the third randomness source. They argue that this third source of randomness has an expectation that is simple to analyze. Using pseudo-dimension, they prove that every parameter s, its empirical and true utilities converge.

A bit more formally, Balcan et al. [2017] use the notation $p_{(i,\boldsymbol{Z},A,s)}$ to denote the distribution that the binary value x_i is drawn from when Algorithm 1 is given A as input and uses the rounding function $r = \phi_s$ and the hyperplane $\boldsymbol{Z}$ in Step 3. Using this notation, the parameter s has a true utility of $\mathbb{E}_{A,\boldsymbol{Z}\sim\mathcal{D}\times\mathcal{Z}}\left[\mathbb{E}_{x_i\sim p_{(i,\boldsymbol{Z},A,s)}}\left[\sum_{i,j} a_{ij}x_ix_j\right]\right]$.[6] We also use the notation $u_s(A, \boldsymbol{Z})$ to denote the expected objective value of the solution Algorithm 1 returns given input A, using the vector $\boldsymbol{Z}$ and ϕ_s in Step 3. The expectation is over the final assignment of each variable x_i. Specifically, $u_s(A, \boldsymbol{Z}) = \mathbb{E}_{x_i\sim p_{(i,\boldsymbol{Z},A,s)}}\left[\sum_{i,j} a_{ij}x_ix_j\right]$. By definition, a parameter's true utility equals $\mathbb{E}_{A,\boldsymbol{Z}\sim\mathcal{D}\times\mathcal{Z}}\left[u_s(A, \boldsymbol{Z})\right]$. Given a set $\left(A^{(1)}, \boldsymbol{Z}^{(1)}\right), \ldots, \left(A^{(m)}, \boldsymbol{Z}^{(m)}\right) \sim \mathcal{D} \times \mathcal{Z}$, a parameter's empirical utility is $\frac{1}{m}\sum_{i=1}^{m} u_s\left(A^{(i)}, \boldsymbol{Z}^{(i)}\right)$.

Both we and Balcan et al. [2017] bound the pseudo-dimension of the function class $\mathcal{U} = \{u_s : s > 0\}$. Balcan et al. [2017] prove that the functions in $\mathcal{U}$ are piecewise structured: roughly speaking, for a fixed matrix A and vector $\boldsymbol{Z}$, each function in $\mathcal{U}$ is a piecewise, inverse-quadratic function of the parameter s. To present this lemma, we use the following notation: given a tuple $(A, \boldsymbol{Z})$, let $u_{A,\boldsymbol{Z}} : \mathbb{R} \to \mathbb{R}$ be defined such that $u_{A,\boldsymbol{Z}}(s) = u_s\left(A, \boldsymbol{Z}\right)$.

Lemma 4.32 (Balcan et al. [2017])**.** *For any matrix A and vector $\boldsymbol{Z}$, the function $u_{A,\boldsymbol{Z}} : \mathbb{R}_{>0} \to \mathbb{R}$ is made up of $n + 1$ piecewise components of the form $\frac{a}{s^2} + \frac{b}{s} + c$ for some $a, b, c \in \mathbb{R}$. Moreover, if the border between two components falls at some $s \in \mathbb{R}_{>0}$, then it must be that $s = |\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle|$ for some $\boldsymbol{u}_i$ in the optimal SDP embedding of A.*

This piecewise structure immediately implies the following corollary about the dual class $\mathcal{U}^*$.

Corollary 4.33. *Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_s : s > 0\}$. The dual class $\mathcal{U}^*$ is $(\mathcal{W}, \mathcal{G}, n)$-piecewise decomposable, where $\mathcal{G} = \{g_a : \mathcal{U} \to \{0, 1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_s \mapsto \mathbb{I}_{\{s\leq a\}}$ and $\mathcal{W} = \{w_{a,b,c} : \mathcal{U} \to \mathbb{R} \mid a, b, c \in \mathbb{R}\}$ consists of inverse-quadratic functions $w_{a,b,c} : u_s \mapsto \frac{a}{s^2} + \frac{b}{s} + c$.*

Corollaries 3.6 and 4.33 imply the following pseudo-dimension bound.

Corollary 4.34. *Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_s : s > 0\}$. The pseudo-dimension of $\mathcal{U}$ is at most $4\left(4\ln 768 + 1\right)\ln\left(4en\left(4\ln 768 + 1\right)\right)$.*

Corollary 4.34 matches the pseudo-dimension bound that Balcan et al. [2017] prove.

4.3.3 Greedy algorithms

Gupta and Roughgarden [2017] provide pseudo-dimension bounds for greedy algorithm configuration, analyzing two canonical combinatorial problems: the maximum weight independent set problem and the knapsack problem. We recover their bounds in both cases.

[6] We, like Balcan et al. [2017], use the abbreviated notation

$$\mathbb{E}_{A,\boldsymbol{Z}\sim\mathcal{D}\times\mathcal{Z}}\left[\mathbb{E}_{x_i\sim p_{(i,\boldsymbol{Z},A,s)}}\left[\sum_{i,j} a_{ij}x_ix_j\right]\right] = \mathbb{E}_{A,\boldsymbol{Z}\sim\mathcal{D}\times\mathcal{Z}}\left[\mathbb{E}_{x_1\sim p_{(1,\boldsymbol{Z},A,s)},\ldots,x_n\sim p_{(n,\boldsymbol{Z},A,s)}}\left[\sum_{i,j} a_{ij}x_ix_j\right]\right].$$

Maximum weight independent set (MWIS). In the MWIS problem, there is a graph and a weight $w(v) \in \mathbb{R}_{\geq 0}$ for each vertex v. The goal is to find a set of non-adjacent vertices with maximum weight. The classic greedy algorithm proceeds over a series of rounds: on each round, it adds the vertex v that maximizes $w(v) / (1 + \deg(v))$ to the independent set and deletes v and its neighbors from the graph. Gupta and Roughgarden [2017] propose the greedy heuristic $w(v) / (1 + \deg(v))^{\rho}$ where $\rho \geq 0$ is a tunable parameter. We represent a graph as a tuple $(\boldsymbol{w}, \boldsymbol{e}) \in \mathbb{R}^n \times \{0,1\}^{\binom{n}{2}}$, ordering the vertices $v_1, \ldots, v_n$ in a fixed but arbitrary way. In this context, the function u_ρ maps each graph $(\boldsymbol{w}, \boldsymbol{e})$ to the weight of the vertices in the set returned by the algorithm parameterized by ρ, denoted $u_\rho(\boldsymbol{w}, \boldsymbol{e})$. Gupta and Roughgarden [2017] implicitly prove the following lemma about each function u_ρ (made explicit in work by Balcan et al. [2018b]). To present this lemma, we use the following notation: given a tuple $(\boldsymbol{w}, \boldsymbol{e})$, let $u_{\boldsymbol{w},\boldsymbol{e}} : \mathbb{R} \to \mathbb{R}$ be defined such that $u_{\boldsymbol{w},\boldsymbol{e}}(\rho) = u_\rho(\boldsymbol{w}, \boldsymbol{e})$.

Lemma 4.35 (Gupta and Roughgarden [2017])**.** *For any tuple $(\boldsymbol{w}, \boldsymbol{e})$, the function $u_{\boldsymbol{w},\boldsymbol{e}} : \mathbb{R} \to \mathbb{R}$ is piecewise constant with at most n^4 discontinuities.*

This structure immediately implies that the function class $\mathcal{U} = \{u_\rho : \rho > 0\}$ has a piecewise-structured dual class.

Corollary 4.36. *Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_\rho : \rho > 0\}$. The dual class $\mathcal{U}^*$ is $(\mathcal{W}, \mathcal{G}, n^4)$-piecewise decomposable, where $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\gamma \mapsto \mathbb{I}_{\{\gamma < a\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_\gamma \mapsto c$.*

Corollaries 3.6 and 4.36 imply the following pseudo-dimension bound.

Corollary 4.37. *Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_\rho : \rho > 0\}$. The pseudo-dimension of $\mathcal{U}$ is at most $4 \ln\left(4en^4\right)$.*

This matches the pseudo-dimension bound by Gupta and Roughgarden [2017].

Knapsack. Moving to the classic knapsack problem, the input is a knapsack capacity C and a set of n items i each with a value ν_i and a size s_i. The goal is to determine a set $I \subseteq \{1, \ldots, n\}$ with maximium total value $\sum_{i \in I} \nu_i$ such that $\sum_{i \in I} s_i \leq C$. Gupta and Roughgarden [2017] suggest the family of algorithms parameterized by $\rho > 0$ where each algorithm returns the better of the following two solutions:

- Greedily pack items in order of nonincreasing value ν_i subject to feasibility.
- Greedily pack items in order of ν_i / s_i^ρ subject to feasibility.

It is well-known that the algorithm with $\rho = 1$ achieves a 2-approximation. We use the notation $u_\rho(\boldsymbol{\nu}, \boldsymbol{s}, C)$ to denote the total value of the items returned by the algorithm parameterized by ρ given input $(\boldsymbol{\nu}, \boldsymbol{s}, C)$.

Gupta and Roughgarden [2017] prove the following fact about the functions u_ρ (made explicit in work by Balcan et al. [2018b]). To present this lemma, we use the following notation: given a tuple $(\boldsymbol{\nu}, \boldsymbol{s}, C)$, let $u_{\boldsymbol{\nu},\boldsymbol{s},C} : \mathbb{R} \to \mathbb{R}$ be defined such that $u_{\boldsymbol{\nu},\boldsymbol{s},C}(\rho) = u_\rho(\boldsymbol{\nu}, \boldsymbol{s}, C)$.

Lemma 4.38 (Gupta and Roughgarden [2017])**.** *For any tuple $(\boldsymbol{\nu}, \boldsymbol{s}, C)$, the function $u_{\boldsymbol{\nu},\boldsymbol{s},C} : \mathbb{R} \to \mathbb{R}$ is piecewise constant with at most n^2 discontinuities.*

This structure immediately implies that the function class $\mathcal{U} = \{u_\rho : \rho > 0\}$ has a piecewise-structured dual class.

Corollary 4.39. *Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_\rho : \rho > 0\}$. The dual class $\mathcal{U}^*$ is $(\mathcal{W}, \mathcal{G}, n^2)$-piecewise decomposable, where $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\gamma \mapsto \mathbb{I}_{\{\gamma < a\}}$ and $\mathcal{W} = \{w_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $w_c : u_\gamma \mapsto c$.*

Corollaries 3.6 and 4.39 imply the following pseudo-dimension bound.

Corollary 4.40. *Let $\mathcal{U}$ be the set of functions $\mathcal{U} = \{u_\rho : \rho > 0\}$. The pseudo-dimension of $\mathcal{U}$ is at most $4 \ln \left(4en^2\right)$.*

This matches the pseudo-dimension bound by Gupta and Roughgarden [2017].

4.3.4 Revenue maximization

The design of revenue-maximizing multi-item mechanisms is a notoriously challenging problem. Remarkably, the revenue-maximizing mechanism is not known even when there are just two items for sale. In this setting, the mechanism designer's goal is to field a mechanism with high expected revenue on the distribution over agents' values. Balcan et al. [2018c] study generalization guarantees for mechanism design in the context of revenue maximization. They focus on sales settings: there is a seller, not included among the agents, who will use a mechanism to allocate a set of goods among the agents. The agents submit bids describing their values for the goods for sale. The mechanism determines which agents receive which items and how much the agents pay. The seller's revenue is the sum of the agents' payments. The mechanism designer's goal is to select a mechanism that maximizes the revenue. In contrast to the mechanisms we analyze in Section 4.2, Balcan et al. [2018c] study mechanisms that are not necessarily budget-balanced. Specifically, under every mechanism they study, the sum of the agents' payments—the revenue—is at least zero. As in Section 4.2, all of the mechanisms they analyze are incentive compatible.

Balcan et al. [2018c] provide generalization guarantees for a variety of widely-studied, parameterized mechanism classes, including posted-price mechanisms, multi-part tariffs, second-price auctions with reserves, affine maximizer auctions, virtual valuations combinatorial auctions mixed-bundling auctions, and randomized mechanisms. They do so by uncovering structure shared by all of these mechanisms: for any set of buyers' values, revenue is a piecewise linear function of the mechanism's parameters. This structure is captured by our definition of piecewise decomposability. Moreover, we recover their generalization guarantees.

Balcan et al. [2018c] study the problem of selling m heterogeneous goods to n buyers. They denote a bundle of goods as a subset $b \subseteq [m]$. Each buyer $j \in [n]$ has a valuation function $v_j : 2^{[m]} \to \mathbb{R}$ over bundles of goods. The set Π of problem instances consists of n-tuples of buyer values $\boldsymbol{v} = (v_1, \ldots, v_n)$. As in Section 4.2, every mechanism that Balcan et al. [2018c] study is defined by an *allocation function* and a set of *payment functions*. Every auction in the classes they study is incentive compatible, so they assume that the bids equal the bidders' valuations. An allocation function $f : \Pi \to \left(2^{[m]}\right)^n$ maps the values $\boldsymbol{v} \in \Pi$ to a division of the goods $(b_1, \ldots, b_n) \in \left(2^{[m]}\right)^n$, where $b_i \subseteq [m]$ is the set of goods buyer i receives. For each agent $i \in [n]$, there is a payment function $p_i : \Pi \to \mathbb{R}$ which maps values $\boldsymbol{v} \in \Pi$ to a payment $p_i(\boldsymbol{v}) \in \mathbb{R}_{\geq 0}$ that agent i must make.

Balcan et al. [2018c] study a variety of mechanism classes, each of which is parameterized by a d-dimensional vector $\boldsymbol{\rho} \in \mathcal{P} \subseteq \mathbb{R}^d$ for some $d \geq 1$. For example, when $d = m$, $\boldsymbol{\rho}$ might be a vector of prices for each of the items. The revenue of a mechanism is the sum of the agents' payments. Given a mechanism parameterized by a vector $\boldsymbol{\rho} \in \mathbb{R}^d$, we denote the revenue as $u_{\boldsymbol{\rho}} : \Pi \to \mathbb{R}$, where $u_{\boldsymbol{\rho}}(\boldsymbol{v}) = \sum_{i=1}^n p_i(\boldsymbol{v})$.

Balcan et al. [2018c] provide psuedo-dimension bounds for any mechanism class that is *delineable*. To define this notion, for any fixed valuation vector $\boldsymbol{v} \in \Pi$, we use the notation $u_{\boldsymbol{v}}(\boldsymbol{\rho})$ to denote revenue as a function of the mechanism's parameters.

Definition 4.7 ((d,t)-delineable [Balcan et al., 2018c])**.** A mechanism class is (d,t)-*delineable* if:

1. The class consists of mechanisms parameterized by vectors $\boldsymbol{p}$ from a set $\mathcal{P} \subseteq \mathbb{R}^d$; and
2. For any $\boldsymbol{v} \in \Pi$, there is a set $\mathcal{H}$ of t hyperplanes such that for any connected component $\mathcal{P}'$ of $\mathcal{P} \setminus \mathcal{H}$, the function $u_{\boldsymbol{v}}(\boldsymbol{p})$ is linear over $\mathcal{P}'$.

Delineability naturally translates to decomposability, as we formalize below.

Lemma 4.41. *Let $\mathcal{U}$ be a set of revenue functions corresponding to a (d,t)-delineable mechanism class. The dual class $\mathcal{U}^*$ is $(\mathcal{W},\mathcal{G},t)$-piecewise decomposable, where $\mathcal{G} = \{g_{\boldsymbol{a},\theta} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^d, \theta \in \mathbb{R}\}$ consists of halfspace indicator functions $g_{\boldsymbol{a},\theta} : u_{\boldsymbol{\rho}} \mapsto \mathbb{I}_{\{\boldsymbol{\rho}\cdot\boldsymbol{a} \le \theta\}}$ and $\mathcal{W} = \{w_{\boldsymbol{a},\theta} : \mathcal{U} \to \mathbb{R} \mid \boldsymbol{a} \in \mathbb{R}^d, \theta \in \mathbb{R}\}$ consists of linear functions $w_{\boldsymbol{a},\theta} : u_{\boldsymbol{\rho}} \mapsto \boldsymbol{\rho}\cdot\boldsymbol{a} + \theta$.*

Lemmas 3.7 and 4.41 imply the following bound.

Corollary 4.42. *Let $\mathcal{U}$ be a set of revenue functions corresponding to a (d,t)-delineable mechanism class. The pseudo-dimension of $\mathcal{U}$ is at most $8(d+1)\ln(8et(d+1))$.*

Corollary 4.42 matches the pseudo-dimension bound that Balcan et al. [2017] prove.

4.4 Experiments

In this section, we provide experiments demonstrating that parameter tuning can have a significant impact on an algorithm's solution quality. Moreover, we show that the more samples we use to tune parameters, the better the resulting algorithm.

4.4.1 Sequence alignment experiments

Changing the alignment parameter can alter the accuracy of the produced alignments. Figure 8 shows the regions of the gap-open/gap-extension penalty plane divided into regions such that each region corresponds to a different computed alignment. The regions in the figure are produced using the `XPARAL` software of Gusfield and Stelling [1996], with using the BLOSUM62 amino acid replacement matrix, the scores in each region were computed using Robert Edgar's `qscore` package[7]. The alignment sequences are a single pairwise alignment from the data set described below.

To test the influence of the training set size on the parameters chosen for pairwise sequence alignment we use the `IPA` tool [Kim and Kececioglu, 2007] to learn optimal parameter choices for a given set of example pairwise sequence alignments. We used 861 protein multiple sequence alignment benchmarks that had been previously been used in DeBlasio and Kececioglu [2018], which split these benchmarks into 12 cross-validation folds that evenly distributed the "difficulty" of an alignment (the accuracy of the alignment produced using aligner defaults parameter choice). All pairwise alignments were extracted from each multiple sequence alignment. We then took randomized increasing sized subsets of the pairwise sequence alignments from each training set and found the optimal parameter choices for affine gap costs and alphabet-dependent substitution costs. These parameters were then given to the `Opal` aligner [v3.1b, Wheeler and Kececioglu, 2007] to realign the pairwise alignments in the associated test sets.

Figure 9 shows the impact of increasing the number of training examples used to optimize parameter choices. As the number of training examples increases, the optimized parameter choice is less able to fit the training data exactly and thus the training accuracy decreases, for the same

[7] `http://drive5.com/qscore/`

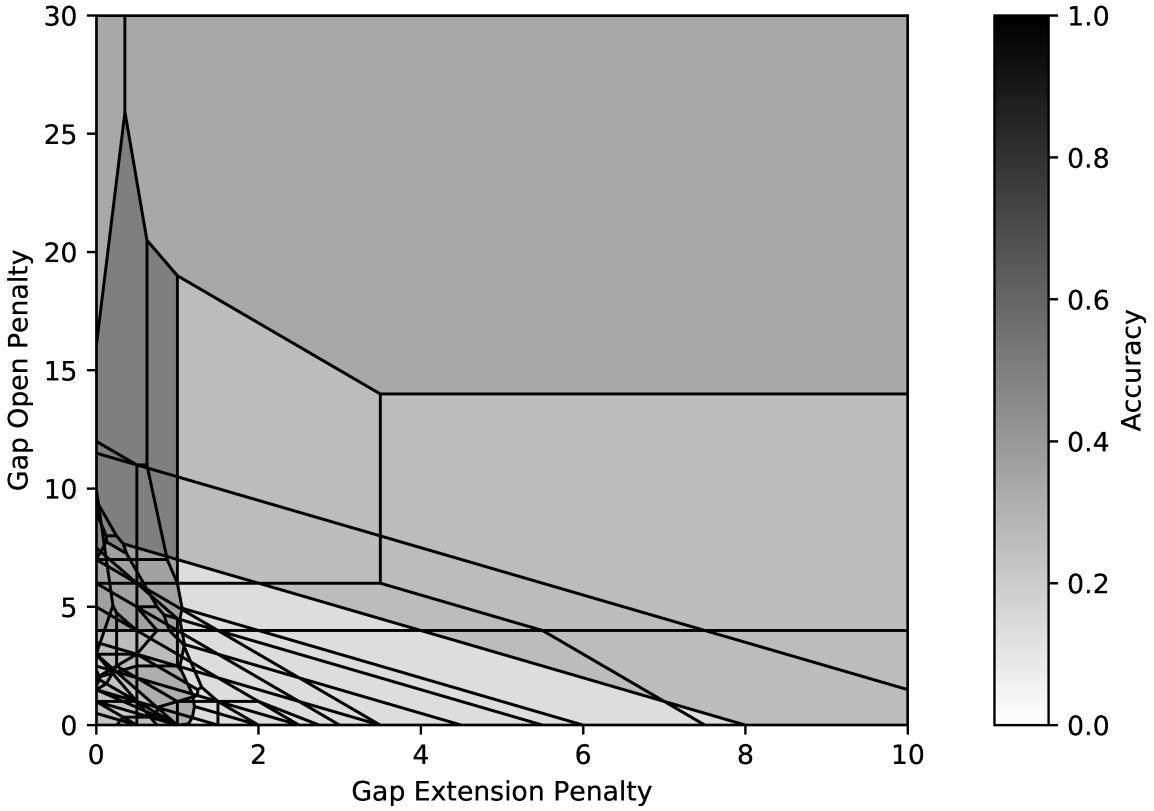


Figure 8: Parameter space decomposition for a single example.

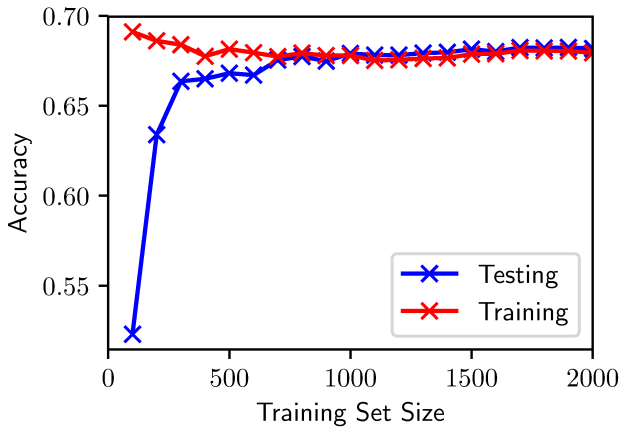


Figure 9: Average accuracy of training and test examples using parameter choices optimized for various training set sizes.

reason the parameter choices are more general and the test accuracy increases. The test and training accuracies are roughly equal when the training set size is close to 1000 examples and remains equal for larger training sets. This number is much lower than the predicted number of samples needed for generalization. The test accuracy is actually slightly higher and this is likely due to the training subset not representing the distribution of inputs as well as the full test set due to the randomization being on all of the alignments rather than across difficulty as was done to create the cross-validation separations.

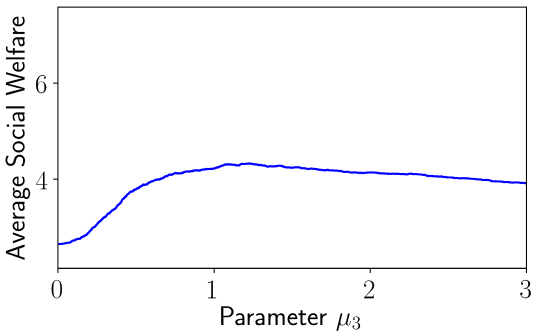
4.4.2 Mechanism design experiments

We now demonstrate that tuning *neutral affine maximizer (NAM)* (Definition 4.5 in Section 4.2) parameters can have a substantial effect on the resulting social welfare.

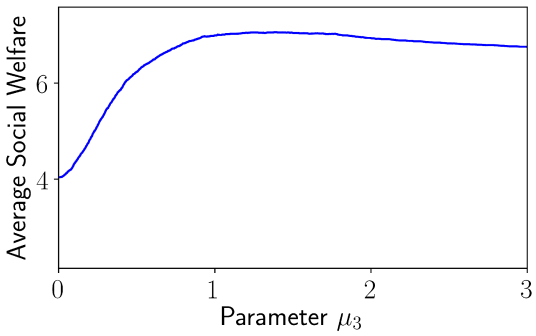
Experimental setup. Following work by Nath and Sandholm [2019] on NAMs, we use the Jester Collaborative Filtering Dataset [Goldberg et al., 2001], which consists of ratings from 24,983 users of 100 jokes—in this example, the jokes could be proxies for comedians, one of whom you and your college alumni council will hire for your upcoming reunion. In our setup, there will be three council members who are members of a large pool of alumni (the 24,983 users). We filter out 90 of the jokes that are sparsely rated, narrowing down on 10 jokes, each of which was rated by a total of 24,952 users. The ratings are continuous and range from -10 to 10. To aid the visualization of our results, we set the number n of agents equal to 3 (we describe the distribution over agents in the next paragraph). To ensure that the social welfare of any joke is contained in $[-1, 1]$, we divide each agents' rating by 30. We use the notation V to denote the set of all users' ratings for the 10 jokes, so $V \subseteq \left[-\frac{1}{3}, \frac{1}{3}\right]^{10}$ and $|V| = 24,952$.

Our goal is to learn a neutral affine maximizer (Definition 4.5 in Section 4.2) that takes as input three agents' bids for the ten jokes and returns one of the ten jokes along with a set of payments that each agent will either pay or receive. Our learning algorithm receives a set of valuation vectors sampled from a distribution over three agents' preferences. For our experimental setup, one option would be to define the distribution over agents to be uniform over V. However, then all agents would be identical in expectation, and this type of homogeneity is unrealistic in real-world settings. In order to define a distribution over heterogeneous agents, we categorize the users into two groups which we call *differentiating* and *indifferent*. A user is *differentiating* if the sample variance of their ratings for the 10 jokes is greater than 20. Meanwhile, we call a user is *indifferent* if the sample variance of their ratings for the 10 jokes is less than 15. Let $V_i \subseteq \left[-\frac{1}{3}, \frac{1}{3}\right]^{10}$ be the set of indifferent users and let $V_d \subseteq \left[-\frac{1}{3}, \frac{1}{3}\right]^{10}$ be the set of differentiating users. We define the set Π of problem instances to consist of three-tuples of ratings, one from a differentiating agent and two from indifferent agents ($\Pi = V_d \times V_i \times V_i$). We define the distribution Γ to be uniform over Π.

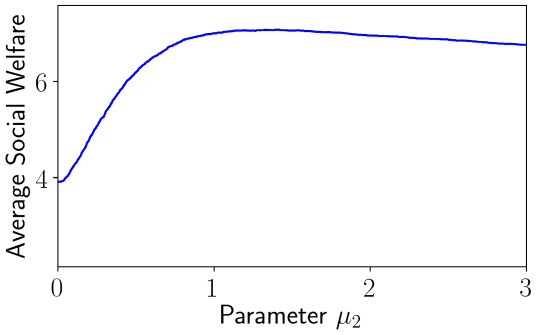
Experimental results. In Figures 10a, 10b, and 10c, we draw a set of samples and illustrate average social welfare over the samples as a function of the parameters. As a reminder, if $\mu_i = 0$, then agent i is a sink agent (Definition 4.5 in Section 4.2). In Figure 10a, agent 1 is the sink agent ($\mu_1 = 0$), in Figure 10b, agent 2 is the sink agent ($\mu_2 = 0$), and in Figure 10c, agent 3 is the sink agent ($\mu_3 = 0$). Without loss of generality, we may assume that one of the agents' weights is fixed as 1. In Figure 10a, we set $\mu_2 = 1$ and in Figures 10b and 10c, we set $\mu_1 = 1$. We draw 5000 samples from Γ and plot average social welfare for varying parameter settings. Setting $\mu_1 = 1$, $\mu_2 = 0$, and $\mu_3 = 1.25$ achieves an average social welfare of 7.03, whereas the mechanism by Faltings and Nguyen [2005], which chooses the agents weights $\boldsymbol{\mu}$ uniformly at random among $\{(0, 1, 1), (1, 0, 1), (1, 1, 0)\}$, achieves an average social welfare of 5.991. Our mechanism therefore



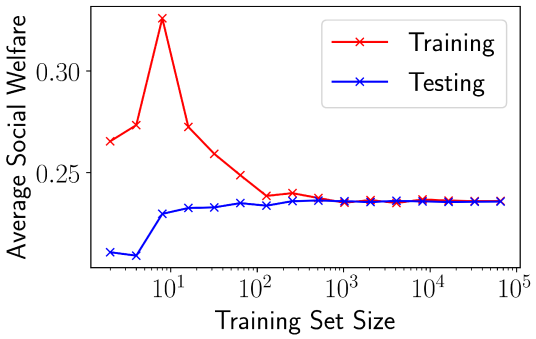
(a) Average social welfare as a function of μ_3 when $\mu_1 = 0$ and $\mu_2 = 1$.



(b) Average social welfare as a function of μ_3 when $\mu_1 = 1$ and $\mu_2 = 0$.



(c) Average social welfare as a function of μ_2 when $\mu_1 = 1$ and $\mu_3 = 0$.



(d) Average social welfare over the training and test sets as a function of the training set size. We calculate social welfare using the parameters with the highest average social welfare over the training set. We average over ten training-test set pairs.

Figure 10: Plots illustrating our experimental results for tuning neutral affine maximizer parameters.

achieves a 17.296% improvement. Intuitively, it makes sense that the empirically optimal NAM has agent 2 as the sink agent (the agent whose bid is ignored and who receives the other agents' payments), since agent 2 is identical to agent 3 in expectation. In other words, agent 2's bid is a noisy signal of agent 3's bid. For this reason, it also intuitively makes sense that the empirically optimal NAM emphasizes the bid of agent 3, since agent 2's bid is ignored.

In Figure 10d, we demonstrate that average social welfare over the training set converges to average social welfare over the test set as the training set grows. To do so, we repeat the following procedure ten times: For each power of 2 up to 2^{16}, we draw a training set from the distribution Γ of that size. We calculate the NAM parameters $\hat{\boldsymbol{\mu}}$ with the highest average social welfare over the training set. We then draw a test set of size 500,000 from the distribution Γ. We calculate the average social welfare of $\hat{\boldsymbol{\mu}}$ over the test set. In Figure 10d, we plot the averages over all ten rounds. The difference between the two curves converges to zero.

These experiments demonstrate that tuning NAM parameters can be extremely beneficial, allowing for significant social welfare gains compared to existing mechanisms [Faltings and Nguyen,

2005]. Moreover, they illustrate the importance of selecting a sufficiently large training set. As Figure 10d demonstrates, a NAM with high social welfare on the training set may have low social welfare on the test set; the training set is too small to guarantee generalization. As the training set grows, average social welfare over the training set converges to average social welfare over the test set.

5 Conclusion

We provided a general sample complexity theorem for learning high-performing algorithm parameters. Our bound applies to any parameterized algorithm for which the performance as a function of its parameters is piecewise structured: for any fixed problem instance, boundary functions partition the parameters into regions where the algorithm's performance is a well-structured function. Our sample complexity bound grows slowly with the intrinsic complexity of both the boundary functions and the well-structured functions. We proved this guarantee by exploiting intricate connections between primal function classes (measuring a parameterized algorithm's performance as a function of its input) with dual function classes (measuring an algorithm's performance on a fixed input as a function of its parameters). We demonstrated that a diverse array of algorithm configuration problems exhibit this structure, and thus our main theorem implies strong sample complexity guarantees for a broad array of algorithms and application domains. This applies both to optimizing an algorithms run time (as we exemplified in integer linear programming and constraint satisfaction applications) and to optimizing an algorithms solution quality (as we exemplified in computational biology, voting, pricing, auction, clustering, greedy algorithm, and integer nonlinear programming applications).

Acknowledgments

This research is funded in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative (GBMF4554 to C.K.), the US National Institutes of Health (R01GM122935 to C.K.), the US National Science Foundation (a Graduate Research Fellowship to E.V., and grants IIS-1901403 to M.B. and T.S., IIS-1618714, and CCF-1535967 to M.B., IIS-1718457, IIS-1617590, and CCF-1733556 to T.S.), the US Army Research Office (W911NF-17-1-0082 to T.S.), an Amazon Research Award to M.B., a Microsoft Research Faculty Fellowship to M.B., a Bloomberg Data Science research grant to M.B., and by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program.

References

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

Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

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

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. *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, Tuomas Sandholm, and Ellen Vitercik. A general theory of sample complexity for multi-item profit maximization. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2018c. Extended abstract. Full version available on arXiv with the same title.

R. C. Buck. Partition of space. *Amer. Math. Monthly*, 50:541–544, 1943. ISSN 0002-9890.

Peter Cramton, Robert Gibbons, and Paul Klemperer. Dissolving a partnership efficiently. *Econometrica*, pages 615–632, 1987.

Dan DeBlasio and John D Kececioglu. *Parameter Advising for Multiple Sequence Alignment*. Springer, 2018.

Boi Faltings and Quang Huy Nguyen. Multi-agent coordination using local search. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh, UK, 2005.

Uriel Feige and Michael Langberg. The RPR^2 rounding technique for semidefinite programs. *Journal of Algorithms*, 60(1):1–23, 2006.

Da-Fei Feng and Russell F Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, 1987.

David Fernández-Baca, Timo Seppäläinen, and Giora Slutzki. Parametric multiple sequence alignment and phylogeny construction. *Journal of Discrete Algorithms*, 2(2):271–287, 2004.

Darya Filippova, Rob Patro, Geet Duggal, and Carl Kingsford. Identification of alternative topological domains in chromatin. *Algorithms for Molecular Biology*, 9:14, May 2014.

Andrew Gilpin and Tuomas Sandholm. Information-theoretic approaches to branching in search. *Discrete Optimization*, 8(2):147–159, 2011. Early version in IJCAI-07.

Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705 – 708, 1982. ISSN 0022-2836.

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

D Gusfield, K Balasubramanian, and D Naor. Parametric optimization of sequence alignment. *Algorithmica*, 12(4-5):312–326, 1994.

Dan Gusfield and Paul Stelling. Parametric and inverse-parametric sequence alignment with xparal. In *Methods in enzymology*, volume 266, pages 481–494. Elsevier, 1996.

Desmond G Higgins and Paul M Sharp. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1):237–244, 1988.

Robert W. Holley, Jean Apgar, George A. Everett, James T. Madison, Mark Marquisee, Susan H. Merrill, John Robert Penswick, and Ada Zamir. Structure of a ribonucleic acid. *Science*, 147 (3664):1462–1465, 1965.

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, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*, pages 507–523, 2011.

John D Kececioglu and Dean Starrett. Aligning alignments exactly. In *Proceedings of the Annual International Conference on Computational Molecular Biology, RECOMB*, volume 8, pages 85–96, 2004.

Eagu Kim and John Kececioglu. Inverse sequence alignment from partial examples. *Proceedings of the International Workshop on Algorithms in Bioinformatics*, pages 359–370, 2007.

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

Erez Lieberman-Aiden, Nynke L. van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R. Lajoie, Peter J. Sabo, Michael O. Dorschner, Richard Sandstrom, Bradley Bernstein, M. A. Bender, Mark Groudine, Andreas Gnirke, John Stamatoyannopoulos, Leonid A. Mirny, Eric S. Lander, and Job Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326(5950):289–293, 2009. ISSN 0036-8075. doi: 10.1126/science.1181369.

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

Darío G Lupiáñez, Malte Spielmann, and Stefan Mundlos. Breaking TADs: how alterations of chromatin domains result in disease. *Trends in Genetics*, 32(4):225–237, 2016.

William H. Majoros and Steven L. Salzberg. An empirical analysis of training protocols for probabilistic gene finders. *BMC Bioinformatics*, 5:206, December 2004.

Damon H. May, Kaipo Tamura, and William S. Noble. `Param-Medic`: A tool for improving ms/ms database search yield by optimizing parameter settings. *Journal of Proteome Research*, 16(4): 1817–1824, 2017.

R Preston McAfee. Amicable divorce: Dissolving a partnership with simple mechanisms. *Journal of Economic Theory*, 56(2):266–293, 1992.

Debasis Mishra and Arunava Sen. Roberts' theorem with neutrality: A social welfare ordering approach. *Games and Economic Behavior*, 75(1):283–298, 2012.

Eugene W. Myers and Webb Miller. Optimal alignments in linear space. *Bioinformatics*, 4(1): 11–17, 03 1988. ISSN 1367-4803.

Roger Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

Swaprava Nath and Tuomas Sandholm. Efficiency and budget balance in general quasi-linear domains. *Games and Economic Behavior*, 113:673 – 693, 2019.

Saket Navlakha, James White, Niranjan Nagarajan, Mihai Pop, and Carl Kingsford. Finding biologically accurate clusterings in hierarchical tree decompositions using the variation of information. In *Annual International Conference on Research in Computational Molecular Biology*, volume 5541, pages 400–417. Springer, 2009.

Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*. Cambridge University press, 2007.

Ruth Nussinov and Ann B Jacobson. Fast algorithm for predicting the secondary structure of single-stranded rna. *Proceedings of the National Academy of Sciences*, 77(11):6309–6313, 1980.

Lior Pachter and Bernd Sturmfels. Parametric inference for biological sequence analysis. *Proceedings of the National Academy of Sciences*, 101(46):16138–16143, 2004a. doi: 10.1073/pnas.0406011101.

Lior Pachter and Bernd Sturmfels. Tropical geometry of statistical models. *Proceedings of the National Academy of Sciences*, 101(46):16132–16137, 2004b. doi: 10.1073/pnas.0406010101.

László Patthy. Detecting homology of distantly related proteins with consensus sequences. *Journal of molecular biology*, 198(4):567–577, 1987.

David Pollard. *Convergence of Stochastic Processes*. Springer, 1984.

Kevin Roberts. The characterization of implementable social choice rules. In J-J Laffont, editor, *Aggregation and Revelation of Preferences*. North-Holland Publishing Company, 1979.

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.

David Sankoff and Robert J Cedergren. Simultaneous comparison of three or more sequences related by a tree. *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison/edited by David Sankoff and Joseph B. Krustal*, 1983.

J. Michael Sauder, Jonathan W. Arthur, and Roland L. Dunbrack Jr. Large-scale comparison of protein sequence alignment algorithms with structure alignments. *Proteins: Structure, Function, and Bioinformatics*, 40(1):6–22, 2000.

Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13 (1):145–147, 1972.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981. ISSN 0022-2836.

Timo Tossavainen. On the zeros of finite sums of exponential functions. *Australian Mathematical Society Gazette*, 33(1):47–50, 2006.

Vladimir Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

Lusheng Wang and Tao Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

Travis J. Wheeler and John D. Kececioglu. Multiple alignment by aligning alignments. *Bioinformatics*, 23(13):i559–i568, 07 2007.

L. Xu, F. Hutter, H.H. Hoos, and K. Leyton-Brown. Satzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32(1):565–606, 2008.