How Much Data Is Sufficient to Learn High-Performing Algorithms? Generalization Guarantees for Data-Driven Algorithm Design

Maria-Florina Balcan
ninamf@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Dan DeBlasio
dfdeblasio@utep.edu
University of Texas at El Paso
El Paso, Texas, USA

Travis Dick
tbd@seas.upenn.edu
University of Pennsylvania
Philadelphia, Pennsylvania, USA

Carl Kingsford
carlk@cs.cmu.edu
Carnegie Mellon University
Ocean Genomics, Inc.
Pittsburgh, Pennsylvania, USA

Tuomas Sandholm
sandholm@cs.cmu.edu
Carnegie Mellon University
Strategic Machine, Inc.
Strategy Robot, Inc.
Optimized Markets, Inc.
Pittsburgh, Pennsylvania, USA

Ellen Vitercik
vitercik@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

ABSTRACT

Algorithms often have tunable parameters that impact performance metrics such as runtime and solution quality. For many algorithms used in practice, no parameter settings admit meaningful worst-case bounds, so the parameters are made available for the user to tune. Alternatively, parameters may be tuned implicitly within the proof of a worst-case guarantee. Worst-case instances, however, may be rare or nonexistent in practice. A growing body of research has demonstrated that *data-driven algorithm design* can lead to significant improvements in performance. This approach uses a *training set* of problem instances sampled from an unknown, application-specific distribution and returns a parameter setting with strong average performance on the training set.

We provide a broadly applicable theory for deriving *generalization guarantees* that bound the difference between the algorithm's average performance over the training set and its expected performance on the unknown distribution. Our results apply no matter how the parameters are tuned, be it via an automated or manual approach. The challenge is that for many types of algorithms, performance is a volatile function of the parameters: slightly perturbing the parameters can cause a large change in behavior. Prior research [e.g., 8, 10, 12, 38] has proved generalization bounds by employing case-by-case analyses of greedy algorithms, clustering algorithms, integer programming algorithms, and selling mechanisms. We uncover a unifying structure which we use to prove extremely general guarantees, yet we recover the bounds from prior research. Our guarantees, which are tight up to logarithmic factors in the worst case, apply whenever an algorithm's performance is a piecewise-constant, -linear, or—more generally—*piecewise-structured* function of its parameters. Our theory also implies novel bounds for voting mechanisms and dynamic programming algorithms from computational biology.

CCS CONCEPTS

• **Theory of computation → Sample complexity and generalization bounds**.

KEYWORDS

Automated algorithm design, data-driven algorithm design, automated algorithm configuration, machine learning theory, computational biology, mechanism design

ACM Reference Format:
Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. 2021. How Much Data Is Sufficient to Learn High-Performing Algorithms? Generalization Guarantees for Data-Driven Algorithm Design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3406325.3451036

1 INTRODUCTION

Algorithms often have tunable parameters that impact performance metrics such as runtime, solution quality, and memory usage. These parameters may be set explicitly, as is often the case in applied disciplines. For example, integer programming solvers expose over one hundred parameters for the user to tune. There may not be parameter settings that admit meaningful worst-case bounds, but after careful parameter tuning, these algorithms can quickly find solutions to computationally challenging problems. However, applied approaches to parameter tuning have rarely come with provable guarantees. Alternatively, an algorithm's parameters may be set implicitly, as is often the case in theoretical computer science: a proof may implicitly optimize over a parameterized family of algorithms in order to guarantee a worst-case approximation factor or runtime bound. Worst-case bounds, however, can be overly pessimistic in

STOC '21, June 21–25, 2021, Virtual, Italy

ACM ISBN 978-1-4503-8053-9/21/06.
https://doi.org/10.1145/3406325.3451036

practice. A growing body of research (surveyed by Balcan [5]) has demonstrated the power of *data-driven algorithm design*, where machine learning is used to find parameter settings that work particularly well on problems from the application domain at hand.

We present a broadly applicable theory for proving *generalization guarantees* in the context of data-driven algorithm design. We adopt a natural learning-theoretic model of data-driven algorithm design introduced by Gupta and Roughgarden [38]. As in the applied literature on automated algorithm configuration [e.g., 41, 43, 45, 48, 66, 76, 77], we assume there is an unknown, application-specific distribution over the algorithm's input instances. A learning procedure receives a *training set* sampled from this distribution and returns a parameter setting—or *configuration*—with strong average performance over the training set. If the training set is too small, this configuration may have poor expected performance. *Generalization guarantees* bound the difference between average performance over the training set and expected performance. Our guarantees apply no matter how the parameters are optimized, via an algorithmic search as in *automated algorithm configuration* [e.g., 22, 66, 76, 77], or manually as in *experimental algorithmics* [e.g., 16, 44, 54].

Across many types of algorithms—for example, combinatorial algorithms, integer programs, and dynamic programs—the algorithm's performance is a volatile function of its parameters. This is a key challenge that distinguishes our results from prior research on generalization guarantees. For well-understood functions in machine learning theory such as linear separators or other smooth curves in Euclidean spaces, there is generally a simple connection between a function's parameters and the value of the function. Meanwhile, slightly perturbing an algorithm's parameters can cause significant changes in its behavior and performance. To provide generalization bounds, we uncover structure that governs these volatile performance functions.

The structure we discover depends on the relationship between *primal* and *dual* functions [2]. To derive generalization bounds, a common strategy is to calculate the *intrinsic complexity* of a function class $\mathcal{U}$ which we refer to as the *primal class*. Every function $u_{\boldsymbol{\rho}} \in \mathcal{U}$ is defined by a parameter setting $\boldsymbol{\rho} \in \mathbb{R}^d$ and $u_{\boldsymbol{\rho}}(x) \in \mathbb{R}$ measures the performance of the algorithm parameterized by $\boldsymbol{\rho}$ given the input x. We measure intrinsic complexity using the classic notion of *pseudo-dimension* [63]. This is a challenging task because the domain of every function in $\mathcal{U}$ is a set of problem instances, so there are no obvious notions of Lipschitz continuity or smoothness on which we can rely. Instead, we use structure exhibited by the *dual class* $\mathcal{U}^*$. Every *dual function* $u_x^* \in \mathcal{U}^*$ is defined by a problem instance x and measures the algorithm's performance as a function of its parameters given x as input. The dual functions have a simple, Euclidean domain $\mathbb{R}^d$ and we demonstrate that they have ample structure which we can use to bound the pseudo-dimension of $\mathcal{U}$.

1.1 Our contributions

Our results apply to any parameterized algorithm with dual functions that exhibit a clear-cut, ubiquitous structural property: the duals are piecewise constant, piecewise linear, or—more broadly—piecewise *structured*. The parameter space decomposes into a small number of regions such that within each region, the algorithm's performance is "well behaved." As an example, Figure 1 illustrates

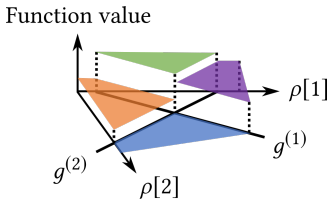


Figure 1: A piecewise-constant function over $\mathbb{R}^2_{\geq 0}$ with linear boundary functions $g^{(1)}$ and $g^{(2)}$.

a piecewise-structured function of two parameters $\boldsymbol{\rho} = (\rho[1], \rho[2])$. There are two functions $g^{(1)}$ and $g^{(2)}$ that define a partition of the parameter space and four constant functions that define the function value on each subset from this partition.

More formally, the dual class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-*piecewise decomposable* if for every problem instance, there are at most k boundary functions from a set $\mathcal{G}$ (for example, the set of linear separators) that partition the parameter space into regions such that within each region, algorithmic performance is defined by a function from a set $\mathcal{F}$ (for example, the set of constant functions). We bound the pseudo-dimension of $\mathcal{U}$ in terms of the pseudo- and VC-dimensions of the dual classes $\mathcal{F}^*$ and $\mathcal{G}^*$, denoted $\text{Pdim}(\mathcal{F}^*)$ and $\text{VCdim}(\mathcal{G}^*)$. This yields our main theorem: if $[0, H]$ is the range of the functions in $\mathcal{U}$, then with probability $1 - \delta$ over the draw of N training instances, for any parameter setting, the difference between the algorithm's average performance over the training set and its expected performance is $\tilde{O}\left(H\sqrt{\frac{1}{N}\left(\text{Pdim}(\mathcal{F}^*) + \text{VCdim}(\mathcal{G}^*)\ln k + \ln\frac{1}{\delta}\right)}\right)$. Specifically, we prove that $\text{Pdim}(\mathcal{U}) = \tilde{O}(\text{Pdim}(\mathcal{F}^*) + \text{VCdim}(\mathcal{G}^*)\ln k)$ and that this bound is tight up to log factors. The classes $\mathcal{F}$ and $\mathcal{G}$ are often so well structured that bounding $\text{Pdim}(\mathcal{F}^*)$ and $\text{VCdim}(\mathcal{G}^*)$ is straightforward.

This is the most broadly applicable generalization bound for data-driven algorithm design in the distributional learning model that applies to arbitrary input distributions. A nascent line of research [3, 8–10, 12, 38] provides generalization bounds for a selection of parameterized algorithms. Unlike the results in this paper, those papers analyze each algorithm individually, case by case. Our approach recovers those bounds, implying guarantees for configuring greedy algorithms [38], clustering algorithms [10], and integer programming algorithms [8, 10], as well as mechanism design for revenue maximization [12]. We also derive novel bounds for computational biology algorithms and voting mechanisms.

Proof insights. At a high level, we prove this guarantee by counting the number of parameter settings with significantly different performance over any set $\mathcal{S}$ of problem instances. To do so, we first count the number of regions induced by the $|\mathcal{S}|k$ boundary functions that correspond to these problem instances. This step subtly depends not on the VC-dimension of the class of boundary functions $\mathcal{G}$, but rather on $\text{VCdim}(\mathcal{G}^*)$. These $|\mathcal{S}|k$ boundary functions partition the parameter space into regions where across all instances x in $\mathcal{S}$, the dual functions u_x^* are simultaneously structured. Within any one region, we use the pseudo-dimension of the dual class $\mathcal{F}^*$ to count the number of parameter settings in that

region with significantly different performance. We aggregate these bounds over all regions to bound the pseudo-dimension of $\mathcal{U}$.

Parameterized dynamic programming algorithms from computational biology. Our results imply bounds for a variety of computational biology algorithms that are used in practice. We analyze parameterized sequence alignment algorithms [29, 35, 39, 61, 62] as well as RNA folding algorithms [60], which predict how an input RNA strand would naturally fold, offering insight into the molecule's function. We also provide guarantees for algorithms that predict topologically associating domains in DNA sequences [30], which shed light on how DNA wraps into three-dimensional structures that influence genome function.

Parameterized voting mechanisms. A *mechanism* is a special type of algorithm designed to help a set of agents come to a collective decision. For example, a town's residents may want to build a public resource such as a park, pool, or skating rink, and a mechanism would help them decide which to build. We analyze *neutral affine maximizers* [55, 59, 65], a well-studied family of parameterized mechanisms. The parameters impact social welfare, which is the sum of the agents' values for the mechanism's outcome.

1.2 Additional related research

A growing body of theoretical research investigates how machine learning can be incorporated in the process of algorithm design [1, 3, 4, 8–10, 12, 14, 15, 17, 19, 26, 27, 31, 38, 42, 46, 47, 53, 56, 64, 73–75]. A chapter by Balcan [5] provides a comprehensive survey. We highlight a few papers that are most related to ours below.

Runtime optimization with provable guarantees. Kleinberg et al. [46, 47] and Weisz et al. [74, 75] provide configuration procedures with provable guarantees when the goal is to minimize runtime. In contrast, our bounds apply to arbitrary performance metrics, such as solution quality as well as runtime. Also, their procedures are designed for the case where the set of parameter settings is finite (although they can still offer some guarantees when the parameter space is infinite by first sampling a finite set of parameter settings and then running the configuration procedure; Balcan et al. [8, 13] study what kinds of guarantees discretization approaches can and cannot provide). In contrast, our guarantees apply immediately to infinite parameter spaces. Finally, unlike our results, the guarantees from this prior research are not configuration-procedure-agnostic: they apply only to the specific procedures that are proposed.

Learning-augmented algorithms. A related line of research has designed algorithms that replace some steps of a classic worst-case algorithm with a machine-learned oracle that makes predictions about structural aspects of the input [26, 27, 42, 53, 56, 64, 73]. If the prediction is accurate, the algorithm's performance (for example, its error or runtime) is superior to the original worst-case algorithm, and if the prediction is incorrect, the algorithm performs as well as that worst-case algorithm. Though similar, our approach to data-driven algorithm design is different because we are not attempting to learn structural aspects of the input; rather, we are optimizing the algorithm's parameters directly using the training set. Moreover, we can also compete with the best-known worst-case algorithm by including it in the algorithm class over which we optimize. Just adding one extra algorithm—however different—does not increase our sample complexity bounds. That best-in-the-worst-case algorithm does not have to be a special case of the parameterized algorithm.

Dispersion. Balcan et al. [4, 9] provide provable guarantees for algorithm configuration, with a particular focus on online learning and privacy-preserving algorithm configuration. These tasks are impossible in the worst case, so these papers identify a property of the dual functions under which online and private configuration are possible. This condition is *dispersion*, which, roughly speaking, requires that the discontinuities of the dual functions are not too concentrated in any ball. Online learning guarantees imply sample complexity guarantees due to online-to-batch conversion, and Balcan et al. [9] also provide sample complexity guarantees based on dispersion using Rademacher complexity.

Proofs that dispersion holds typically follow by exploiting properties of the input distribution under certain assumptions or—when applicable—by appealing to the random nature of the parameterized algorithm. Thus, for arbitrary distributions and deterministic algorithms, dispersion does not necessarily hold. In contrast, our results hold even when the discontinuities concentrate, and thus are applicable to a broader set of problems in the distributional learning model. In other words, the results from this paper cannot be recovered using the techniques of Balcan et al. [4, 9].

2 NOTATION AND PROBLEM STATEMENT

We study algorithms parameterized by a set $\mathcal{P} \subseteq \mathbb{R}^d$. As a concrete example, parameterized algorithms are often used for sequence alignment [35]. There are many features of an alignment one might wish to optimize, such as the number of matches, mismatches, or indels (defined in Section 4.1). A parameterized objective function is defined by weighting these features. As another example, hierarchical clustering algorithms often use linkage routines such as single, complete, and average linkage. Parameters can be used to interpolate between these three classic procedures [10], which can be outperformed with a careful parameter tuning [3].

We use $\mathcal{X}$ to denote the set of problem instances the algorithm takes as input. We measure the performance of the algorithm parameterized by $\boldsymbol{\rho} = (\rho[1], \ldots, \rho[d]) \in \mathbb{R}^d$ via a utility function $u_{\boldsymbol{\rho}} : \mathcal{X} \to [0, H]$, with $\mathcal{U} = \left\{u_{\boldsymbol{\rho}} : \boldsymbol{\rho} \in \mathcal{P}\right\}$ denoting the set of all such functions. We assume there is an unknown, application-specific distribution $\mathcal{D}$ over $\mathcal{X}$.

Our goal is to find a parameter vector in $\mathcal{P}$ with high performance in expectation over $\mathcal{D}$. We provide *generalization guarantees* for this problem. Given a training set of problem instances $\mathcal{S}$ sampled from $\mathcal{D}$, a generalization guarantee bounds the difference—for any choice of the parameters $\boldsymbol{\rho}$—between the average performance of the algorithm over $\mathcal{S}$ and its expected performance.

Specifically, our main technical contribution is a bound on the *pseudo-dimension* [63] of the set $\mathcal{U}$. For any arbitrary set of functions $\mathcal{H}$ that map an abstract domain $\mathcal{Y}$ to $\mathbb{R}$, the *pseudo-dimension of* $\mathcal{H}$, denoted $\text{Pdim}(\mathcal{H})$, is the size of the largest set $\{y_1, \ldots, y_N\} \subseteq \mathcal{Y}$ such that for some set of *targets* $z_1, \ldots, z_N \in \mathbb{R}$,

$$\left|\left\{\left(\text{sign}\left(h\left(y_1\right) - z_1\right), \ldots, \text{sign}\left(h\left(y_N\right) - z_N\right)\right) \mid h \in \mathcal{H}\right\}\right| = 2^N. \quad (1)$$

Classic results from learning theory [63] translate pseudo-dimension bounds into generalization guarantees. For example, suppose $[0, H]$

is the range of the functions in $\mathcal{H}$. For any $\delta \in (0,1)$ and any distribution $\mathcal{D}$ over $\mathcal{Y}$, with probability $1-\delta$ over the draw of $\mathcal{S} \sim \mathcal{D}^N$, for all functions $h \in \mathcal{H}$, the difference between the average value of h over $\mathcal{S}$ and its expected value is bounded as follows:

$$\left| \frac{1}{N} \sum_{y \in \mathcal{S}} h(y) - \mathop{\mathbb{E}}_{y \sim \mathcal{D}} [h(y)] \right| = O\left(H \sqrt{\frac{1}{N} \left(\text{Pdim}(\mathcal{H}) + \ln \frac{1}{\delta} \right)} \right). \quad (2)$$

When $\mathcal{H}$ is a set of binary-valued functions that map $\mathcal{Y}$ to $\{0,1\}$, the pseudo-dimension of $\mathcal{H}$ is more commonly referred to as the *VC-dimension of* $\mathcal{H}$ [70], denoted $\text{VCdim}(\mathcal{H})$.

3 GENERALIZATION GUARANTEES FOR DATA-DRIVEN ALGORITHM DESIGN

In data-driven algorithm design, there are two closely related function classes. First, for each parameter setting $\boldsymbol{\rho} \in \mathcal{P}$, $u_{\boldsymbol{\rho}} : \mathcal{X} \to \mathbb{R}$ measures performance as a function of the input $x \in \mathcal{X}$. Similarly, for each input x, there is a function $u_x : \mathcal{P} \to \mathbb{R}$ defined as $u_x(\boldsymbol{\rho}) = u_{\boldsymbol{\rho}}(x)$ that measures performance as a function of the parameter vector $\boldsymbol{\rho}$. The set $\{u_x \mid x \in \mathcal{X}\}$ is equivalent to Assouad's notion of the *dual class* [2].

Definition 3.1 (Dual class [2]). 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 $\{u_x \mid x \in \mathcal{X}\}$ is equivalent to the dual class $\mathcal{U}^* = \left\{ u_x^* : \mathcal{U} \to [0,H] \mid x \in \mathcal{X} \right\}$ in the sense that for every parameter vector $\boldsymbol{\rho} \in \mathcal{P}$ and every instance $x \in \mathcal{X}$, $u_x(\boldsymbol{\rho}) = u_x^*(u_{\boldsymbol{\rho}})$.

Many combinatorial algorithms share a clear-cut, useful structure: for each instance $x \in \mathcal{X}$, the function u_x is *piecewise structured.* For example, each function u_x might be piecewise constant with a small number of pieces. Given the equivalence of the functions $\{u_x \mid x \in \mathcal{X}\}$ and the dual class $\mathcal{U}^*$, the dual class exhibits this piecewise structure as well. We use this structure to bound the pseudo-dimension of the primal class $\mathcal{U}$.

Intuitively, a function $h : \mathcal{Y} \to \mathbb{R}$ is piecewise structured if we can partition the domain $\mathcal{Y}$ into subsets $\mathcal{Y}_1, \ldots, \mathcal{Y}_M$ so that when we restrict h to one piece $\mathcal{Y}_i$, h equals some well-structured function $f : \mathcal{Y} \to \mathbb{R}$. In other words, for all $y \in \mathcal{Y}_i$, $h(y) = f(y)$. We define the partition $\mathcal{Y}_1, \ldots, \mathcal{Y}_M$ using *boundary functions* $g^{(1)}, \ldots, g^{(k)} : \mathcal{Y} \to \{0,1\}$. Each function $g^{(i)}$ divides the domain $\mathcal{Y}$ into two sets: the points it labels 0 and the points it labels 1. Figure 2 illustrates a partition of $\mathbb{R}^2$ by boundary functions. Together, the k boundary functions partition the domain $\mathcal{Y}$ into at most 2^k regions, each one corresponding to a bit vector $\boldsymbol{b} \in \{0,1\}^k$ that describes on which side of each boundary the region belongs. For each region, we specify a *piece function* $f_{\boldsymbol{b}} : \mathcal{Y} \to \mathbb{R}$ that defines the function values of h restricted to that region. Figure 1 shows an example of a piecewise-structured function with two boundary functions and four piece functions.

For many algorithms, every function in the dual class is piecewise structured. Moreover, across dual functions, the 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 captures this structure.

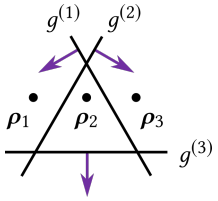


Figure 2: Boundary functions partitioning $\mathbb{R}^2$. The arrows indicate on which side of each function $g^{(i)}(\boldsymbol{\rho}) = 0$ and on which side $g^{(i)}(\boldsymbol{\rho}) = 1$. For example, $g^{(1)}(\boldsymbol{\rho}_1) = 1$, $g^{(1)}(\boldsymbol{\rho}_2) = 1$, and $g^{(1)}(\boldsymbol{\rho}_3) = 0$.

Definition 3.2. A function class $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{Y}}$ that maps a domain $\mathcal{Y}$ to $\mathbb{R}$ is $(\mathcal{F}, \mathcal{G}, k)$*-piecewise decomposable* for a class $\mathcal{G} \subseteq \{0,1\}^{\mathcal{Y}}$ of boundary functions and a class $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{Y}}$ of piece functions if the following holds: for every $h \in \mathcal{H}$, there are 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 $y \in \mathcal{Y}$, $h(y) = f_{\boldsymbol{b}_y}(y)$ where $\boldsymbol{b}_y = \left(g^{(1)}(y), \ldots, g^{(k)}(y) \right) \in \{0,1\}^k$.

Our main theorem shows that when the dual class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable, we can bound the pseudo-dimension of $\mathcal{U}$ in terms of the VC-dimension of $\mathcal{G}^*$ and the pseudo-dimension of $\mathcal{F}^*$. Later, we show that for many common classes $\mathcal{F}$ and $\mathcal{G}$, we can easily bound the complexity of their duals.

Theorem 3.3. *Suppose that the dual function class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable with boundary functions $\mathcal{G} \subseteq \{0,1\}^{\mathcal{U}}$ and piece functions $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{U}}$. The pseudo-dimension of $\mathcal{U}$ is bounded as follows:* $\text{Pdim}(\mathcal{U}) = \tilde{O}\left(\text{Pdim}(\mathcal{F}^*) + \text{VCdim}(\mathcal{G}^*) \ln k\right)$.

To make the theorem's proof succinct, we extract a key insight in the following lemma. Given a set $\mathcal{H}$ of functions that map a domain $\mathcal{Y}$ to $\{0,1\}$, Lemma 3.4 bounds the number of binary vectors

$$(h_1(y), \ldots, h_N(y)) \quad (3)$$

we can obtain for any N functions $h_1, \ldots, h_N \in \mathcal{H}$ as we vary the input $y \in \mathcal{Y}$. Pictorially, if we partition $\mathbb{R}^2$ using the functions $g^{(1)}$, $g^{(2)}$, and $g^{(3)}$ from Figure 2 for example, Lemma 3.4 bounds the number of regions in the partition. This bound depends not on the VC-dimension of the class $\mathcal{H}$, but rather on that of its dual $\mathcal{H}^*$. We use a classic lemma by Sauer [69] to prove Lemma 3.4. Sauer's lemma [69] bounds the number of binary vectors of the form $(h(y_1), \ldots, h(y_N))$ we can obtain for any N elements $y_1, \ldots, y_N \in \mathcal{Y}$ as we vary the function $h \in \mathcal{H}$ by $(eN)^{\text{VCdim}(\mathcal{H})}$. Therefore, it does not immediately imply a bound on the number of vectors from Equation (3). In order to apply Sauer's lemma, we must transition to the dual class.

Lemma 3.4. *Let $\mathcal{H}$ be a set of functions that map a domain $\mathcal{Y}$ to $\{0,1\}$. For any functions $h_1, \ldots, h_N \in \mathcal{H}$, the number of binary*

vectors $(h_1(y), \ldots, h_N(y))$ *obtained by varying the input* $y \in \mathcal{Y}$ *is bounded as follows:*

$$|\{(h_1(y), \ldots, h_N(y)) \mid y \in \mathcal{Y}\}| \leq (eN)^{\mathrm{VCdim}(\mathcal{H}^*)}. \tag{4}$$

Proof. We rewrite the left-hand-side of Equation (4) as

$$\left|\left\{\left(h_y^*(h_1), \ldots, h_y^*(h_N)\right) \middle| y \in \mathcal{Y}\right\}\right|.$$

Since we fix N inputs and vary the function h_y^*, the lemma statement follows from Sauer's lemma [69]. □

We now prove Theorem 3.3.

Proof of Theorem 3.3. Fix an arbitrary set of problem instances $x_1, \ldots, x_N \in \mathcal{X}$ and targets $z_1, \ldots, z_N \in \mathbb{R}$. We bound the number of ways that $\mathcal{U}$ can label the problem instances $x_1, \ldots, x_N$ with respect to the target thresholds $z_1, \ldots, z_N \in \mathbb{R}$. In other words, as per Equation (1), we bound the size of the set

$$\left|\left\{\begin{pmatrix} \mathrm{sign}\left(u_{\boldsymbol{\rho}}(x_1) - z_1\right) \\ \vdots \\ \mathrm{sign}\left(u_{\boldsymbol{\rho}}(x_N) - z_N\right) \end{pmatrix} \middle| \boldsymbol{\rho} \in \mathcal{P}\right\}\right| = \left|\left\{\begin{pmatrix} \mathrm{sign}\left(u_{x_1}^*(u_{\boldsymbol{\rho}}) - z_1\right) \\ \vdots \\ \mathrm{sign}\left(u_{x_N}^*(u_{\boldsymbol{\rho}}) - z_N\right) \end{pmatrix} \middle| \boldsymbol{\rho} \in \mathcal{P}\right\}\right| \tag{5}$$

by $(ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}(eN)^{\mathrm{Pdim}(\mathcal{F}^*)}$. Then solving for the largest N such that $2^N \leq (ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}(eN)^{\mathrm{Pdim}(\mathcal{F}^*)}$ gives a bound on $\mathrm{Pdim}(\mathcal{U})$. Our bound on Equation (5) has two main steps:

(1) In Claim 3.5, we show that there are $M < (ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}$ subsets $\mathcal{P}_1, \ldots, \mathcal{P}_M$ partitioning the parameter space $\mathcal{P}$ such that within any one subset, the dual functions $u_{x_1}^*, \ldots, u_{x_N}^*$ are simultaneously structured. In particular, for each subset $\mathcal{P}_j$, there exist piece functions $f_1, \ldots, f_N \in \mathcal{F}$ such that $u_{x_i}^*(u_{\boldsymbol{\rho}}) = f_i(u_{\boldsymbol{\rho}})$ for all $\boldsymbol{\rho} \in \mathcal{P}_j$ and $i \in [N]$. This is the partition of $\mathcal{P}$ induced by aggregating all of the boundary functions corresponding to the dual functions $u_{x_1}^*, \ldots, u_{x_N}^*$.
(2) We then show that for any region $\mathcal{P}_j$ of the partition, as we vary the parameter vector $\boldsymbol{\rho} \in \mathcal{P}_j$, $u_{\boldsymbol{\rho}}$ can label the problem instances $x_1, \ldots, x_N$ in at most $(eN)^{\mathrm{Pdim}(\mathcal{F}^*)}$ ways with respect to the target thresholds $z_1, \ldots, z_N$. It follows that the total number of ways that $\mathcal{U}$ can label the problem instances $x_1, \ldots, x_N$ is bounded by $(ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}(eN)^{\mathrm{Pdim}(\mathcal{F}^*)}$.

We now prove Claim 3.5.

Claim 3.5. *There are* $M < (ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}$ *subsets* $\mathcal{P}_1, \ldots, \mathcal{P}_M$ *partitioning the parameter space* $\mathcal{P}$ *such that within any one subset, the dual functions* $u_{x_1}^*, \ldots, u_{x_N}^*$ *are simultaneously structured. In particular, for each subset* $\mathcal{P}_j$*, there exist piece functions* $f_1, \ldots, f_N \in \mathcal{F}$ *such that* $u_{x_i}^*(u_{\boldsymbol{\rho}}) = f_i(u_{\boldsymbol{\rho}})$ *for all* $\boldsymbol{\rho} \in \mathcal{P}_j$ *and* $i \in [N]$.

Proof of Claim 3.5. Let $u_{x_1}^*, \ldots, u_{x_N}^* \in \mathcal{U}^*$ be the dual functions defined by the instances $x_1, \ldots, x_N$. Since $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable, we know that for each function $u_{x_i}^*$, there are k boundary functions $g_i^{(1)}, \ldots, g_i^{(k)} \in \mathcal{G} \subseteq \{0,1\}^{\mathcal{U}}$ that define its piecewise decomposition. Let $\hat{\mathcal{G}} = \bigcup_{i=1}^N \left\{g_i^{(1)}, \ldots, g_i^{(k)}\right\}$ be the union of these boundary functions across all $i \in [N]$. For ease of notation, we relabel the functions in $\hat{\mathcal{G}}$, calling them $g_1, \ldots, g_{kN}$. Let M be the total number of kN-dimensional vectors we can obtain by applying the functions in $\hat{\mathcal{G}} \subseteq \{0,1\}^{\mathcal{U}}$ to elements of $\mathcal{U}$:

$$M := \left|\left\{\begin{pmatrix} g_1(u_{\boldsymbol{\rho}}) \\ \vdots \\ g_{kN}(u_{\boldsymbol{\rho}}) \end{pmatrix} : \boldsymbol{\rho} \in \mathcal{P}\right\}\right|. \tag{6}$$

By Lemma 3.4, $M < (ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}$. Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M$ be the binary vectors in the set from Equation (6). For each $i \in [M]$, let $\mathcal{P}_j = \left\{\boldsymbol{\rho} \mid (g_1(u_{\boldsymbol{\rho}}), \ldots, g_{kN}(u_{\boldsymbol{\rho}})) = \boldsymbol{b}_j\right\}$. By construction, for each set $\mathcal{P}_j$, the values of all the boundary functions $g_1(u_{\boldsymbol{\rho}}), \ldots, g_{kN}(u_{\boldsymbol{\rho}})$ are constant as we vary $\boldsymbol{\rho} \in \mathcal{P}_j$. Therefore, there is a fixed set of piece functions $f_1, \ldots, f_N \in \mathcal{F}$ so that $u_{x_i}^*(u_{\boldsymbol{\rho}}) = f_i(u_{\boldsymbol{\rho}})$ for all vectors $\boldsymbol{\rho} \in \mathcal{P}_j$ and indices $i \in [N]$. Therefore, the claim holds. □

Claim 3.5 and Equation (5) imply that for every subset $\mathcal{P}_j$ of the partition,

$$\left|\left\{\begin{pmatrix} \mathrm{sign}\left(u_{\boldsymbol{\rho}}(x_1) - z_1\right) \\ \vdots \\ \mathrm{sign}\left(u_{\boldsymbol{\rho}}(x_N) - z_N\right) \end{pmatrix} \middle| \boldsymbol{\rho} \in \mathcal{P}_j\right\}\right| = \left|\left\{\begin{pmatrix} \mathrm{sign}\left(f_1(u_{\boldsymbol{\rho}}) - z_1\right) \\ \vdots \\ \mathrm{sign}\left(f_N(u_{\boldsymbol{\rho}}) - z_N\right) \end{pmatrix} \middle| \boldsymbol{\rho} \in \mathcal{P}_j\right\}\right|. \tag{7}$$

Lemma 3.4 implies that Equation (7) is bounded by $(eN)^{\mathrm{Pdim}(\mathcal{F}^*)}$. In other words, for any region $\mathcal{P}_j$ of the partition, as we vary the parameter vector $\boldsymbol{\rho} \in \mathcal{P}_j$, $u_{\boldsymbol{\rho}}$ can label the problem instances $x_1, \ldots, x_N$ in at most $(eN)^{\mathrm{Pdim}(\mathcal{F}^*)}$ ways with respect to the target thresholds $z_1, \ldots, z_N$. Because there are $M < (ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}$ regions $\mathcal{P}_j$ of the partition, we can conclude that that $\mathcal{U}$ can label the instances $x_1, \ldots, x_N$ in at most $(ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}(eN)^{\mathrm{Pdim}(\mathcal{F}^*)}$ distinct ways relative to the targets $z_1, \ldots, z_N$. In other words, Equation (5) is bounded by $(ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}(eN)^{\mathrm{Pdim}(\mathcal{F}^*)}$. On the other hand, if $\mathcal{U}$ shatters the problem instances $x_1, \ldots, x_N$, then the number of distinct labelings must be 2^N. Therefore, $\mathrm{Pdim}(\mathcal{U})$ is at most the largest value of N such that $2^N \leq (ekN)^{\mathrm{VCdim}(\mathcal{G}^*)}(eN)^{\mathrm{Pdim}(\mathcal{F}^*)}$, which implies that $N = \tilde{O}\left(\mathrm{Pdim}(\mathcal{F}^*) + \mathrm{VCdim}(\mathcal{G}^*)\ln k\right)$. □

We prove several lower bounds which show that Theorem 3.3 is tight up to logarithmic factors.

Theorem 3.6. *The following lower bounds hold:*

(1) *There is a parameterized sequence alignment algorithm with* $\mathrm{Pdim}(\mathcal{U}) = \Omega(\log n)$ *for some* $n \geq 1$*. Its dual class* $\mathcal{U}^*$ *is* $(\mathcal{F}, \mathcal{G}, n)$*-piecewise decomposable for classes* $\mathcal{F}$ *and* $\mathcal{G}$ *with* $\mathrm{Pdim}(\mathcal{F}^*) = \mathrm{VCdim}(\mathcal{G}^*) = 1$.
(2) *There is a parameterized voting mechanism with* $\mathrm{Pdim}(\mathcal{U}) = \Omega(n)$ *for some* $n \geq 1$*. Its dual class* $\mathcal{U}^*$ *is* $(\mathcal{F}, \mathcal{G}, 2)$*-piecewise decomposable for classes* $\mathcal{F}$ *and* $\mathcal{G}$ *with* $\mathrm{Pdim}(\mathcal{F}^*) = 1$ *and* $\mathrm{VCdim}(\mathcal{G}^*) = n$.

Proof. In Theorem 4.3, we prove the result for sequence alignment, in which case n is the maximum length of the sequences, $\mathcal{F}$ is the set of constant functions, and $\mathcal{G}$ is the set of threshold

functions. In Theorem 5.2, we prove the result for voting mechanisms, in which case n is the number of agents that participate in the mechanism, $\mathcal{F}$ is the set of constant functions, and $\mathcal{G}$ is the set of homogeneous linear separators in $\mathbb{R}^n$. □

Applications to representative function classes

We now instantiate Theorem 3.3 in a general setting inspired by data-driven algorithm design. Let $\mathcal{U} = \{u_\rho \mid \rho \in \mathbb{R}\}$ be a set of utility functions defined over a single-dimensional parameter space. We often find that the dual functions are piecewise constant, linear, or polynomial. More generally, the dual functions are piecewise structured with piece functions that oscillate a fixed number of times. In other words, the dual class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable where the boundary functions $\mathcal{G}$ are thresholds and the piece functions $\mathcal{F}$ oscillate a bounded number of times, as formalized below.

Definition 3.7. 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.

Figure 3 illustrates three common types of functions with bounded oscillations. In the following lemma, we prove that if $\mathcal{H}$ is a class of functions that map $\mathbb{R}$ to $\mathbb{R}$, each of which has at most B oscillations, then $\text{Pdim}(\mathcal{H}^*) = O(\ln B)$.

Lemma 3.8. *Let $\mathcal{H}$ be a class of functions mapping $\mathbb{R}$ to $\mathbb{R}$, each of which has at most B oscillations. Then* $\text{Pdim}(\mathcal{H}^*) = O(\ln B)$.

Proof. Suppose that $\text{Pdim}(\mathcal{H}^*) = N$. Then there exist functions $h_1, \ldots, h_N \in \mathcal{H}$ and witnesses $z_1, \ldots, z_N \in \mathbb{R}$ such that for every subset $T \subseteq [N]$, there exists a parameter setting $\rho \in \mathbb{R}$ 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 $T \subseteq [N]$, there exists a parameter setting $\rho \in \mathbb{R}$ such that $h_i(\rho) \geq z_i$ if and only if $i \in T$. Let $\mathcal{P}^*$ be the set of 2^N parameter settings corresponding to each subset $T \subseteq [N]$. By definition, these parameter settings induce 2^N distinct binary vectors as follows:

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

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

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

which is a contradiction. Thus, $2^N \leq BN + 1$, so $N = O(\ln B)$. □

Lemma 3.8 implies the following pseudo-dimension bound when the dual function class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-piecewise decomposable, where the boundary functions $\mathcal{G}$ are thresholds and the piece functions $\mathcal{F}$ oscillate a bounded number of times.

Lemma 3.9. *Let $\mathcal{U} = \{u_\rho \mid \rho \in \mathbb{R}\}$ be a set of utility functions and suppose the dual class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, k)$-decomposable, where the boundary functions $\mathcal{G} = \{g_a \mid a \in \mathbb{R}\}$ are thresholds $g_a : u_\rho \mapsto \mathbb{I}_{\{a \leq \rho\}}$. Suppose for each $f \in \mathcal{F}$, the function $\rho \mapsto f(u_\rho)$ has at most B oscillations. Then* $\text{Pdim}(\mathcal{U}) = O((\ln B)\ln(k \ln B))$.

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 be a parameter setting $\rho \in \mathbb{R}$ where $g_{u_\rho}^*(g_a) = g_a(u_\rho) = \mathbb{I}_{\{a \leq \rho\}} = 0$ and $g_{u_\rho}^*(g_b) = g_b(u_\rho) = \mathbb{I}_{\{b \leq \rho\}} = 1$. Therefore, $b \leq \rho < a$, which is a contradiction, so $\text{VCdim}(\mathcal{G}^*) = 1$.

Next, we claim that $\text{Pdim}(\mathcal{F}^*) = O(\ln B)$. 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 $N = \text{Pdim}(\mathcal{H}^*)$. By Lemma 3.8, we know that $N = O(\ln B)$. We claim that $\text{Pdim}(\mathcal{H}^*) \geq \text{Pdim}(\mathcal{F}^*)$. For a contradiction, suppose the class $\mathcal{F}^*$ can shatter $N + 1$ functions $f_1, \ldots, f_{N+1}$ using witnesses $z_1, \ldots, z_{N+1} \in \mathbb{R}$. By definition, this means that

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

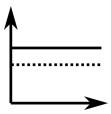
For any function $f \in \mathcal{F}$ and any parameter setting $\rho \in \mathbb{R}$, $f_{u_\rho}^*(f) = f(u_\rho) = h_f(\rho) = h_\rho^*(h_f)$. Therefore,

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

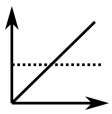
which contradicts the fact that $\text{Pdim}(\mathcal{H}^*) = N$. Thus, $\text{Pdim}(\mathcal{F}^*) \leq N = O(\ln B)$. The corollary then follows from Theorem 3.3. □

4 PARAMETERIZED COMPUTATIONAL BIOLOGY ALGORITHMS

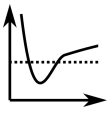
We study algorithms that are used in practice for three biological problems: sequence alignment, RNA folding, and predicting topologically associated domains in DNA. In these applications, there are two unifying similarities. First, algorithmic performance is measured in terms of the distance between the algorithm's output and a ground-truth solution. In most cases, this solution is discovered using laboratory experimentation, so it is only available for the instances in the training set. Second, these algorithms use dynamic programming to maximize parameterized objective functions. This objective function represents a surrogate optimization criterion for the dynamic programming algorithm, whereas utility measures how well the algorithm's output resembles the ground truth. There



(a) Constant function (zero oscillations).



(b) Linear function (one oscillation).



(c) Inverse-quadratic function of the form $h(\rho) = \frac{a}{\rho^2} + b\rho + c$ (two oscillations).

Figure 3: Each solid line is a function with bounded oscillations and each dotted line is an arbitrary threshold. Many parameterized algorithms have piecewise-structured duals with piece functions from these families.

may be multiple solutions that maximize this objective function, which we call *co-optimal*. Although co-optimal solutions have the same objective function value, they may have different utilities. To handle tie-breaking, we assume that in any region of the parameter space where the set of co-optimal solutions is fixed, the algorithm's output is also fixed, which is typically true in practice.

4.1 Global pairwise sequence alignment

In pairwise sequence alignment, the goal is to line up strings in order to identify regions of similarity. In biology, for example, these similar regions indicate functional, structural, or evolutionary relationships between the sequences. Formally, let Σ be an alphabet and let $S_1, S_2 \in \Sigma^n$ be two sequences. 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 del is a function that deletes every $-$, or *gap character*. There are many features of an alignment that one might wish to optimize, such as the number of *matches* ($\tau_1[i] = \tau_2[i]$), *mismatches* ($\tau_1[i] \neq \tau_2[i]$), *indels* ($\tau_1[i] = -$ or $\tau_2[i] = -$), and *gaps* (maximal sequences of consecutive gap characters in $\tau \in \{\tau_1, \tau_2\}$). We denote these features using functions $\ell_1, \ldots, \ell_d$ that map pairs of sequences (S_1, S_2) and alignments L to $\mathbb{R}$.

A common dynamic programming algorithm $A_{\boldsymbol{\rho}}$ [35, 72] returns the alignment L that maximizes the objective function

$$\rho[1] \cdot \ell_1(S_1, S_2, L) + \cdots + \rho[d] \cdot \ell_d(S_1, S_2, L), \tag{8}$$

where $\boldsymbol{\rho} = (\rho[1], \ldots, \rho[d]) \in \mathbb{R}^d$ is a parameter vector. We denote the output alignment as $A_{\boldsymbol{\rho}}(S_1, S_2)$. As Gusfield et al. [39] wrote, "there is considerable disagreement among molecular biologists about the correct choice" of a parameter setting $\boldsymbol{\rho}$. We assume there is a utility function that characterizes an alignment's quality, denoted $u(S_1, S_2, L) \in \mathbb{R}$. For example, $u(S_1, S_2, L)$ might measure the distance between L and a "ground truth" alignment of S_1 and S_2 [68]. We then define $u_{\boldsymbol{\rho}}(S_1, S_2) = u\left(S_1, S_2, A_{\boldsymbol{\rho}}(S_1, S_2)\right)$ to be the utility of the alignment returned by the algorithm $A_{\boldsymbol{\rho}}$.

In the following lemma, we prove that the set of utility functions $u_{\boldsymbol{\rho}}$ has piecewise-structured dual functions.

Lemma 4.1. *Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \left\{u_{\boldsymbol{\rho}} : (S_1, S_2) \mapsto u\left(S_1, S_2, A_{\boldsymbol{\rho}}(S_1, S_2)\right) \mid \boldsymbol{\rho} \in \mathbb{R}^d\right\}.$$

The dual class $\mathcal{U}^*$ is $\left(\mathcal{F}, \mathcal{G}, 4^n n^{4n+2}\right)$-piecewise decomposable, where $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_{\boldsymbol{\rho}} \mapsto c$ and $\mathcal{G} = \left\{g_{\boldsymbol{a}} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^d\right\}$ consists of halfspace indicator functions $g_{\boldsymbol{a}} : u_{\boldsymbol{\rho}} \mapsto \mathbb{I}_{\{\boldsymbol{a} \cdot \boldsymbol{\rho} < 0\}}$.

Proof. Fix a sequence pair S_1 and S_2. Let $\mathcal{L}$ be the set of alignments the algorithm returns as we range over all parameter vectors $\boldsymbol{\rho} \in \mathbb{R}^d$. In other words, $\mathcal{L} = \left\{A_{\boldsymbol{\rho}}(S_1, S_2) \mid \boldsymbol{\rho} \in \mathbb{R}^d\right\}$. In the full version [7], we prove that $|\mathcal{L}| \leq 2^n n^{2n+1}$. For any alignment $L \in \mathcal{L}$, the algorithm $A_{\boldsymbol{\rho}}$ will return L if and only if

$$\begin{aligned} &\rho[1] \cdot \ell_1(S_1, S_2, L) + \cdots + \rho[d] \cdot \ell_d(S_1, S_2, L) \\ >\ &\rho[1] \cdot \ell_1\left(S_1, S_2, L'\right) + \cdots + \rho[d] \cdot \ell_d\left(S_1, S_2, L'\right) \end{aligned} \tag{9}$$

for all $L' \in \mathcal{L} \setminus \{L\}$. 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 $\boldsymbol{\rho}$ in a single connected component of $\mathbb{R}^d \setminus \mathcal{H}$, the output of the algorithm parameterized by $\boldsymbol{\rho}$, $A_{\boldsymbol{\rho}}(S_1, S_2)$, is fixed. (As is standard, $\mathbb{R}^d \setminus \mathcal{H}$ indicates set removal.) This means that for any connected component R of $\mathbb{R}^d \setminus \mathcal{H}$, there exists a real value c_R such that $u_{\boldsymbol{\rho}}(S_1, S_2) = c_R$ for all $\boldsymbol{\rho} \in R$. By definition of the dual, this means that $u^*_{S_1, S_2}\left(u_{\boldsymbol{\rho}}\right) = u_{\boldsymbol{\rho}}(S_1, S_2) = c_R$ as well.

We now use this structure to show that the dual class $\mathcal{U}^*$ is $\left(\mathcal{F}, \mathcal{G}, 4^n n^{4n+2}\right)$-piecewise decomposable, as per Definition 3.2. 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 $g_{\boldsymbol{a}} : u_{\boldsymbol{\rho}} \mapsto \mathbb{I}_{\{\boldsymbol{a} \cdot \boldsymbol{\rho} < 0\}}$ and $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_{\boldsymbol{\rho}} \mapsto c$. For each pair of alignments $L, L' \in \mathcal{L}$, let $g^{(L,L')} \in \mathcal{G}$ correspond to the halfspace represented in Equation (9). Order these $k := \binom{|\mathcal{L}|}{2}$ 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 the function $f^{(\boldsymbol{b}_R)} \in \mathcal{F}$ as $f^{(\boldsymbol{b}_R)} = f_{c_R}$, so $f^{(\boldsymbol{b}_R)}\left(u_{\boldsymbol{\rho}}\right) = c_R$ for all $\boldsymbol{\rho} \in \mathbb{R}^d$. Meanwhile, for every vector $\boldsymbol{b}$ not corresponding to a sign pattern of the k hyperplanes, let $f^{(\boldsymbol{b})} = f_0$, so $f^{(\boldsymbol{b})}\left(u_{\boldsymbol{\rho}}\right) = 0$ for all $\boldsymbol{\rho} \in \mathbb{R}^d$. In this way, for every $\boldsymbol{\rho} \in \mathbb{R}^d$,

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

as desired. □

In the full version [7], we prove $\text{Pdim}(\mathcal{F}^*) = 0$ and $\text{VCdim}(\mathcal{G}^*) = d + 1$. Theorem 3.3 and Lemma 4.1 therefore imply that $\text{Pdim}(\mathcal{U}) = O(nd \ln n)$. Moreover, in the full version [7], we provide guarantees for algorithms that align more than two sequences.

Tighter guarantees for a structured algorithm subclass: the affine-gap model. A line of prior work [29, 39, 61, 62] analyzed a specific instantiation of the objective function (8) where $d = 3$. In this case, we can obtain a pseudo-dimension bound of $O(\ln n)$, which is

exponentially better than the bound implied by Lemma 4.1. Given a pair of sequences $S_1, S_2 \in \Sigma^n$, the dynamic programming algorithm $A_{\boldsymbol{\rho}}$ returns the alignment L maximizes the objective function

$$\textsc{mt}(S_1,S_2,L)-\rho[1]\textsc{ms}(S_1,S_2,L)-\rho[2]\textsc{id}(S_1,S_2,L)-\rho[3]\textsc{gp}(S_1,S_2,L),$$

where $\textsc{mt}(S_1,S_2,L)$ is the number of matches, $\textsc{ms}(S_1,S_2,L)$ is the number of mismatches, $\textsc{id}(S_1,S_2,L)$ equals the number of indels, $\textsc{gp}(S_1,S_2,L)$ is the number of gaps, and $\boldsymbol{\rho} = (\rho[1],\rho[2],\rho[3]) \in \mathbb{R}^3$ is a parameter vector. We denote the output alignment as $A_{\boldsymbol{\rho}}(S_1,S_2)$. This is known as the *affine-gap scoring model*. We exploit specific structure exhibited by this algorithm family to obtain the exponential pseudo-dimension improvement. This useful structure guarantees that for any pair of sequences S_1 and S_2, there are only $O\left(n^{3/2}\right)$ different alignments the algorithm family $\left\{A_{\boldsymbol{\rho}} \mid \boldsymbol{\rho} \in \mathbb{R}^3\right\}$ might produce as we range over parameter vectors [29, 39, 61]. This bound is exponentially smaller than the generic bound of $4^n n^{4n+2}$ that we use in the proof of Lemma 4.1. The proof is in the full version [7].

Lemma 4.2. *Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \left\{u_{\boldsymbol{\rho}} : (S_1,S_2) \mapsto u\left(S_1,S_2,A_{\boldsymbol{\rho}}(S_1,S_2)\right) \mid \boldsymbol{\rho} \in \mathbb{R}^3_{\geq 0}\right\}.$$

The dual class $\mathcal{U}^*$ is $\left(\mathcal{F},\mathcal{G},O\left(n^3\right)\right)$-piecewise decomposable, where $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_{\boldsymbol{\rho}} \mapsto c$ and where $\mathcal{G} = \left\{g_{\boldsymbol{a}} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^4\right\}$ consists of halfspace indicator functions $g_{\boldsymbol{a}} : u_{\boldsymbol{\rho}} \mapsto \mathbb{I}_{\{a[1]\rho[1]+a[2]\rho[2]+a[3]\rho[3]<a[4]\}}$.

Theorem 3.3 and Lemma 4.2 imply that $\text{Pdim}(\mathcal{U}) = O(\ln n)$. We also prove that this pseudo-dimension bound is tight up to constant factors. In this lower bound proof, our utility function u is the *Q score* between a given alignment L of two sequences (S_1,S_2) and the ground-truth alignment L^* (the Q score is also known as the *SPS score* in the case of multiple sequence alignment [24]). The Q score between L and the ground-truth alignment L^* is the fraction of aligned letter pairs in L^* that are correctly reproduced in L. For example, the following alignment L has a Q score of $\frac{2}{3}$ because it correctly aligns the two pairs of Cs, but not the pair of Gs:

$$L = \begin{bmatrix} \mathtt{G} & \mathtt{A} & \mathtt{T} & \mathtt{C} & \mathtt{C} \\ \mathtt{A} & \mathtt{G} & \mathtt{-} & \mathtt{C} & \mathtt{C} \end{bmatrix} \qquad L^* = \begin{bmatrix} \mathtt{-} & \mathtt{G} & \mathtt{A} & \mathtt{T} & \mathtt{C} & \mathtt{C} \\ \mathtt{A} & \mathtt{G} & \mathtt{-} & \mathtt{-} & \mathtt{C} & \mathtt{C} \end{bmatrix}.$$

We use the notation $u(S_1,S_2,L) \in [0,1]$ to denote the Q score between L and the ground-truth alignment of S_1 and S_2. The full proof of the following theorem is in the full version [7].

Theorem 4.3. *There exists a set $\left\{A_{\boldsymbol{\rho}} \mid \boldsymbol{\rho} \in \mathbb{R}^3_{\geq 0}\right\}$ of co-optimal-constant algorithms and an alphabet Σ such that the set of functions $\mathcal{U} = \left\{u_{\boldsymbol{\rho}} : (S_1,S_2) \mapsto u\left(S_1,S_2,A_{\boldsymbol{\rho}}(S_1,S_2)\right) \mid \boldsymbol{\rho} \in \mathbb{R}^3_{\geq 0}\right\}$, which map sequence pairs $S_1,S_2 \in \cup_{i=1}^n \Sigma^i$ of length at most n to $[0,1]$, has a pseudo-dimension of $\Omega(\log n)$.*

Proof sketch. In this proof sketch, we illustrate the way in which two sequences pairs can be shattered, and then describe how the proof can be generalized to $\Theta(\log n)$ sequence pairs.

Setup. Our setup consists of the following three elements: the alphabet, the two sequence pairs $\left(S_1^{(1)},S_2^{(1)}\right)$ and $\left(S_1^{(2)},S_2^{(2)}\right)$, and ground-truth alignments of these pairs. We detail these elements below:

(1) Our alphabet consists of twelve characters: $\{\mathtt{a}_i,\mathtt{b}_i,\mathtt{c}_i,\mathtt{d}_i\}_{i=1}^3$.

(2) The two sequence pairs are comprised of three subsequence pairs: $\left(t_1^{(1)},t_2^{(1)}\right)$, $\left(t_1^{(2)},t_2^{(2)}\right)$, and $\left(t_1^{(3)},t_2^{(3)}\right)$, where

$$t_1^{(1)} = \mathtt{a}_1\mathtt{b}_1\mathtt{d}_1 \quad t_1^{(2)} = \mathtt{a}_2\mathtt{a}_2\mathtt{b}_2\mathtt{d}_2 \quad \text{and} \quad t_1^{(3)} = \mathtt{a}_3\mathtt{a}_3\mathtt{a}_3\mathtt{b}_3\mathtt{d}_3 \\ t_2^{(1)} = \mathtt{b}_1\mathtt{c}_1\mathtt{d}_1 \quad t_2^{(2)} = \mathtt{b}_2\mathtt{c}_2\mathtt{c}_2\mathtt{d}_2 \quad \text{and} \quad t_2^{(3)} = \mathtt{b}_3\mathtt{c}_3\mathtt{c}_3\mathtt{c}_3\mathtt{d}_3. \tag{10}$$

We define the two sequence pairs as

$$S_1^{(1)} = t_1^{(1)}t_1^{(2)}t_1^{(3)} = \mathtt{a}_1\mathtt{b}_1\mathtt{d}_1\mathtt{a}_2\mathtt{a}_2\mathtt{b}_2\mathtt{d}_2\mathtt{a}_3\mathtt{a}_3\mathtt{a}_3\mathtt{b}_3\mathtt{d}_3 \\ S_2^{(1)} = t_2^{(1)}t_2^{(2)}t_2^{(3)} = \mathtt{b}_1\mathtt{c}_1\mathtt{d}_1\mathtt{b}_2\mathtt{c}_2\mathtt{c}_2\mathtt{d}_2\mathtt{b}_3\mathtt{c}_3\mathtt{c}_3\mathtt{c}_3\mathtt{d}_3$$

$$\text{and} \quad S_1^{(2)} = t_1^{(2)} = \mathtt{a}_2\mathtt{a}_2\mathtt{b}_2\mathtt{d}_2 \\ S_2^{(2)} = t_2^{(2)} = \mathtt{b}_2\mathtt{c}_2\mathtt{c}_2\mathtt{d}_2.$$

(3) Finally, we define ground-truth alignments of the two sequence pairs $\left(S_1^{(1)},S_2^{(1)}\right)$ and $\left(S_1^{(2)},S_2^{(2)}\right)$. We define the ground-truth alignment of $\left(S_1^{(1)},S_2^{(1)}\right)$ to be

$$\begin{matrix} \mathtt{a}_1 & \mathtt{b}_1 & \mathtt{-} & \mathtt{d}_1 & \mathtt{a}_2 & \mathtt{a}_2 & \mathtt{b}_2 & \mathtt{-} & \mathtt{-} & \mathtt{d}_2 & \mathtt{a}_3 & \mathtt{a}_3 & \mathtt{a}_3 & \mathtt{b}_3 & \mathtt{-} & \mathtt{-} & \mathtt{-} & \mathtt{d}_3 \\ \mathtt{b}_1 & \mathtt{-} & \mathtt{c}_1 & \mathtt{d}_1 & \mathtt{-} & \mathtt{-} & \mathtt{b}_2 & \mathtt{c}_2 & \mathtt{c}_2 & \mathtt{d}_2 & \mathtt{b}_3 & \mathtt{-} & \mathtt{-} & \mathtt{-} & \mathtt{c}_3 & \mathtt{c}_3 & \mathtt{c}_3 & \mathtt{d}_3 \end{matrix}. \tag{11}$$

The most important properties of this alignment are that the $\mathtt{d}_j$ characters are always matching and the $\mathtt{b}_j$ characters alternate between matching and not matching. Similarly, we define the ground-truth alignment of $\left(S_1^{(2)},S_2^{(2)}\right)$ to be

$$\begin{matrix} \mathtt{a}_2 & \mathtt{a}_2 & \mathtt{b}_2 & \mathtt{-} & \mathtt{-} & \mathtt{d}_2 \\ \mathtt{-} & \mathtt{-} & \mathtt{b}_2 & \mathtt{c}_2 & \mathtt{c}_2 & \mathtt{d}_2 \end{matrix}.$$

Shattering. We now show that these two sequence pairs can be shattered. A key step is proving that $u_{(0,\rho[2],0)}\left(S_1^{(1)},S_2^{(1)}\right)$ and $u_{(0,\rho[2],0)}\left(S_1^{(2)},S_2^{(2)}\right)$ have the following form:

$$u_{(0,\rho[2],0)}\left(S_1^{(1)},S_2^{(1)}\right) = \begin{cases} \frac{4}{6} & \text{if } \rho[2] \leq \frac{1}{6} \\ \frac{5}{6} & \text{if } \frac{1}{6} < \rho[2] \leq \frac{1}{4} \\ \frac{4}{6} & \text{if } \frac{1}{4} < \rho[2] \leq \frac{1}{2} \\ \frac{5}{6} & \text{if } \rho[2] > \frac{1}{2} \end{cases} \tag{12}$$

$$\text{and} \quad u_{(0,\rho[2],0)}\left(S_1^{(2)},S_2^{(2)}\right) = \begin{cases} 1 & \text{if } \rho[2] \leq \frac{1}{4} \\ \frac{1}{2} & \text{if } \rho[2] > \frac{1}{4} \end{cases}.$$

The form of $u_{(0,\rho[2],0)}\left(S_1^{(1)},S_2^{(1)}\right)$ is illustrated by Figure 4. It is then straightforward to verify that the two sequence pairs are shattered by the parameter settings $(0,0,0)$, $\left(0,\frac{1}{5},0\right)$, $\left(0,\frac{1}{3},0\right)$, and $(0,1,0)$ with the witnesses $z_1 = z_2 = \frac{3}{4}$. In other words, the mismatch and gap parameters are set to 0 and the indel parameter $\rho[2]$ takes the values $\left\{0,\frac{1}{5},\frac{1}{3},1\right\}$.

Proof sketch of Equation (12). The full proof that Equation (12) holds follows the following high-level reasoning:

(1) First, we prove that under the algorithm's output alignment, the $\mathtt{d}_j$ characters will always be matching. Intuitively, this is because the algorithm's objective function will always be maximized when each subsequence $t_1^{(j)}$ is aligned with $t_2^{(j)}$.

(2) Second, we prove that the characters $\mathtt{b}_j$ will be matched if and only if $\rho[2] \leq \frac{1}{2j}$. Intuitively, this is because in order to match these characters, we must pay with $2j$ indels. Since the

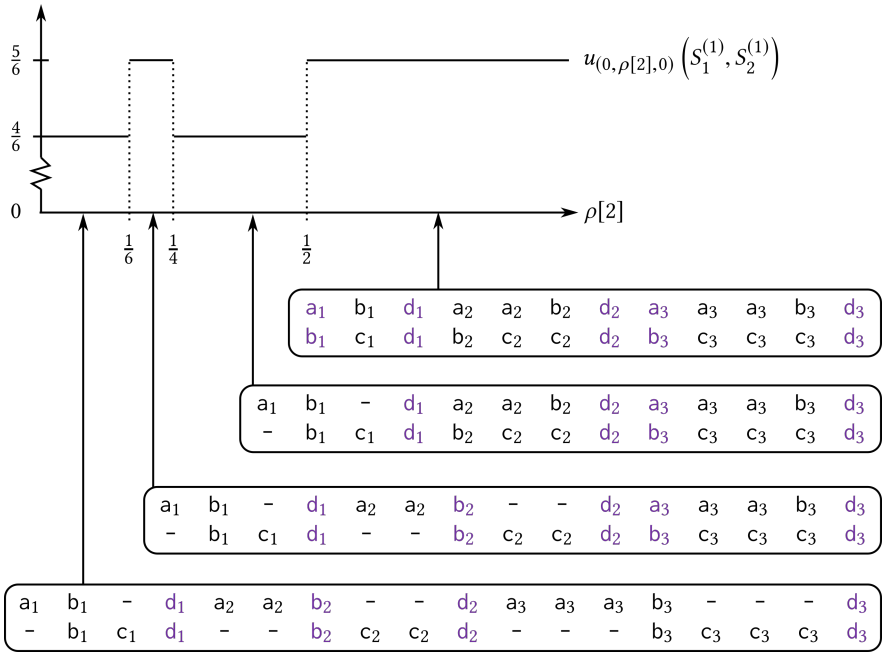


Figure 4: The form of $u_{(0,\rho[2],0)}\left(S_1^{(1)}, S_2^{(1)}\right)$ as a function of the indel parameter $\rho[2]$. When $\rho[2] \le \frac{1}{6}$, the algorithm returns the bottom alignment. When $\frac{1}{6} < \rho[2] \le \frac{1}{4}$, the algorithm returns the alignment that is second to the bottom. When $\frac{1}{4} < \rho[2] \le \frac{1}{2}$, the algorithm returns the alignment that is second to the top. Finally, when $\rho[2] > \frac{1}{2}$, the algorithm returns the top alignment. The purple characters denote which characters are correctly aligned according to the ground-truth alignment (Equation (11)).

objective function is $\textsc{mt}\left(S_1^{(1)}, S_2^{(1)}, L\right) - \rho[2] \cdot \textsc{id}\left(S_1^{(1)}, S_2^{(1)}, L\right)$, the 1 match will be worth the $2j$ indels if and only if $1 \ge 2j\rho[2]$.

These two properties in conjunction mean that when $\rho[2] > \frac{1}{2}$, none of the b_j characters are matched, so the characters that are correctly aligned (as per the ground-truth alignment (Equation (11))) in the algorithm's output are $(\mathsf{a}_1, \mathsf{b}_1)$, $(\mathsf{d}_1, \mathsf{d}_1)$, $(\mathsf{d}_2, \mathsf{d}_2)$, $(\mathsf{a}_3, \mathsf{b}_3)$, and $(\mathsf{d}_3, \mathsf{d}_3)$, as illustrated by purple in the top alignment of Figure 4. Since there are 6 aligned letters in the ground-truth alignment, the Q score is $\frac{5}{6}$, or in other words, $u_{(0,\rho[2],0)}\left(S_1^{(1)}, S_2^{(1)}\right) = \frac{5}{6}$.

When $\rho[2]$ shifts to the next-smallest interval $\left(\frac{1}{4}, \frac{1}{2}\right]$, the indel penalty $\rho[2]$ is sufficiently small that the b_1 characters will align. Thus we lose the correct alignment $(\mathsf{a}_1, \mathsf{b}_1)$, and the Q score drops to $\frac{4}{6}$. Similarly, if we decrease $\rho[2]$ to the next-smallest interval $\left(\frac{1}{6}, \frac{1}{4}\right]$, the b_2 characters will align, which is correct under the ground-truth alignment (Equation (11)). Thus the Q score increases back to $\frac{5}{6}$. Finally, by the same logic, when $\rho[2] \le \frac{1}{6}$, we lose the correct alignment $(\mathsf{a}_3, \mathsf{b}_3)$ in favor of the alignment of the b_3 characters, so the Q score falls to $\frac{4}{6}$. In this way, we prove the form of $u_{(0,\rho[2],0)}\left(S_1^{(1)}, S_2^{(1)}\right)$ from Equation (12). A parallel argument proves the form of $u_{(0,\rho[2],0)}\left(S_1^{(2)}, S_2^{(2)}\right)$.

Generalization to shattering $\Theta(\log n)$ sequence pairs. This proof intuition naturally generalizes to $\Theta(\log n)$ sequence pairs of length $O(n)$ by expanding the number of subsequences $t_i^{(j)}$ *a la* Equation (10). In essence, if we define $S_1^{(1)} = t_1^{(1)} t_1^{(2)} \cdots t_1^{(k)}$ and $S_2^{(1)} = t_2^{(1)} t_2^{(2)} \cdots t_2^{(k)}$ for a carefully-chosen $k = \Theta\left(\sqrt{n}\right)$, then we can force $u_{(0,\rho[2],0)}\left(S_1^{(1)}, S_2^{(1)}\right)$ to oscillate $O(n)$ times. Similarly, if we define $S_1^{(2)} = t_1^{(2)} t_1^{(4)} \cdots t_1^{(k-1)}$ and $S_2^{(2)} = t_2^{(2)} t_2^{(4)} \cdots t_2^{(k-1)}$, then we can force $u_{(0,\rho[2],0)}\left(S_1^{(1)}, S_2^{(1)}\right)$ to oscillate half as many times, and so on. This construction allows us to shatter $\Theta(\log n)$ sequences. □

4.2 RNA folding

RNA molecules have many essential roles including protein coding and enzymatic functions [40]. RNA is assembled as a chain of *bases* denoted A, U, C, and G. It is often found as a single strand folded onto itself with non-adjacent bases physically bound together. RNA folding algorithms infer the way strands would naturally fold, shedding light on their functions. Given a sequence $S \in \{\mathsf{A}, \mathsf{U}, \mathsf{C}, \mathsf{G}\}^n$, we represent a folding by a set of pairs $\phi \subset [n] \times [n]$. If $(i, j) \in \phi$, then the i^{th} and j^{th} bases of S bind together. Typically, the bases A and U bind together, as do C and G. Other matchings are likely less stable. We assume that the foldings do not contain any *pseudoknots*, which are pairs $(i, j), (i', j')$ that cross with $i < i' < j < j'$.

A well-studied algorithm returns a folding that maximizes a parameterized objective function [60]. 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,

the algorithm A_ρ uses dynamic programming to return the folding $A_\rho(S)$ that maximizes

$$\rho\,|\phi| + (1-\rho)\sum_{(i,j)\in\phi} M_{S[i],S[j],S[i-1],S[j+1]}\mathbb{I}_{\{(i-1,j+1)\in\phi\}}, \quad (13)$$

where $\rho \in [0,1]$ is a parameter and $M_{S[i],S[j],S[i-1],S[j+1]} \in \mathbb{R}$ is a score for having neighboring pairs of the letters $(S[i], S[j])$ and $(S[i-1], S[j+1])$. These scores help identify stable sub-structures.

We assume there is a utility function that characterizes a folding's quality, denoted $u(S,\phi)$. For example, $u(S,\phi)$ might measure the fraction of pairs shared between ϕ and a "ground-truth" folding, obtained via expensive computation or laboratory experiments.

Lemma 4.4. *Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \left\{u_\rho : S \mapsto u\left(S, A_\rho\left(S\right)\right) \mid \rho \in \mathbb{R}\right\}.$$

The dual class $\mathcal{U}^*$ is $\left(\mathcal{F}, \mathcal{G}, n^2\right)$-piecewise decomposable, where $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\rho \mapsto \mathbb{I}_{\{\rho<a\}}$ and $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_\rho \mapsto c$.

Proof. Fix a sequence S. Let Φ be the set of alignments that the algorithm returns as we range over all parameters $\rho \in \mathbb{R}$. In other words, $\Phi = \{A_\rho(S) \mid \rho \in [0,1]\}$. We know that every folding has length at most $n/2$. For any $k \in \{0,\ldots,n/2\}$, let ϕ_k be the folding of length k that maximizes the right-hand-side of Equation (13):

$$\phi_k = \text{argmax}_{\phi:|\phi|=k} \sum_{(i,j)\in\phi} M_{S[i],S[j],S[i-1],S[j+1]}\mathbb{I}_{\{(i-1,j+1)\in\phi\}}.$$

The folding the algorithm returns will be one of $\left\{\phi_0,\ldots,\phi_{n/2}\right\}$, so $|\Phi| \le \frac{n}{2}+1$.

Fix an arbitrary folding $\phi \in \Phi$. We know that ϕ will be the folding returned by the algorithm $A_\rho(S)$ if and only if

$$\begin{aligned}&\rho\,|\phi| + (1-\rho)\sum_{(i,j)\in\phi} M_{S[i],S[j],S[i-1],S[j+1]}\mathbb{I}_{\{(i-1,j+1)\in\phi\}}\\ \ge\ &\rho\left|\phi'\right| + (1-\rho)\sum_{(i,j)\in\phi'} M_{S[i],S[j],S[i-1],S[j+1]}\mathbb{I}_{\{(i-1,j+1)\in\phi'\}}\end{aligned}$$

for all $\phi' \in \Phi \setminus \{\phi\}$. Since these functions are linear in ρ, this means that there is a set of $T \le \binom{|\Phi|}{2} \le n^2$ intervals $[\rho_1,\rho_2), [\rho_2,\rho_3), \ldots, [\rho_T,\rho_{T+1}]$ with $\rho_1 := 0 < \rho_2 < \cdots < \rho_T < 1 := \rho_{T+1}$ such that for any one interval I, across all $\rho \in I$, $A_\rho(S)$ is fixed. This means that for any one interval $[\rho_i,\rho_{i+1})$, there exists a real value c_i such that $u_\rho(S) = c_i$ for all $\rho \in [\rho_i,\rho_{i+1})$. By definition of the dual, this means that $u_S^*(u_\rho) = u_\rho(S) = c_i$ as well.

We now use this structure to show that the dual class $\mathcal{U}^*$ is $\left(\mathcal{F},\mathcal{G},n^2\right)$-piecewise decomposable, as per Definition 3.2. Recall that $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\rho \mapsto \mathbb{I}_{\{\rho<a\}}$ and $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_\rho \mapsto c$. We claim that there exists a function $f^{(\boldsymbol{b})} \in \mathcal{F}$ for every vector $\boldsymbol{b} \in \{0,1\}^T$ such that for every $\rho \in [0,1]$,

$$u_S^*(u_\rho) = \sum_{\boldsymbol{b}\in\{0,1\}^T} \mathbb{I}_{\{g_{\rho_i}(u_\rho)=b[i],\,\forall i\in[T]\}} f^{(\boldsymbol{b})}(u_\rho). \quad (14)$$

To see why, suppose $\rho \in [\rho_i,\rho_{i+1})$ for some $i \in [T]$. Then $g_{\rho_j}(u_\rho) = \mathbb{I}_{\{\rho\le\rho_j\}} = 1$ for all $j \ge i+1$ and $g_{\rho_j}(u_\rho) = \mathbb{I}_{\{\rho\le\rho_j\}} = 0$ for all $j \le i$. Let $\boldsymbol{b}_i \in \{0,1\}^T$ be the vector that has only 0's in its first i coordinates and all 1's in its remaining $T-i$ coordinates. For all $i \in [T]$, we define $f^{(\boldsymbol{b}_i)} = f_{c_i}$, so $f^{(\boldsymbol{b}_i)}\left(u_\rho\right) = c_i$ for all $\rho \in [0,1]$. For any other $\boldsymbol{b}$, we set $f^{(\boldsymbol{b})} = f_0$, so $f^{(\boldsymbol{b})}\left(u_\rho\right) = 0$ for all $\rho \in [0,1]$. Therefore, Equation (14) holds. □

Since constant functions have zero oscillations, Lemmas 3.9 and 4.4 imply that $\text{Pdim}(\mathcal{U}) = O(\ln n)$.

4.3 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. *Topologically associating domains (TADs)* are contiguous segments of the genome that fold into compact regions. More formally, given the genome length n, a TAD set is a set

$$T = \{(i_1,j_1),\ldots,(i_t,j_t)\} \subset [n]\times[n]$$

such that $i_1 < j_1 < i_2 < j_2 < \cdots < i_t < j_t$. If $(i,j) \in T$, the bases within the corresponding substring physically interact more frequently with each other than with other bases. Disrupting TAD boundaries can affect the expression of nearby genes, which can trigger diseases such as congenital malformations and cancer [52].

The contact frequency of any two genome locations, denoted by a matrix $M \in \mathbb{R}^{n\times n}$, can be measured via experiments [49]. A dynamic programming algorithm A_ρ introduced by Filippova et al. [30] returns the TAD set $A_\rho(M)$ that maximizes

$$\sum_{(i,j)\in T} s_\rho(i,j) - \mu_\rho(j-i), \quad (15)$$

where $\rho \ge 0$ is a parameter, $s_\rho(i,j) = \frac{1}{(j-i)^\rho}\sum_{i\le p<q\le j} M_{pq}$ is the scaled density of the subgraph induced by the interactions between genomic loci i and j, and $\mu_\rho(d) = \frac{1}{n-d}\sum_{t=0}^{n-d-1} s_\rho(t,t+d)$ is the mean value of s_ρ over all sub-matrices of length d along the diagonal of M. We note that unlike the sequence alignment and RNA folding algorithms, the parameter ρ appears in the exponent of the objective function.

We assume there is a utility function that characterizes the quality of a TAD set T, denoted $u(M,T) \in \mathbb{R}$. For example, $u(M,T)$ might measure the fraction of TADs in T that are in the correct location with respect to a ground-truth TAD set.

Lemma 4.5. *Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \left\{u_\rho : M \mapsto u\left(M, A_\rho\left(M\right)\right) \mid \rho \in \mathbb{R}\right\}.$$

The dual class $\mathcal{U}^*$ is $\left(\mathcal{F},\mathcal{G},2n^2 4^{n^2}\right)$-piecewise decomposable, where $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\rho \mapsto \mathbb{I}_{\{\rho<a\}}$ and $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_\rho \mapsto c$.

Proof. Fix a matrix M. We rewrite Equation (15) as follows:

$$A_\rho(M) = \text{argmax}_{T\subset[n]\times[n]} \sum_{(i,j)\in T} \frac{c_{ij}}{(j-i)^\rho},$$

where

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

is a constant that does not depend on ρ.

Let $\mathcal{T}$ be the set of TAD sets that the algorithm returns as we range over all parameters $\rho \geq 0$. In other words, $\mathcal{T} = \{A_\rho(M) \mid \rho \geq 0\}$. Since each TAD set is a subset of $[n] \times [n]$, $|\mathcal{T}| \leq 2^{n^2}$. For any TAD set $T \in \mathcal{T}$, the algorithm A_ρ will return T if and only if

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

for all $T' \in \mathcal{T} \setminus \{T\}$. This means that as we range ρ over $\mathbb{R}_{\geq 0}$, the TAD set returned by algorithm $A_\rho(M)$ will only change when

$$\sum_{(i,j)\in T} \frac{c_{ij}}{(j-i)^\rho} - \sum_{(i',j')\in T'} \frac{c_{i'j'}}{(j'-i')^\rho} = 0 \tag{16}$$

for some $T, T' \in \mathcal{T}$. As a result of Rolle's Theorem (detailed in the full version [7]), we know that Equation (16) has at most $|T| + |T'| \leq 2n^2$ solutions. This means there are $t \leq 2n^2 \binom{|\mathcal{T}|}{2} \leq 2n^2 4^{n^2}$ intervals $[\rho_1, \rho_2), [\rho_2, \rho_3), \ldots, [\rho_t, \rho_{t+1})$ with $\rho_1 := 0 < \rho_2 < \cdots < \rho_t < \infty := \rho_{t+1}$ that partition $\mathbb{R}_{\geq 0}$ such that across all ρ within any one interval $[\rho_i, \rho_{i+1})$, the TAD set returned by algorithm $A_\rho(M)$ is fixed. Therefore, there exists a real value c_i such that $u_\rho(M) = c_i$ for all $\rho \in [\rho_i, \rho_{i+1})$. By definition of the dual, this means that $u^*_M(u_\rho) = u_\rho(M) = c_i$ as well.

We now use this structure to show that the dual class $\mathcal{U}^*$ is $\left(\mathcal{F}, \mathcal{G}, 2n^2 4^{n^2}\right)$-piecewise decomposable, as per Definition 3.2. Recall that $\mathcal{G} = \{g_a : \mathcal{U} \to \{0,1\} \mid a \in \mathbb{R}\}$ consists of threshold functions $g_a : u_\rho \mapsto \mathbb{I}_{\{\rho < a\}}$ and $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_\rho \mapsto c$. We claim that there exists a function $f^{(\boldsymbol{b})} \in \mathcal{F}$ for every vector $\boldsymbol{b} \in \{0,1\}^t$ such that for every $\rho \geq 0$,

$$u^*_M(u_\rho) = \sum_{\boldsymbol{b}\in\{0,1\}^t} \mathbb{I}_{\{g_{\rho_i}(u_\rho) = b[i], \forall i \in [t]\}} f^{(\boldsymbol{b})}(u_\rho). \tag{17}$$

To see why, suppose $\rho \in [\rho_i, \rho_{i+1})$ for some $i \in [t]$. Then $g_{\rho_j}(u_\rho) = \mathbb{I}_{\{\rho \leq \rho_j\}} = 1$ for all $j \geq i+1$ and $g_{\rho_j}(u_\rho) = \mathbb{I}_{\{\rho \leq \rho_j\}} = 0$ for all $j \leq i$. Let $\boldsymbol{b}_i \in \{0,1\}^t$ be the vector that has only 0's in its first i coordinates and all 1's in its remaining $t-i$ coordinates. For all $i \in [t]$, we define $f^{(\boldsymbol{b}_i)} = f_{c_i}$, so $f^{(\boldsymbol{b}_i)}(u_\rho) = c_i$ for all $\rho \in [0,1]$. For any other $\boldsymbol{b}$, we set $f^{(\boldsymbol{b})} = f_0$, so $f^{(\boldsymbol{b})}(u_\rho) = 0$ for all $\rho \in [0,1]$. Therefore, Equation (17) holds. □

Since constant functions have zero oscillations, Lemmas 3.9 and 4.5 imply that $\text{Pdim}(\mathcal{U}) = O(n^2)$.

5 PARAMETERIZED VOTING MECHANISMS

A large body of economics research studies how to design protocols—or *mechanisms*—that help groups of agents come to collective decisions. For example, when children inherit an estate, how should they divide the property? When a jointly-owned company is dissolved, which partner should buy the others out? There is no one protocol that best answers these questions; the optimal mechanism depends on the setting at hand.

We study a family of mechanisms called *neutral affine maximizers (NAMs)* [55, 59, 65]. A NAM takes as input a set of agents' reported values for each possible outcome and returns one of those outcomes. A NAM can thus be thought of as an algorithm that the agents use to arrive at a single outcome. NAMs are *incentive compatible*, which means that each agent is incentivized to report his values truthfully. In order to satisfy incentive compatibility, each agent may have to make a payment. NAMs are also *budget-balanced* which means that the aggregated payments are redistributed among the agents.

Formally, we study a setting where there is a set of m alternatives and a set of n agents. Each agent i has a value $v_i(j) \in \mathbb{R}$ for each alternative $j \in [m]$. We denote all of his values as $\boldsymbol{v}_i \in \mathbb{R}^m$ and all n agents' values as $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \in \mathbb{R}^{nm}$. In this case, the unknown distribution $\mathcal{D}$ is over vectors $\boldsymbol{v} \in \mathbb{R}^{nm}$.

A NAM is defined by n parameters (one per agent)

$$\boldsymbol{\rho} = (\rho[1], \ldots, \rho[n]) \in \mathbb{R}^n_{\geq 0}$$

such that at least one agent is assigned a weight of zero. There is a *social choice function* $\psi_{\boldsymbol{\rho}} : \mathbb{R}^{nm} \to [m]$ which uses the values $\boldsymbol{v} \in \mathbb{R}^{nm}$ to choose an alternative $\psi_{\boldsymbol{\rho}}(\boldsymbol{v}) \in [m]$. In particular, $\psi_{\boldsymbol{\rho}}(\boldsymbol{v}) = \text{argmax}_{j\in[m]} \sum_{i=1}^n \rho[i] v_i(j)$ maximizes the agents' weighted values. Each agent i with zero weight $\rho[i] = 0$ is called a *sink agent* because his values do not influence the outcome. For every agent who is not a sink agent ($\rho[i] \neq 0$), their payment is defined as in the weighted version of the classic Vickrey-Clarke-Groves mechanism [20, 36, 71]. To achieve budget balance, these payments are given to the sink agent(s). More formally, let $j^* = \psi_{\boldsymbol{\rho}}(\boldsymbol{v})$ and for each agent i, let $j_{-i} = \text{argmax}_{j\in[m]} \sum_{i'\neq i} \rho[i'] v_{i'}(j)$. The payment function is defined as $p_i(\boldsymbol{v}) = \frac{1}{\rho[i]} \left(\sum_{i'\neq i} \rho[i'] v_{i'}(j^*) - \sum_{i'\neq i} \rho[i'] v_{i'}(j_{-i})\right)$ if $\rho[i] \neq 0$, $p_i(\boldsymbol{v}) = -\sum_{i'\neq i} p_{i'}(\boldsymbol{v})$ if $i = \min\{i' : \rho[i'] = 0\}$, and $p_i(\boldsymbol{v}) = 0$ otherwise.

We aim to optimize the expected social welfare

$$\mathop{\mathbb{E}}_{\boldsymbol{v}\sim\mathcal{D}} \left[\sum_{i=1}^n v_i\left(\psi_{\boldsymbol{\rho}}(\boldsymbol{v})\right)\right]$$

of the NAM's outcome $\psi_{\boldsymbol{\rho}}(\boldsymbol{v})$, so we define the utility function $u_{\boldsymbol{\rho}}(\boldsymbol{v}) = \sum_{i=1}^n v_i\left(\psi_{\boldsymbol{\rho}}(\boldsymbol{v})\right)$.

Lemma 5.1. *Let $\mathcal{U}$ be the set of functions*

$$\mathcal{U} = \left\{u_{\boldsymbol{\rho}} \mid \boldsymbol{\rho} \in \mathbb{R}^n_{\geq 0}, \{i \mid \rho[i] = 0\} \neq \emptyset\right\}.$$

The dual class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, m^2)$-piecewise decomposable, where $\mathcal{G} = \{g_{\boldsymbol{a}} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^n\}$ consists of halfspace indicators $g_{\boldsymbol{a}} : u_{\boldsymbol{\rho}} \mapsto \mathbb{I}_{\{\boldsymbol{\rho}\cdot\boldsymbol{a} \leq 0\}}$ and $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_{\boldsymbol{\rho}} \mapsto c$.

Proof. Fix a valuation vector $\boldsymbol{v} \in \mathbb{R}^{nm}$. We know that for any two alternatives $j, j' \in [m]$, the alternative j would be selected over j' so long as

$$\sum_{i=1}^n \rho[i] v_i(j) > \sum_{i=1}^n \rho[i] v_i(j'). \tag{18}$$

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

We now use this structure to show that the dual class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, m^2)$-piecewise decomposable, as per Definition 3.2. Recall that $\mathcal{G} = \{g_{\boldsymbol{a}} : \mathcal{U} \to \{0,1\} \mid \boldsymbol{a} \in \mathbb{R}^n\}$ consists of halfspace indicator functions $g_{\boldsymbol{a}} : u_{\boldsymbol{\rho}} \mapsto \mathbb{I}_{\{\boldsymbol{a}\cdot\boldsymbol{\rho} < 0\}}$ and $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$

consists of constant functions $f_c : u_{\boldsymbol{\rho}} \mapsto c$. For each pair of alternatives $j, j' \in \mathcal{L}$, let $g^{(j,j')} \in \mathcal{G}$ correspond to the halfspace represented in Equation (18). 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 corresponding sign pattern. Define the function $f^{(\boldsymbol{b}_R)} \in \mathcal{F}$ as $f^{(\boldsymbol{b}_R)} = f_{c_R}$, so $f^{(\boldsymbol{b}_R)}\left(u_{\boldsymbol{\rho}}\right) = c_R$ for all $\boldsymbol{\rho} \in \mathbb{R}^n$. Meanwhile, for every vector $\boldsymbol{b}$ not corresponding to a sign pattern of the k hyperplanes, let $f^{(\boldsymbol{b})} = f_0$, so $f^{(\boldsymbol{b})}\left(u_{\boldsymbol{\rho}}\right) = 0$ for all $\boldsymbol{\rho} \in \mathbb{R}^n$. In this way, for every $\boldsymbol{\rho} \in \mathbb{R}^n$,

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

as desired. □

Theorem 3.3 and Lemma 5.1 imply that the pseudo-dimension of $\mathcal{U}$ is $O(n \ln m)$. Next, we prove that the pseudo-dimension of $\mathcal{U}$ is at least $\frac{n}{2}$, which means that our pseudo-dimension upper bound is tight up to log factors.

Theorem 5.2. *Define* $\mathcal{U} = \left\{u_{\boldsymbol{\rho}} \mid \boldsymbol{\rho} \in \mathbb{R}^n_{\geq 0}, \{\rho[i] \mid i = 0\} \neq \emptyset\right\}$. *Then* $\text{Pdim}(\mathcal{U}) \geq \frac{n}{2}$.

Proof. Let the number of alternatives $m = 2$ and without loss of generality, suppose that n is even. To prove this theorem, we will identify a set of $N = \frac{n}{2}$ valuation vectors $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(N)}$ that are shattered by the set $\mathcal{U}$ of social welfare functions.

Let ϵ be an arbitrary number in $\left(0, \frac{1}{2}\right)$. For each $\ell \in [N]$, define agent i's values for the first and second alternatives under the ℓ^{th} valuation vector $\boldsymbol{v}^{(\ell)}$—namely, $v_i^{(\ell)}(1)$ and $v_i^{(\ell)}(2)$—as follows:

$$v_i^{(\ell)}(1) = \begin{cases} 1 & \text{if } \ell = i \\ 0 & \text{otherwise} \end{cases} \text{ and } v_i^{(\ell)}(2) = \begin{cases} \epsilon & \text{if } \ell = \frac{n}{2} + i \\ 0 & \text{otherwise.} \end{cases}$$

For example, if there are $n = 6$ agents, then across the $N = \frac{n}{2} = 3$ valuation vectors $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}$, the agents' values for the first alternative are defined as

$$\begin{bmatrix} v_1^{(1)}(1) & \cdots & v_6^{(1)}(1) \\ v_1^{(2)}(1) & \cdots & v_6^{(2)}(1) \\ v_1^{(3)}(1) & \cdots & v_6^{(3)}(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

and their values for the second alternative are defined as

$$\begin{bmatrix} v_1^{(1)}(2) & \cdots & v_6^{(1)}(2) \\ v_1^{(2)}(2) & \cdots & v_6^{(2)}(2) \\ v_1^{(3)}(2) & \cdots & v_6^{(3)}(2) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \epsilon & 0 & 0 \\ 0 & 0 & 0 & 0 & \epsilon & 0 \\ 0 & 0 & 0 & 0 & 0 & \epsilon \end{bmatrix}.$$

Let $\boldsymbol{b} \in \{0,1\}^N$ be an arbitrary bit vector. We will construct a NAM parameter vector $\boldsymbol{\rho}$ such that for any $\ell \in [N]$, if $b[\ell] = 0$, then the outcome of the NAM given bids $\boldsymbol{v}^{(\ell)}$ will be the second alternative, so $u_{\boldsymbol{\rho}}\left(\boldsymbol{v}^{(\ell)}\right) = \epsilon$ because there is always exactly one agent who has a value of ϵ for the second alternative, and every other agent has a value of 0. Meanwhile, if $b[\ell] = 0$, then the outcome of the NAM given bids $\boldsymbol{v}^{(\ell)}$ will be the first alternative, so $u_{\boldsymbol{\rho}}\left(\boldsymbol{v}^{(\ell)}\right) = 1$ because there is always exactly one agent who has a value of 1 for the first alternative, and every other agent has a value of 0. To construct this parameter vector $\boldsymbol{\rho}$, when $b[\ell] = 0$, $\boldsymbol{\rho}$ must ignore the values of agent ℓ in favor of the values of agent $\frac{n}{2} + \ell$. After all, under $\boldsymbol{v}^{(\ell)}$, agent ℓ has a value of 1 for the first alternative and agent $\frac{n}{2} + \ell$ has a value of ϵ for the second alternative, and all other values are 0. By a similar argument, when $b[\ell] = 1$, $\boldsymbol{\rho}$ must ignore the values of agent $\frac{n}{2} + \ell$ in favor of the values of agent ℓ. Specifically, we define $\boldsymbol{\rho} \in \{0,1\}^n$ as follows: for all $\ell \in [N] = \left[\frac{n}{2}\right]$, if $b[\ell] = 0$, then $\rho[\ell] = 0$ and $\rho\left[\frac{n}{2} + \ell\right] = 1$ and if $b[\ell] = 1$, then $\rho[\ell] = 1$ and $\rho\left[\frac{n}{2} + \ell\right] = 0$. All other entries of $\boldsymbol{\rho}$ are set to 0.

We claim that if $b[\ell] = 0$, then $u_{\boldsymbol{\rho}}\left(\boldsymbol{v}^{(\ell)}\right) = \epsilon$. To see why, we know that $\sum_{i=1}^n \rho[i] v_i^{(\ell)}(1) = \rho[\ell] v_\ell^{(\ell)}(1) = \rho[\ell] = 0$. Meanwhile, $\sum_{i=1}^n \rho[i] v_i^{(\ell)}(2) = \rho\left[\frac{n}{2} + \ell\right] v^{(\ell)}_{\frac{n}{2}+\ell}(1) = \epsilon$. Therefore, the outcome of the NAM is the second alternative. The social welfare of this alternative is ϵ, so $u_{\boldsymbol{\rho}}\left(\boldsymbol{v}^{(\ell)}\right) = \epsilon$.

Next, we claim that if $b[\ell] = 1$, then $u_{\boldsymbol{\rho}}\left(\boldsymbol{v}^{(\ell)}\right) = 1$. To see why, we know that $\sum_{i=1}^n \rho[i] v_i^{(\ell)}(1) = \rho[\ell] v_\ell^{(\ell)}(1) = \rho[\ell] = 1$. Meanwhile, $\sum_{i=1}^n \rho[i] v_i^{(\ell)}(2) = \rho\left[\frac{n}{2} + \ell\right] v^{(\ell)}_{\frac{n}{2}+\ell}(1) = 0$. Therefore, the NAM's outcome is the first alternative. The social welfare of this alternative is 1, so $u_{\boldsymbol{\rho}}\left(\boldsymbol{v}^{(\ell)}\right) = 1$.

We conclude that the valuation vectors $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(N)}$ that are shattered by the set $\mathcal{U}$ of social welfare functions with witnesses $z_1 = \cdots = z_N = \frac{1}{2}$. □

Theorem 5.2 implies that the pseudo-dimension upper bound from Lemma 5.1 is tight up to logarithmic factors.

6 SUBSUMPTION OF PRIOR RESEARCH ON GENERALIZATION GUARANTEES

Theorem 3.3 also recovers existing guarantees for data-driven algorithm design. In all of these cases, Theorem 3.3 implies generalization guarantees that match the existing bounds, but in many cases, our approach provides a more succinct proof.

(1) In the full version [7], we analyze several parameterized clustering algorithms [10], which have piecewise-constant dual functions. These algorithms first run a linkage routine which builds a hierarchical tree of clusters. The parameters interpolate between the popular single, average, and complete linkage. The linkage routine is followed by a dynamic programming procedure that returns a clustering corresponding to a pruning of the hierarchical tree.
(2) In the full version [7], we analyze two integer programming algorithms, which have piecewise-constant and piecewise-inverse-quadratic dual functions (as in Figure 3c). The first is branch-and-bound, which is used by commercial solvers such as CPLEX. Branch-and-bound always finds an optimal solution and its parameters control runtime and memory usage. We also study semidefinite programming approximation algorithms for integer quadratic programming. We analyze a parameterized algorithm introduced by Feige and Langberg [28] which includes the Goemans-Williamson algorithm [32] as a special case. We recover previous generalization bounds in both settings [8, 10].

(3) Gupta and Roughgarden [38] introduced parameterized greedy algorithms for the knapsack and maximum weight independent set problems, which we show have piecewise-constant dual functions. We recover their generalization bounds in the full version [7].
(4) We provide generalization bounds for parameterized selling mechanisms when the goal is to maximize revenue, which have piecewise-linear dual functions (as in Figure 3b). A long line of research has studied revenue maximization via machine learning [6, 11, 18, 21, 23, 25, 33, 34, 37, 50, 51, 57, 58, 67]. In the full version [7], we recover Balcan, Sandholm, and Vitercik's generalization bounds [12] which apply to a variety of pricing, auction, and lottery mechanisms. They proved new bounds for mechanism classes not previously studied in the sample-based mechanism design literature and matched or improved over the best known guarantees for many classes.

7 CONCLUSIONS

We provided a general sample complexity theorem for learning high-performing algorithm configurations. Our bound applies whenever a parameterized algorithm's performance is a piecewise-structured function of its parameters: for any fixed problem instance, boundary functions partition the parameters into regions where performance is a well-structured function. We proved this guarantee by exploiting intricate connections between primal function classes (measuring the algorithm's performance as a function of its input) and dual function classes (measuring the algorithm's performance on a fixed input as a function of its parameters). We demonstrated that many parameterized algorithms exhibit this structure and thus our main theorem implies sample complexity guarantees for a broad array of algorithms and application domains.

A great direction for future research is to build on these ideas for the sake of learning a *portfolio* of configurations, rather than a single high-performing configuration. At runtime, machine learning is used to determine which configuration in the portfolio to employ for the given input. Gupta and Roughgarden [38] and Balcan et al. [15] have provided initial results in this direction, but a general theory of portfolio-based algorithm configuration is yet to be developed.

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, CCF-1535967, CCF-1910321, and SES-1919453 to M.B., IIS-1718457, IIS-1617590, and CCF-1733556 to T.S., and DBI-1937540 to C.K.), the US Army Research Office (W911NF-17-1-0082 and W911NF2010081 to T.S.), the Defense Advanced Research Projects Agency under cooperative agreement HR00112020003 to M.B., an AWS Machine Learning Research Award to M.B., an Amazon Research Award to M.B., a Microsoft Research Faculty Fellowship to M.B., a Bloomberg Research Grant to M.B., a fellowship from Carnegie Mellon University's Center for Machine Learning and Health to E.V., and by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program.

REFERENCES

[1] Daniel Alabi, Adam Tauman Kalai, Katrina Ligett, Cameron Musco, Christos Tzamos, and Ellen Vitercik. 2019. Learning to Prune: Speeding up Repeated Computations. In *Conference on Learning Theory (COLT)*.
[2] Patrick Assouad. 1983. Densité et dimension. *Annales de l'Institut Fourier* 33, 3 (1983), 233–282.
[3] Maria-Florina Balcan, Travis Dick, and Manuel Lang. 2020. Learning to Link. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
[4] Maria-Florina Balcan, Travis Dick, and Wesley Pegden. 2020. Semi-bandit Optimization in the Dispersed Setting. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.
[5] Maria-Florina Balcan. 2020. Data-Driven Algorithm Design. In *Beyond Worst Case Analysis of Algorithms*, Tim Roughgarden (Ed.). Cambridge University Press.
[6] Maria-Florina Balcan, Avrim Blum, Jason D Hartline, and Yishay Mansour. 2005. Mechanism design via machine learning. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*. 605–614.
[7] Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. 2021. How Much Data Is Sufficient to Learn High-performing Algorithms? Generalization Guarantees for Data-driven Algorithm Design. *arXiv preprint arXiv:1908.02894* (2021).
[8] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. 2018. Learning to Branch. *International Conference on Machine Learning (ICML)* (2018).
[9] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. 2018. Dispersion for Data-Driven Algorithm Design, Online Learning, and Private Optimization. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*.
[10] Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. 2017. Learning-Theoretic Foundations of Algorithm Configuration for Combinatorial Partitioning Problems. *Conference on Learning Theory (COLT)* (2017).
[11] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. 2016. Sample Complexity of Automated Mechanism Design. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*.
[12] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. 2018. A General Theory of Sample Complexity for Multi-Item Profit Maximization. In *Proceedings of the ACM Conference on Economics and Computation (EC)*. Extended abstract. Full version available on arXiv with the same title.
[13] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. 2020. Learning to Optimize Computational Resources: Frugal Training with Generalization Guarantees. *AAAI Conference on Artificial Intelligence (AAAI)* (2020).
[14] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. 2020. Refined Bounds for Algorithm Configuration: The Knife-edge of Dual Class Approximability. In *International Conference on Machine Learning (ICML)*.
[15] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. 2021. Generalization in Portfolio-based Algorithm Selection. In *AAAI Conference on Artificial Intelligence (AAAI)*.
[16] Jon Louis Bentley, David S Johnson, Frank Thomson Leighton, Catherine C McGeoch, and Lyle A McGeoch. 1984. Some unexpected expected behavior results for bin packing. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*. 279–288.
[17] Avrim Blum, Chen Dan, and Saeed Seddighin. 2020. Learning Complexity of Simulated Annealing. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
[18] Yang Cai and Constantinos Daskalakis. 2017. Learning Multi-item Auctions with (or without) Samples. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*.
[19] Shuchi Chawla, Evangelia Gergatsouli, Yifeng Teng, Christos Tzamos, and Ruimin Zhang. 2020. Pandora's Box with Correlations: Learning and Approximation. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*.
[20] Ed H. Clarke. 1971. Multipart pricing of public goods. *Public Choice* 11 (1971), 17–33.
[21] Richard Cole and Tim Roughgarden. 2014. The sample complexity of revenue maximization. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*.
[22] Dan DeBlasio and John D Kececioglu. 2018. *Parameter Advising for Multiple Sequence Alignment*. Springer.
[23] Nikhil R Devanur, Zhiyi Huang, and Christos-Alexandros Psomas. 2016. The Sample Complexity of Auctions with Side Information. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*.
[24] Robert C Edgar. 2010. Quality measures for protein alignment benchmarks. *Nucleic acids research* 38, 7 (2010), 2145–2153.
[25] Edith Elkind. 2007. Designing and learning optimal finite support auctions. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.
[26] Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. 2020. Learning Augmented Energy Minimization via Speed Scaling. In *Proceedings of*

the Annual Conference on Neural Information Processing Systems (NeurIPS).

[27] Étienne Bamas, Andreas Maggiori, and Ola Svensson. 2020. The Primal-Dual method for Learning Augmented Algorithms. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS).*

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

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

[30] Darya Filippova, Rob Patro, Geet Duggal, and Carl Kingsford. 2014. Identification of alternative topological domains in chromatin. *Algorithms for Molecular Biology* 9 (May 2014), 14. Issue 1.

[31] Vikas Garg and Adam Kalai. 2018. Supervising Unsupervised Learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS).*

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

[33] Yannai A Gonczarowski and Noam Nisan. 2017. Efficient empirical revenue maximization in single-parameter auction environments. In *Proceedings of the Annual Symposium on Theory of Computing (STOC).* 856–868.

[34] Yannai A Gonczarowski and S Matthew Weinberg. 2018. The Sample Complexity of Up-to-ε Multi-Dimensional Revenue Maximization. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS).*

[35] Osamu Gotoh. 1982. An improved algorithm for matching biological sequences. *Journal of Molecular Biology* 162, 3 (1982), 705 – 708.

[36] Theodore Groves. 1973. Incentives in Teams. *Econometrica* 41 (1973), 617–631.

[37] Chenghao Guo, Zhiyi Huang, and Xinzhi Zhang. 2019. Settling the sample complexity of single-parameter revenue maximization. *Proceedings of the Annual Symposium on Theory of Computing (STOC)* (2019).

[38] Rishi Gupta and Tim Roughgarden. 2017. A PAC approach to application-specific algorithm selection. *SIAM J. Comput.* 46, 3 (2017), 992–1017.

[39] Dan Gusfield, Krishnan Balasubramanian, and Dalit Naor. 1994. Parametric optimization of sequence alignment. *Algorithmica* 12, 4-5 (1994), 312–326.

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

[41] Eric Horvitz, Yongshao Ruan, Carla Gomez, Henry Kautz, Bart Selman, and Max Chickering. 2001. A Bayesian Approach to Tackling Hard Computational Problems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI).*

[42] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. 2019. Learning-based frequency estimation algorithms. In *Proceedings of the International Conference on Learning Representations (ICLR).*

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

[44] Raj Iyer, David Karger, Hariharan Rahul, and Mikkel Thorup. 2002. An Experimental Study of Polylogarithmic, Fully Dynamic, Connectivity Algorithms. *ACM Journal of Experimental Algorithmics* 6 (Dec. 2002), 4–es.

[45] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. 2010. ISAC-Instance-Specific Algorithm Configuration.. In *Proceedings of the European Conference on Artificial Intelligence (ECAI).*

[46] Robert Kleinberg, Kevin Leyton-Brown, and Brendan Lucier. 2017. Efficiency Through Procrastination: Approximately Optimal Algorithm Configuration with Runtime Guarantees. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

[47] Robert Kleinberg, Kevin Leyton-Brown, Brendan Lucier, and Devon Graham. 2019. Procrastinating with Confidence: Near-Optimal, Anytime, Adaptive Algorithm Configuration. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)* (2019).

[48] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. 2009. Empirical hardness models: Methodology and a case study on combinatorial auctions. *J. ACM* 56, 4 (2009), 1–52.

[49] 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. 2009. Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome. *Science* 326, 5950 (2009), 289–293. https://doi.org/10.1126/science.1181369

[50] Anton Likhodedov and Tuomas Sandholm. 2004. Methods for Boosting Revenue in Combinatorial Auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI).* San Jose, CA, 232–237.

[51] Anton Likhodedov and Tuomas Sandholm. 2005. Approximating Revenue-Maximizing Combinatorial Auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI).* Pittsburgh, PA.

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

[53] Thodoris Lykouris and Sergei Vassilvitskii. 2018. Competitive caching with machine learned advice. In *International Conference on Machine Learning (ICML).*

[54] Catherine C McGeoch. 2012. *A guide to experimental algorithmics.* Cambridge University Press.

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

[56] Michael Mitzenmacher. 2018. A model for learned bloom filters and optimizing by sandwiching. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS).* 464–473.

[57] Mehryar Mohri and Andrés Muñoz. 2014. Learning Theory and Algorithms for revenue optimization in second price auctions with reserve. In *International Conference on Machine Learning (ICML).*

[58] Jamie Morgenstern and Tim Roughgarden. 2016. Learning Simple Auctions. In *Conference on Learning Theory (COLT).*

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

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

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

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

[63] David Pollard. 1984. *Convergence of Stochastic Processes.* Springer.

[64] Manish Purohit, Zoya Svitkina, and Ravi Kumar. 2018. Improving online algorithms via ML predictions. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS).* 9661–9670.

[65] Kevin Roberts. 1979. The characterization of implementable social choice rules. In *Aggregation and Revelation of Preferences*, J-J Laffont (Ed.). North-Holland Publishing Company.

[66] Tuomas Sandholm. 2013. Very-Large-Scale Generalized Combinatorial Multi-Attribute Auctions: Lessons from Conducting $60 Billion of Sourcing. In *Handbook of Market Design*, Zvika Neeman, Alvin Roth, and Nir Vulkan (Eds.). Oxford University Press.

[67] Tuomas Sandholm and Anton Likhodedov. 2015. Automated Design of Revenue-Maximizing Combinatorial Auctions. *Operations Research* 63, 5 (2015), 1000–1025. Special issue on Computational Economics. Subsumes and extends over a AAAI-05 paper and a AAAI-04 paper.

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

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

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

[71] William Vickrey. 1961. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance* 16 (1961), 8–37.

[72] Michael S Waterman, Temple F Smith, and William A Beyer. 1976. Some biological sequence metrics. *Advances in Mathematics* 20, 3 (1976), 367–387.

[73] Alexander Wei and Fred Zhang. 2020. Optimal Robustness-Consistency Tradeoffs for Learning-Augmented Online Algorithms. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS).*

[74] Gellért Weisz, András György, and Csaba Szepesvári. 2018. LeapsAndBounds: A Method for Approximately Optimal Algorithm Configuration. In *International Conference on Machine Learning (ICML).*

[75] Gellért Weisz, András György, and Csaba Szepesvári. 2019. CapsAndRuns: An improved method for approximately optimal algorithm configuration. In *International Conference on Machine Learning (ICML).*

[76] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2008. SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research* 32, 1 (2008), 565–606.

[77] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Hydra-MIP: Automated algorithm configuration and selection for mixed integer programming. In *RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion at the International Joint Conference on Artificial Intelligence (IJCAI).*