Semi-bandit Optimization in the Dispersed Setting

Maria-Florina Balcan
Carnegie Mellon University
ninamf@cs.cmu.edu

Travis Dick
University of Pennsylvania
tbd@seas.upenn.edu

Wesley Pegden
Carnegie Mellon University
wes@math.cmu.edu

Abstract

The goal of data-driven algorithm design is to obtain high-performing algorithms for specific application domains using machine learning and data. Across many fields in AI, science, and engineering, practitioners will often fix a family of parameterized algorithms and then optimize those parameters to obtain good performance on example instances from the application domain. In the online setting, we must choose algorithm parameters for each instance as they arrive, and our goal is to be competitive with the best fixed algorithm in hindsight.

There are two major challenges in online data-driven algorithm design. First, it can be computationally expensive to evaluate the loss functions that map algorithm parameters to performance, which often require the learner to run a combinatorial algorithm to measure its performance. Second, the losses can be extremely volatile and have sharp discontinuities. However, we show that in many applications, evaluating the loss function for one algorithm choice can sometimes reveal the loss for a range of similar algorithms, essentially for free. We develop online optimization algorithms capable of using this kind of extra information by working in the semi-bandit feedback setting. Our algorithms achieve regret bounds that are essentially as good as algorithms under full-information feedback and are significantly more computationally efficient. We apply our semi-bandit results to obtain the first provable guarantees for data-driven algorithm design for linkage-based clustering and we improve the best regret bounds for designing greedy knapsack algorithms.

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

1 INTRODUCTION

Overview. This paper concerns data-driven algorithm design for combinatorial settings, which is an important area at the intersection of machine learning and computing that has been long of interest to the AI community [23, 41, 29, 18]. However, until recently, most algorithm design procedures did not have any provable guarantees on their performance, especially in the realistic online scenario. The first general online data-driven algorithm design procedures with regret bounds were given by Balcan et al. [11], who studied the problem under full-information and bandit feedback regimes. We develop efficient semi-bandit algorithms that achieve nearly the same regret as their full information algorithms, while being as efficient as their bandit algorithms.

The goal of data-driven algorithm design is to use machine learning and data to decide what algorithm to use from a large (typically parametrized) family of algorithms for a given problem domain. For example, we may want to decide which clustering algorithm to use from a large family of clustering procedures in order to obtain the highest quality results. We are concerned with the online setting, where at each round the *learner* chooses an algorithm from the family and receives a new instance of the problem. The problem is characterized by a loss function that measures the performance of each algorithm in the family for the given instance, and the goal is to select algorithms so that the cumulative performance of the learner is nearly as good as the best algorithm in hindsight for that sequence of problems.

The major challenge in these settings is that it is potentially computationally expensive for the learner to characterize the loss function for each round, since each run of the algorithm reveals the value of the loss function for just the selected parameters. Moreover, for combinatorial problems, small differences between two algorithms can lead to a cascade of changes in their behavior and significantly change their performance. However, when the algorithm family is parameterized, it can often be shown that the losses—though highly nonconvex in general—are at least piecewise Lipschitz in the algorithm parameters, so we can phrase the problem as online optimization of piecewise Lipschitz functions.

Prior work on piecewise Lipschitz optimization was limited to two extreme feedback regimes: Either the learner carries out a computationally expensive process to obtain full-information feedback (i.e., it observes the loss of every algorithm in the family on each instance), or accepts suboptimal regret bounds to work in the bandit feedback setting (i.e., it only observes the loss of one algorithm for each instance). This creates a tradeoff between computational efficiency and good regret bounds. However, many data-driven algorithm design problems exhibit rich additional structure that is ignored by these two approaches. We show that, surprisingly, evaluating the loss function for a single algorithm can sometimes reveal the loss for a range of similar algorithms, essentially for free; in the context of the loss function, we show that an entire Lipschitz region can often be learned at once. This motivates us to define a new learning model, which we call the semi-bandit feedback setting for learning piecewise Lispchitz functions. Our new results in this model achieve the best of both worlds: we can efficiently obtain the necessary feedback while also having regret bounds that are nearly as good as under full-information.

We instantiate our results for data-driven algorithm design on two combinatorial problems. These are machine learning problems where the goal is to learn an optimal algorithm, rather than a prediction rule. Our results for optimizing over a family of greedy knapsack algorithms improve over the procedures of Balcan et al. [11], Gupta and Roughgarden [21], and Cohen-Addad and Kanade [16] by simultaneously being more efficient and having tighter regret bounds. We also provide the first online data-driven algorithm design procedures for a rich family of linkage based clustering algorithms introduced by Balcan et al. [9] that interpolates between single and complete linkage, which are algorithms that are widely used in practice [6, 34, 40] and known to perform optimally in many settings [5, 8, 7, 20]. Balcan et al. [9] consider the data-driven algorithm design problem for this family of algorithms in the batch setting, rather than the online setting, where they model the application domain as a distribution over problem instances, the goal is to find the algorithm with the highest expected performance given an i.i.d. sample from the distribution as training data.

Problem Setup. We study the problem of online piecewise Lipschitz optimization. The learning protocol is as follows: on each round t, the learner chooses a parameter ρ_t belonging to a d-dimensional parameter space $\mathcal{C} \subset \mathbb{R}^d$, the adversary chooses a piecewise Lipschitz loss function $\ell_t : \mathcal{C} \to [0,1]$, and the learner incurs a loss equal to $\ell_t(\rho_t)$. A function $\ell_t : \mathcal{C} \to [0,1]$ is piecewise L-Lipschitz if we can partition the parameter space $\mathcal{C}$ into regions such that ℓ_t is L-Lipschitz when restricted to each region. Many important instances of data-driven algorithm design require optimizing piecewise Lipschitz functions, including greedy combinatorial algorithms [21], clustering algorithms and SDP-rounding schemes [9], branch and bound mixed integer program solvers [10], initialization procedures for k-means clustering [12], and various auction design problems [13]. In these problems, the family of algorithms is parameterized and each parameter $\rho \in \mathcal{C}$ corresponds to one algorithm. We suppose that on each round t there is a partition $A_1^{(t)}, \dots, A_M^{(t)}$ of the parameter space $\mathcal{C}$, called the feedback system. If the learner's parameter ρ_t belongs to the set $A_i^{(t)}$, then they observe both the set $A_i^{(t)}$ as well as the loss $\ell_t(\rho)$ for every $\rho \in A_i^{(t)}$. We consider the uninformed setting, where the learner does not know the feedback system for round t in advance of selecting a parameter. For simplicity, we consider oblivious adversaries that choose their sequence of loss functions $\ell_1, \ell_2, \dots$ adversarially, but before the interaction with the learner begins. The learner's goal is to minimize regret, which is the difference between their total accumulated loss and that of the best parameter in hindsight: $\sum_{t=1}^T \ell_t(\rho_t) - \min_{\rho \in \mathcal{C}} \sum_{t=1}^T \ell_t(\rho)$.

Throughout the paper, we use the notation $\tilde{O}(\cdot)$ to optionally suppress all logarithmic terms and dependence on parameters other than the time horizon T and the dimension of the parameter space d.

Main Results and Techniques.
Semi-bandit Regret Bounds in the Dispersed Setting. It is not always possible to achieve sub-linear regret for piecewise Lipschitz loss functions [30, 14, 32]. Balcan et al. [11] provide regret bounds in the full-information and bandit feedback settings under a dispersion condition that roughly measures the number of discontinuous functions in any ball of a given radius, and which is satisfied for a diverse collection of combinatorial algorithm configuration problems. In this paper, we introduce a related and more general version of this condition that captures what is asymptotically important for our regret bounds.

Definition 1. The sequence of loss functions $\ell_1, \ell_2, \dots$ is *β-point-dispersed* for the Lipschitz constant L if for all T and for all $\epsilon \geq T^{-\beta}$, we have that, in expectation, the maximum number of functions among $\ell_1, \dots, \ell_T$ that fail the L-Lipschitz condition for any pair of points at distance ϵ in $\mathcal{C}$ is at most $\tilde{O}(\epsilon T)$. That is, for all T and for all $\epsilon \geq T^{-\beta}$, we have $\mathbb{E}\big[\max_{\rho,\rho'} \big|\{t \in [T] : |\ell_t(\rho) - \ell_t(\rho')| > L\|\rho - \rho'\|_2\}\big|\big] = \tilde{O}(\epsilon T)$. where the max is taken over all $\rho, \rho' \in \mathcal{C} : \|\rho - \rho'\|_2 \leq \epsilon$.

Note that the righthandside $\tilde{O}(\epsilon T)$ is roughly the number L-Lipschitz failures one would expect across T functions for a pair of points at distance ϵ if Lipschitz failures are distributed reasonably randomly, and their probability of occuring between a pair of points at distance ϵ is roughly proportional to ϵ. The definition of β-dispersion measures how small ϵ can be while maintaining the correctness of this rough bound for the loss functions ℓ_i.

In our applications, the sequence of loss functions will be chosen by a smoothed adversary, in the sense of Spielman and Teng [36]. Informally, the discontinuity locations of the functions chosen by a smoothed adversary are randomly perturbed. The expectation in Definition 1 is over this randomness in the sequence of loss functions. (Balcan et al. [11] also show examples where sufficient randomness can arise from the algorithm itself, rather than smoothness constraints on the adversary.) In all of our applications, we prove β-dispersion with $\beta = 1/2$. We provide an algorithm for online piecewise Lipschitz optimization under semi-bandit feedback whose regret is characterized by the β-dispersion parameter of the losses. In Section 2, we prove the following result:

Theorem 2. *Let $\mathcal{C} \subset \mathbb{R}^d$ be a bounded parameter space and $\ell_1, \ell_2, \dots : \mathcal{C} \to [0,1]$ be piecewise Lipschitz functions that are β-point-dispersed. Running the continuous Exp3-SET algorithm (Algorithm 1) under semi-bandit feedback with an appropriate parameter λ has expected regret bounded by $\mathbb{E}\big[\sum_{t=1}^T \ell_t(\rho_t) - \ell_t(\rho^*)\big] \le \tilde{O}\big(\sqrt{dT} + T^{1-\beta}\big)$.*

In comparison, the bandit-feedback algorithm of Balcan et al. [11] has expected regret bounded by $\tilde{O}(dT^{\frac{d+1}{d+2}}3^d + T^{1-\beta})$. Even in one-dimensional problems, this bound is $\tilde{O}(T^{2/3} + T^{1-\beta})$, which is worse than our results. Under different assumptions, the bandit algorithm of Cohen-Addad and Kanade [16] has $\tilde{O}(T^{2/3})$ regret for the special case of one-dimensional piecewise constant functions.

General Tools for Verifying Dispersion. We also provide general tools for proving that a sequence of piecewise Lipschitz functions satisfies dispersion. When the sequence $\ell_1, \ell_2, \dots$ is random, we can usually directly bound the expected number of loss functions that are not L-Lipschitz between any fixed pair of points ρ and ρ' with $\|\rho - \rho'\|_2 \le \epsilon$ by $\tilde{O}(T\epsilon)$. However, this does not imply that the functions are β-point-dispersed, since the expected number of non-Lipschitz functions between the *worst* pair of points at distance ϵ will typically be larger than the expected number for any fixed pair. Building on uniform convergence from learning theory [35], we show that if each loss function has a one-dimensional domain, at most K discontinuities and any interval of radius ϵ has at most $\tilde{O}(T\epsilon)$ non-Lipschitz functions in expectation, then the expected number of non-Lipschitz losses on the worst interval of length ϵ is at most $\tilde{O}(T\epsilon + \sqrt{T\log(TK)})$. This implies that for all pairs of points at distance ϵ, at most $O(T\epsilon + \sqrt{T\log(TK)})$ functions are non-Lipschitz between them and demonstrates β-dispersion with $\beta = 1/2$. Our result gives an exponential improvement in the dependence on K compared to the results of Balcan et al. [11], who upper bound the expected number of non-Lipschitz losses in the worst interval of length ϵ by $\tilde{O}(TK\epsilon + K\sqrt{T\log(TK)})$.

Semi-bandit Online Data-driven Algorithm Design. In Section 4, we combine our general regret analysis from Theorem 2 together with application-specific dispersion analysis to obtain practical data-driven algorithm design procedures for linkage-based clustering and the knapsack problem. In both applications, we show that the discontinuities of each loss function are the roots of polynomials depending on the corresponding problem instance, and that the roots are dispersed under mild smoothness assumptions on the adversary. We obtain the first online data-driven algorithm design procedures for linkage based clustering, and algorithm design procedures for the knapsack problem with substantial computational improvements over the prior work, while at the same time achieving nearly the same regret bound.

Explicit Comparison for Knapsack. To highlight the benefits of our new learning model and results applied to data-driven algorithm design, we give an explicit comparison of the computational complexity for obtaining different types of feedback and the corresponding regret bounds for the family of greedy knapsack algorithms introduced in Section 4.1. In each round of the online game, the algorithm chooses a parameter ρ, a new knapsack instance with n items arrives, and our goal is for the total value of items selected by the learner to be close to the total value of the best fixed parameter ρ in hindsight. We compare our results to the best prior full-information and bandit feedback procedures.

- *Full-information.* Balcan et al. [11] show that the exponentially weighted forecaster with full-information feedback achieves a regret bound of $\tilde{O}(n^2\sqrt{T})$. Our tighter analysis improves the bound to $\tilde{O}(\sqrt{T})$. Obtaining full-information feedback has a total cost of $O(n^3 \log n)$ time per round.
- *Bandit Feedback.* The discretization-based bandit algorithm of Balcan et al. [11] has regret $\tilde{O}(T^{2/3}n^2)$, but only requires $O(n \log n)$ time per round.
- *Semi-bandit Feedback.* In this paper we give an algorithm whose regret is $\tilde{O}(n\sqrt{T})$ using semi-bandit feedback obtainable in time $O(n \log n)$ per round. Note that our algorithm is as efficient as the bandit-feedback algorithm, yet its regret is only larger by a factor of n.

Related Work. There is a rich literature on data-driven algorithm design. Most prior work focuses on the statistical setting, where the learner is given a large iid sample of problem instances from some distribution, and the goal is to find the algorithm with the best performance in expectation. Gupta and Roughgarden [21] introduced this formal setting and provide sample complexity results for several families of greedy algorithms. Balcan et al. [9] consider semidefinite rounding schemes for integer quadratic programs and linkage based clustering algorithms, Balcan et al. [10] consider learning the best branch and bound algorithms for mixed integer programs, and Balcan et al. [12] consider learning the best initialization procedures for k-means clustering. Aamand et al. [1] and Hsu et al. [24] use learned algorithms for streaming frequency estimation. Indyk et al. [25] study the problem of using a learned sketching matrix to improve low-rank approximation algorithms. Dong et al. [19] use learned space partitions to improve nearest neighbor search. In addition to these formal results, this statistical setting has been the predominant model for data-driven algorithm configuration in artificial intelligence [33], combinatorial auctions [28], numerical linear algebra [17], vehicle routing [15], and SAT solving [41].

Another related line of work focuses on the problem of choosing the algorithm with the shortest running time over a distribution of problem instances [27, 39, 38]. This work makes minimal assumptions about the algorithm family and instead designs procedures that can avoid running every algorithm to completion, since this may be very expensive. Our work, on the other hand, explores special structure in algorithm families and can be used to optimize more general performance measures in the online rather than stochastic setting.

For online optimization of one-dimensional piecewise constant functions, Cohen-Addad and Kanade [16] provide full-information and bandit online optimization procedures. Balcan et al. [11] consider the more general setting of multi-dimensional piecewise Lipschitz functions. They introduce a dispersion condition that roughly measures how many functions are not Lipschitz in any ball, and provide algorithms with dispersion-dependent full-information and bandit regret bounds. They also verify that dispersion is satisfied for a diverse collection of data-driven algorithm design problems.

Prior work on semi-bandit feedback has focused predominantly on finite-armed bandits. Semi-bandit feedback was first considered for online shortest path problems, where on each round the learner selects a path through a graph and observes the length of the edges along that path (but not for other edges) [22, 26]. Audibert et al. [3] obtain minimax bounds for a generalization to combinatorial bandits, where the learner's action space is described by boolean vectors in $\{0,1\}^d$, the losses are linear, and the on each round the learner observes the entries of the loss vector corresponding to the non-zero entries in their action. Alon et al. [2] introduce the Exp3-SET algorithm for semi-bandit feedback for finite-armed bandits. They consider the graph-feedback setting introduced by Mannor and Shamir [31], where on each round t, there is a feedback graph G_t over the arms of the bandit and playing arm i reveals the loss for arm i and all arms adjacent in the graph G_t. We extend the Exp3-SET algorithm to online optimization problems where there are infinitely many arms and where the feedback system on each round is a partition of the parameter space $\mathcal{C}$.

2 SEMI-BANDIT OPTIMIZATION OF PIECEWISE LIPSCHITZ LOSSES

In this section we provide an algorithm for online piecewise Lispchitz optimization and analyze its regret under dispersion. Our results are for the following continuous semi-bandit setting.

Definition 3 (Uninformed Semi-bandit Feedback.)**.** An online optimization problem with loss functions $\ell_1, \ell_2, \dots$ has semi-bandit feedback if for each time t, there is partition $A_1^{(t)}, \dots, A_M^{(t)}$ of the parameter space $\mathcal{C}$, called a feedback system, such that when the learner plays point $\rho_t \in A_i^{(t)}$, they observe the set $A_i^{(t)}$ and $\ell_t(\rho)$ for all $\rho \in A_i^{(t)}$. For any $\rho \in \mathcal{C}$, we let $A^{(t)}(\rho)$ denote the feedback set that contains ρ.

We analyze a continuous version of the Exp3-SET algorithm of Alon et al. [2]. This algorithm uses importance weighting to construct unbiased estimates of the complete loss function on each round, which it passes as input to a continuous version of the exponentially weighted forecaster. Pseudocode is given in Algorithm 1. Unlike the Exp3 algorithm of Auer et al. [4], the Exp3-SET algorithm and our continuous version do not include an explicit exploration term (i.e., we do not mix the distribution p_t with a uniform distribution over $\mathcal{C}$). Stoltz [37] was the first to show that mixing with the uniform distribution is unnecessary for the Exp3 algorithm to have optimal expected regret.

In Appendix A.2, we show how to implement this algorithm with $O(\log T)$ per round time complexity for one dimensional piecewise constant losses using the interval tree data structure of Cohen-Addad and Kanade [16].

Given the learner's observations on round t, Algorithm 1 uses importance weighting to estimate the complete loss function by $\hat{\ell}_t(\rho) = \frac{\mathbb{I}\{\rho \in A^{(t)}(\rho_t)\}}{p_t(A^{(t)}(\rho_t))} \ell_t(\rho)$. The estimate $\hat{\ell}_t(\rho)$ is only non-zero for parameters ρ that belong to the feedback set observed by the algorithm at round

Algorithm 1 Continuous Exp3-SET

```
Parameter: Step size $\lambda \in [0, 1]$
1. Let $w_1(\rho) = 1$ for all $\rho \in \mathcal{C}$
2. For $t = 1, \dots, T$
    (a) Let $p_t(\rho) = \frac{w_t(\rho)}{W_t}$, where $W_t = \int_{\mathcal{C}} w_t(\rho)\, d\rho$.
    (b) Sample $\rho_t$ from $p_t$, play it, and observe feedback set $A^{(t)}(\rho)$ and losses $\ell_t(\rho)$ for all $\rho \in A_t$.
    (c) Let $\hat{\ell}_t(\rho) = \frac{\mathbb{I}\{\rho \in A^{(t)}(\rho_t)\}}{p_t(A^{(t)}(\rho_t))} \ell_t(\rho)$, where we define $p_t(A^{(t)}(\rho_t)) = \int_{A^{(t)}(\rho_t)} p_t(\rho)\, d\rho$.
    (d) Let $w_{t+1}(\rho) = w_t(\rho) \exp(-\lambda \hat{\ell}_t(\rho))$ for all $\rho$.
```

t. The key property of $\hat{\ell}_t$ is that it is an unbiased estimate of the true loss function conditioned on the history until the beginning of round t. More formally, let $\mathbb{E}_t[\cdot] = \mathbb{E}[\cdot | \rho_1, \dots, \rho_{t-1}, \ell_1, \dots, \ell_t]$ denote the conditional expectation given the learner's choices until round $t-1$ and the first t loss functions. This expectation is only over the randomness of the learner's choice of ρ_t at time t. For clarity, we also use the notation $\mathbb{E}_{<t}[\cdot]$ to denote the expectation of any random variable that is a function of only $\rho_1, \dots, \rho_{t-1}$ and $\ell_1, \dots, \ell_t$ so that for any random quantity X, we have $\mathbb{E}[X] = \mathbb{E}_{<t}\big[\mathbb{E}_t[X]\big]$. For any $\rho \in \mathcal{C}$ and t, a straight forward calculation shows that $\mathbb{E}_t[\hat{\ell}_t(\rho)] = \ell_t(\rho)$.

To simplify presentation, we assume that the sequence of loss functions has an r_0-*interior minimizer*: with probability one, for all times T there exists $\rho^* \in \operatorname{argmin}_{\mathcal{C}} \sum_{t=1}^T \ell_t(\rho)$ such that $B(\rho^*, r_0) \subset \mathcal{C}$. We can usually modify a sequence of loss functions to obtain an equivalent optimization problem that is guaranteed to have an r_0-interior minimizer. In Appendix A we discuss such a transformation that works whenever the parameter space $\mathcal{C}$ is convex (with no condition on the losses).

We bound the regret of Algorithm 1 under a slightly more precise version of β-point-dispersion which leads to more precise bounds and broader applicability.

Definition 4. The sequence of loss functions $\ell_1, \ell_2, \dots$ is f-*point-dispersed* for the Lipschitz constant L and dispersion function $f : \mathbb{N} \times [0, \infty) \to \mathbb{R}$ if for all T and for all $\epsilon > 0$, we have $\mathbb{E}\big[\max_{\rho,\rho'} \big|\{t \in [T] : |\ell_t(\rho) - \ell_t(\rho')| > L\|\rho - \rho'\|_2\}\big|\big] \le f(T, \epsilon)$. where the max is taken over all $\rho, \rho' \in \mathcal{C} : \|\rho - \rho'\|_2 \le \epsilon$.

We can express both β-point-dispersion and (w, k)-dispersion from Balcan et al. [11] in terms of f-point-dispersion. For any $T \in \mathbb{N}$ and $\epsilon > 0$, let $D(T, \epsilon) = \mathbb{E}[\max_{\|\rho - \rho'\|_2 \le \epsilon} \big|\{1 \le t \le T : |\ell_t(\rho) - \ell_t(\rho')| \ge L\|\rho - \rho'\|_2\}\big|$ be the expected number of non-Lipschitz functions among $\ell_1, \dots, \ell_T$ across the worst pair of points within distance at most ϵ. If the loss functions are β-point-dispersed, then we know that for all T and $\epsilon \ge T^{-\beta}$, we have $D(T, \epsilon) = \tilde{O}(T\epsilon)$. Since $D(T, \epsilon)$ is a non-decreasing function of the distance ϵ, we are guaranteed that for any $\epsilon < T^{-\beta}$ we have $D(T, \epsilon) \le D(T, T^{-\beta}) = \tilde{O}(T^{1-\beta})$. It follows that the functions are also f-point-dispersed for $f(T, \epsilon) = \tilde{O}(T\epsilon + T^{1-\beta})$. Similarly, the functions are (w, k)-dispersed if every ball of radius w in $\mathcal{C}$ has at most k non-Lipschitz functions. Since any pair of points within distance ϵ are contained in a ball of radius ϵ, it follows that for $\epsilon \le w$ we ahve $D(T, \epsilon) \le k$, but for $\epsilon > w$ we could have $D(T, \epsilon)$ as large as T. It follows that the functions are f-point-dispersed where $f(T, \epsilon) = k$ for all $\epsilon < w$ and $f(T, \epsilon) = T$ otherwise.

We bound the regret of Algorithm 1 in terms of the f-point-dispersion of the losses. The proof is given in Appendix A.

Theorem 5. *Let $\mathcal{C} \subset \mathbb{R}^d$ be contained in a ball of radius R and $\ell_1, \ell_2, \dots : \mathcal{C} \to [0, 1]$ be piecewise L-Lipschitz functions that are f-point-dispersed with an r_0-interior minimizer. Moreover, suppose the learner gets semi-bandit feedback and, on each round t, the feedback system $A_1^{(t)}, \dots, A_M^{(t)}$ has M feedback sets. For any $r \in (0, r_0]$, running Algorithm 1 with $\lambda = \sqrt{d \log(R/r)/(TM)}$ satisfies the following regret bound: $\mathbb{E}\big[\sum_{t=1}^T \ell_t(\rho_t) - \ell_t(\rho^*)\big] \le O\big(\sqrt{dTM \log(R/r)} + f(T, r) + TLr\big)$.*

Our regret bound for β-dispersed losses given in Theorem 2 follows immediately from Theorem 5.

Note that our results are also applicable in two closely related settings: maximizing dispersed piecewise Lipschitz utility functions, and the case when losses are bounded in $[0, H]$ for some known bound H instead of $[0, 1]$. A discussion of the necessary transformations can be found in Appendix A.1.

3 A RECIPE FOR VERIFYING DISPERSION

In this section we illustrate a general recipe for proving dispersion in data-driven algorithm design problems. We work in the framework of smoothed analysis [36] and suppose that nature injects a small amount of randomness into the problem instances chosen by the adversary before the learner sees them. Our goal is to leverage this framework to prove that the loss functions are dispersed.

At a high-level, a general strategy for proving dispersion in this setting which has proved successful across a range of examples is to:

1. Bound the probability density of the random set of discontinuities of the loss functions, to obtain a bound on the typical rate of Lipschitz condition violations.
2. Use a VC-dimension based uniform convergence argument to transform this typical rate into a bound on the

dispersion of the loss functions.

In this section, we give general tools which can be used to accomplish each of these steps in real-world problems.

For many combinatorial algorithm families, the loss function for a given instance is piecewise L-Lipschitz on a partition of $\mathcal{C}$ whose boundaries are defined by the roots of a collection of polynomials. In the smoothed analysis setting, the coefficients of these polynomials have bounded probability density, and may (or may not) be independent. The following theorem translates this randomness in the coefficients into a statement about the randomness of their roots, making it easy to accomplish Step 1 in the strategy above.

Theorem 6. *Consider a random degree d polynomial $\phi(\rho)$ with leading coefficient 1 and subsequent coefficients which are real of absolute value at most R, whose joint density is at most κ. There is an absolute constant K depending only on d and R such that every interval I of length $\leq \epsilon$ satisfies $\Pr(\phi$ has a root in $I) \leq \kappa\epsilon/K$.*

(In Appendix B we prove a generalization of Theorem 6 that allows for less structured coefficient vectors.)

In the 1-dimensional setting (i.e., when optimizing a single-parameter family of algorithms), Theorem 6 often allows us to argue that no interval of width ϵ contains any discontinuity from each loss function with large probability. In the multidimensional setting, the sets of discontinuities of the L-Lipschitz loss functions will often be algebraic curves (or in more than 2 dimensions, algebraic varieties) defined as the zero sets of multivariate polynomials. In this case, Theorem 6 can still be used to accomplish Step 1 of the dispersion strategy, by showing that few zeros are likely to occur on any fixed piecewise-linear path (on whose pieces the zero sets of the multivariate polynomial is the zero set of a single-variable polynomial). In particular, this accomplishes Step 1 of the basic strategy for proving dispersion.

For Step 2, we wish to transform our bound on the typical rate of Lipschitz violations to a uniform bound on the worst number of Lipschitz violations, over all pairs of points ρ, ρ'. For example, the following theorem accomplishes this in the 1-dimensional case:

Theorem 7. *Let $\ell_1, \ell_2, \cdots : \mathbb{R} \to \mathbb{R}$ be independent piecewise L-Lipschitz functions, each having at most K discontinuities. Let $D(T, \epsilon, \rho) = \left|\{1 \leq t \leq T \mid \ell_t \text{ is not } L\text{-Lipschitz on } [\rho - \epsilon, \rho + \epsilon]\}\right|$ be the number of functions in $\ell_1, \ldots, \ell_T$ that are not L-Lipschitz on the ball $[\rho - \epsilon, \rho + \epsilon]$. Then we have $\mathbb{E}[\max_{\rho \in \mathbb{R}} D(T, \epsilon, \rho)] \leq \max_{\rho \in \mathbb{R}} \mathbb{E}[D(T, \epsilon, \rho)] + O(\sqrt{T \log(TK)})$.*

To see the general utility of Theorem 7, observe that if in Step 1 we show that for all times T, radiuses $\epsilon > 0$ and any fixed interval I of radius ϵ, the expected number of non-Lipschitz functions on interval I is at most $\tilde{O}(T\epsilon)$, then Theorem 7 guarantees that the losses are $\frac{1}{2}$-dispersed.

To accomplish Step 2 in the case of higher dimensions with discontinuities given as the 0 sets of (multivariate) polynomials, the 0-sets are now not finite sets but finite-degree algebraic curves (or varieties). To verify dispersion, we need a uniform-convergence bound on the number of Lipschitz failures between the worst pair of points ρ, ρ' at distance $\leq \varepsilon$, but the definition allows us to bound the worst rate of discontinuities along any path between ρ, ρ' of our choice. The following theorem bounds the VC dimension of axis aligned segments against bounded-degree algebraic curves, which will allow us to accomplish Step 2 by considering piecewise axis-aligned paths between points ρ and ρ'.

Theorem 8. *There is a constant K_d (e.g., $K_2 \leq 11$) depending only on d such that axis-aligned line segments cannot shatter any collection of K_d algebraic curves of degree at most d.*

The proof, which appears in the appendix, makes repeated use of Bezout's theorem which bounds the number of intersection points of algebraic curves in terms of their degrees. In particular, a family of k algebraic curves will always a $\text{poly}(k)$-bounded number of intersection points and local extrema, which, one can show, makes it impossible to label the exponentially-many subsets of such curves with axis-aligned segments.

Theorem 8 allows us now to obtain a 2-dimensional analog of Theorem 7 as follows, giving an implementation of Step 2 in this setting.

Theorem 9. *Let $\ell_1, \ell_2, \cdots : \mathbb{R}^2 \to \mathbb{R}$ be independent piecewise L-Lipschitz functions, each having a set of discontinuities specified by a collection of K algebraic curves of bounded degree. Let $\mathcal{L}$ denote the set of axis-aligned line-segments in $\mathbb{R}^2$. For each $s \in \mathcal{L}$, define $D(T, s) = \left|\{1 \leq t \leq T : \ell_t \text{ has a discontinuity along } s\}\right|$. Then we have $\mathbb{E}[\sup_{s \in \mathcal{L}} D(T, s)] \leq \sup_{s \in \mathcal{L}} \mathbb{E}[D(T, s)] + O(\sqrt{T \log(TK)})$.*

4 ONLINE DATA-DRIVEN ALGORITHM DESIGN WITH SEMI-BANDIT FEEDBACK

In this section we apply our semi-bandit optimization results to online data-driven algorithm design for two rich parameterized families of algorithms. For both families, we show how to obtain semi-bandit feedback by running a single algorithm from the family. We also analyze dispersion for these problems under the assumption that the adversary is smoothed. In both cases, we obtain $\tilde{O}(\sqrt{T})$ regret bounds in the semi-bandit feedback setting. Finally, in Appendix C.1 we show how to use binary search to obtain semi-bandit feedback for a large class single-parameter algorithm families.

Smoothed adversaries. We consider adversaries that are smoothed in the sense of Spielman and Teng [36], where their decisions are corrupted by small random perturbations. Formally, we say that a parameter chosen by the adversary is κ-smooth if it is a random variable whose density is bounded by κ. After the adversary chooses the density for each smoothed parameter, nature samples each parameter value independently from their corresponding distributions. Small values of κ correspond to larger random perturbations of the problem parameters, while in the limit as $\kappa \to \infty$, the adversary is able to choose the parameters deterministically. In each application, we will specify which problem parameters are smoothed, together with the bound κ on their density. For simplicity, we assume that all κ-smooth random variables are independent (i.e., the corruption of the adversary's choices is not correlated across variables), though many of our results can be exteneded to allow for some correlation between the parameters of each instance.

4.1 GREEDY ALGORITHMS FOR KNAPSACK

First, we consider selecting the best algorithm from a parameterized family of a greedy algorithms for the knapsack problem. An instance of the knapsack problem consists of n items, where item i has a value v_i and a size s_i, and a knapsack capacity C. Our goal is to find the most valuable subset of items whose total size does not exceed C. Gupta and Roughgarden [21] propose using the following parameterized family of greedy knapsack algorithms: for a given parameter $\rho \in [0, R]$, set the score of item i to be $\sigma_\rho(i) = v_i/s_i^\rho$. Then, in decreasing order of score, add each item to the knapsack if there is enough capacity left. This algorithm runs in time $O(n \log n)$. In our analysis, we assume that the adversary's item values are κ-smooth.

First, we show how to obtain semi-bandit feedback for this family of greedy knapsack algorithms by running a single algorithm in the family. Pseudocode is given in Algorithm 2.

Lemma 10. *Consider a knapsack instance with capacity C and n items with values $v_1, \ldots, v_n$ and sizes $s_1, \ldots, s_n$. Algorithm 2 runs in time $O(n \log n)$. Moreover, there is a feedback system $A_1, \ldots, A_M$ partitioning $\mathcal{C}$ into $M = O(n^2)$ intervals such that set of items output by the algorithm is constant for $\rho \in A_i$. When run with parameter ρ, in addition to the item set S, the algorithm outputs the interval A_i containing ρ.*

Algorithm 2 Semi-bandit Knapsack

```
Input: Parameter $\rho \geq 0$, item values $v_1, \ldots, v_n$, item sizes $s_1, \ldots, s_n$, knapsack capacity $C \geq 0$.
1. Let $\pi : [n] \to [n]$ be the item permutation such that
   $\sigma_\rho(\pi(1)) \geq \cdots \geq \sigma_\rho(\pi(n))$.
2. Initialize $S \leftarrow \emptyset$.
3. For $i = 1, \ldots, n$: if $s_{\pi(i)} \leq C$ then add $\pi(i)$ to $S$ and
   set $C \leftarrow C - s_{\pi(i)}$.
4. For $i = 1, \ldots, n-1$: let $c_i \leftarrow \frac{\log(v_{\pi(i)}/v_{\pi(i+1)})}{\log(s_{\pi(i)}/s_{\pi(i+1)})}$.
5. Let $\rho_{\min} \leftarrow \max\{c_i \mid c_i \leq \rho\}$.
6. Let $\rho_{\max} \leftarrow \min\{c_i \mid c_i > \rho\}$.
7. Return $S$ and interval $A = (\rho_{\min}, \rho_{\max})$.
```

Proof sketch. The items selected by the algorithm only depend on the item ordering π. Steps 4 and 5 compute the largest parameter interval containing ρ with the same item ordering as ρ, and therefore the items output by the algorithm is constant on this interval. Based on the work of Gupta and Roughgarden [21], we know there are at most $O(n^2)$ such intervals. □

In contrast to Algorithm 2, the most direct approach to obtaining full-information feedback for this family of knapsack algorithms is to first compute a set of $O(n^2)$ critical parameter values arising from all pairs of points and to run the algorithm once for each cell in the corresponding partition, taking $O(n^3 \log n)$ time.

Next, we provide a dispersion analysis for selecting the parameter $\rho \in [0, R]$ in order to maximize the value of items selected. We assume that each instance has the same capacity C, item sizes are in $[1, C]$, and the item values are in $[0, 1]$ and κ-smooth. The corresponding loss function is $\ell(\rho) = C - \sum_{i \in S_\rho} v_i \in [0, C]$, where S_ρ is the set of items selected by Algorithm 2 when run with parameter ρ.

Lemma 11. *Consider an adversary choosing knapsack instances with a fixed knapsack capacity C where the t^{th} instance has item sizes $s_1^{(t)}, \ldots, s_n^{(t)} \in [1, C]$, and κ-smooth item values $v_1^{(t)}, \ldots, v_n^{(t)} \in [0, 1]$. The loss functions $\ell_1, \ell_2, \ldots$ defined above are piecewise constant, f-dispersed for $f(T, \epsilon) = T\epsilon n^2 \kappa^2 \ln(C) + O(\sqrt{T \log(Tn)})$, and β-dispersed for $\beta = 1/2$.*

Proof. Let $c_{ij}^{(t)} = \log(v_i^{(t)}/v_j^{(t)})/\log(s_i^{(t)}/s_j^{(t)})$ be the critical parameter value such that at $\rho = c_{ij}^{(t)}$, items i and j swap their relative order in the t^{th} instance. Balcan et al. [11] show that each critical value $c_{ij}^{(t)}$ is random and has a density function bounded by $\kappa^2 \ln(C)/2$. It follows that for any interval I of radius ϵ, the expected total number of critical values $c_{ij}^{(t)}$ summed over all pairs of items and

$t = 1, \ldots, T$ is at most $T\epsilon n^2\kappa^2 \ln(C)$. This is also an upper bound on the expected number of loss functions in $\ell_1, \ldots, \ell_T$ that are not constant on I. Applying Theorem 7, it follows that the functions are f-dispersed for $f(T, \epsilon) = T\epsilon n^2\kappa^2 \ln(C) + O(\sqrt{T \log(Tn)}) = \tilde{O}(T\epsilon + \sqrt{T})$, which implies β-dispersion with $\beta = 1/2$. □

Running Algorithm 1 using the semi-bandit feedback returned by Algorithm 2, we obtain the following.

Corollary 12. *Under the same conditions as Lemma 11, using Algorithm 1 to tune the parameter $\rho \in [0, R]$ of Algorithm 2 under semi-bandit feedback has expected regret bounded by $O(Cn\sqrt{T \log(RTn\kappa \log(C))})$.*

The full-information regret bound obtained by Balcan et al. [11] is $\tilde{O}(Cn^2\sqrt{T})$, which is worse than our semi-bandit bound (but can be improved to $\tilde{O}(C\sqrt{T})$ using our tighter dispersion analysis).

4.2 INTERPOLATING BETWEEN SINGLE AND COMPLETE LINKAGE CLUSTERING

Next, we consider a rich family of linkage-based clustering algorithms introduced by Balcan et al. [9] that interpolates between the classic single and complete linkage procedures. Clustering instances are described by a matrix $D = (d_{ij}) \in \mathbb{R}^{n\times n}$ giving the pairwise distances between a collection of n data points and the goal is to organize the points into a hierarchy or cluster tree. We provide the first dispersion analysis and online configuration procedures for this class of algorithms. We assume that each distance d_{ij} is κ-smooth.

The algorithm family we consider, called ρ-linkage, is family of agglomerative clustering algorithms with a single parameter $\rho \in [0, 1]$. These algorithms take as input a distance matrix $D \in \mathbb{R}^{n\times n}$ with entries d_{ij} and the parameter value $\rho \in [0, 1]$ and output a cluster tree, which is a binary tree where each node corresponds to a cluster in the data. The leaves of the tree are the individual data points, while the root node corresponds to the entire dataset. The children of each node subdivide that cluster into two subclusters. The ρ-linkage algorithm starts with each point belonging to its own cluster. Then, it repeatedly merges the closest pair of clusters according the distance defined by $\mathrm{d}_\rho(A, B) = (1 - \rho)\,\mathrm{d}_{\min}(A, B) + \rho\,\mathrm{d}_{\max}(A, B)$, where A and B are clusters (i.e., subsets of $[n]$), $\mathrm{d}_{\min}(A, B) = \min_{a\in A, b\in B} d_{ab}$ and $\mathrm{d}_{\max}(A, B) = \max_{a\in A, b\in B} d_{ab}$. When there is only a single cluster remaining, the algorithm outputs the constructed cluster tree.

For any pair of candidate cluster merges (C_1, C_2) and (C_1', C_2'), where C_1, C_2, C_1' and C_2' are clusters, there is a critical parameter value c such that $d_\rho(C_1, C_2) = d_\rho(C_1', C_2')$ only when $\rho = c$. To simplify notation in the rest of this section, we let $c(C_1, C_2, C_1', C_2') = \Delta_{\min}/(\Delta_{\min} - \Delta_{\max})$, where $\Delta_{\min} = \mathrm{d}_{\min}(C_1', C_2') - \mathrm{d}_{\min}(C_1, C_2)$ and $\Delta_{\max} = \mathrm{d}_{\max}(C_1', C_2') - \mathrm{d}_{\max}(C_1, C_2)$.

First, we show how to obtain semi-bandit feedback for this family of linkage algorithms by running a single algorithm in the family. Our modified algorithm maintains an interval $(\rho_{\min}, \rho_{\max})$ with the invariant that at any iteration, for all parameters $\rho' \in (\rho_{\min}, \rho_{\max})$, the algorithm would make the same merges that have been made so far. Pseudocode for this procedure is given in Algorithm 3

Algorithm 3 Semi-bandit ρ-Linkage

```
Input: Parameter $\rho \in [0,1]$, distance matrix $D \in \mathbb{R}^{n\times n}$.
1. Let $S \leftarrow \{\mathrm{Leaf}(i) \text{ for } i \in [n]\}$.
2. Let $\rho_{\min} \leftarrow 0$ and $\rho_{\max} \leftarrow 1$.
3. While $|S| > 1$:
   (a) Let $(C_1, C_2) = \mathrm{argmin}_{C_1, C_2 \in S}\, d_\rho(C_1, C_2)$.
   (b) For each pair $(C_1', C_2') \neq (C_1, C_2)$ in $S$
       i. Let $c' \leftarrow c(C_1, C_2, C_1', C_2')$.
       ii. If $c' > \rho$ then set $\rho_{\max} \leftarrow \min(\rho_{\max}, c')$,
           otherwise set $\rho_{\min} \leftarrow \max(\rho_{\min}, c')$.
   (c) Remove $C_1$ and $C_2$ and add $\mathrm{Node}(C_1, C_2)$ to $S$.
4. Return the only element $T$ of $S$ and $A = [\rho_{\min}, \rho_{\max}]$.
```

Lemma 13. *Consider a clustering instance with distance matrix $D \in \mathbb{R}^{n\times n}$. Algorithm 3 runs in time $O(n^3)$. Moreover, there is a feedback system $A_1, \ldots, A_M$ partitioning $[0, 1]$ into $M = O(n^8)$ intervals such that the cluster tree output by the algorithm is constant for $\rho \in A_i$. When run with parameter ρ, in addition to the cluster tree T, the algorithm outputs the interval A_i containing ρ.*

Proof sketch. On each iteration, we compute the critical parameter values where the pair of clusters chosen in step (a) of Algorithm 3 would change. All parameters in the largest interval containing ρ and no critical parameter values from any iterations will result in exactly the same clustering. Balcan et al. [9] showed that each clustering instance has at most $O(n^8)$ discontinuities, which bounds the number of feedback sets obtained in this way. □

Similarly to the knapsack example, the most direct approach for obtaining full-information feedback is to first calculate a set of $O(n^8)$ critical parameter values arising from all $O(n^8)$ subsets of 8 points and to run ρ-linkage once for each interval in the corresponding partition. By using a priority queue to maintain the distances between clusters, it is possible to implement ρ-linkage in $O(n^2 \log n)$ time. This leads to a total running time of $O(n^{10} \log n)$—much higher than the $O(n^3)$ running time in Lemma 13. Note that using a priority queue in Algorithm 3 does not reduce the running time to $O(n^2 \log n)$, since updating the interval $(\rho_{\min}, \rho_{\max})$ requires a linear pass through all $O(n^2)$ pairs of clusters, so finding the closest pair faster does not reduce the running time.

Next, we provide a dispersion analysis for selecting the parameter ρ of Algorithm 3 when the clustering instances are chosen by a smoothed adversary. In particular, we suppose that on each round the adversary chooses a distance matrix $D^{(t)}$ where each distance $d_{ij}^{(t)}$ is κ-smooth and takes values in $[0, B]$. The quantity $B/(1/\kappa) = B\kappa$ roughly captures the scale of the perturbations relative to the true distances. Our analysis leads to regret that depends on $B\kappa$ only logarithmically and give good bounds even for exponentially small perturbations.

Fix any loss function $g : \mathbb{R}^{n\times n} \times \textsc{ClusterTrees} \to [0, 1]$, where $g(D, T)$ measures the cost of cluster tree T for distance matrix D. For example, $g(D, T)$ could be the k-means cost of the best k-pruning of the tree T or the distance to a ground-truth target clustering. We study the loss functions given by $\ell_t(\rho) = g(D^{(t)}, \mathcal{A}(D^{(t)}; \rho))$, where $\mathcal{A}(D; \rho)$ denotes the output cluster tree of Algorithm 3 run on distance matrix D with parameter ρ.

Lemma 14. *Consider an adversary choosing clustering instances where the t^{th} instance has symmetric distance matrix $D^{(t)} \in [0, B]^{n\times n}$ and for all $i \le j$, $d_{ij}^{(t)}$ is κ-smooth. The losses $\ell_1, \ell_2, \dots$ defined above are piecewise constant, f-dispersed for $f(T, \epsilon) = 32T\epsilon n^8\kappa^2 M^2 + O(\sqrt{T\log(Tn)})$ and β-dispersed for $\beta = 1/2$.*

Proof sketch. In the proof of Lemma 13, we showed that for each time t, there are $O(n^8)$ critical parameter values partitioning $\mathcal{C}$ into regions so that the algorithm output is constant on each region. Since the loss ℓ_t only depends on ρ through the algorithm output, ℓ_t is also piecewise constant with at most $O(n^8)$ pieces.

Moreover, we argued that every discontinuity of ℓ_t occurs at a critical parameter value of the form $c = (d_{rr'}^{(t)} - d_{ii'}^{(t)})/(d_{jj'}^{(t)} - d_{ii'}^{(t)} + d_{rr'}^{(t)} - d_{ss'}^{(t)})$ where $i, i', j, j', r, r', s, s'$ are 8 point indices. Similarly to the knapsack example, we show that each critical parameter value is random and has a density function bounded by $16(\kappa B)^2$. From this, it follows that for any interval I of radius ϵ, summing over all times $t = 1, \dots, T$ and all subsets of 8 points, we have that the expected total number of critical values that land in interval I is at most $32T\epsilon(\kappa B)^2$. This also bounds the expected number of functions $\ell_1, \dots, \ell_T$ that are not constant on I. By Theorem 7, the functions are f-dispersed for $f(T, \epsilon) = 32T\epsilon(\kappa B)^2 + \sqrt{T\log(Tn)} = \tilde{O}(T\epsilon + \sqrt{T})$, also implying $\frac{1}{2}$-dispersion.

There are several cases when bounding the density of the critical value c, depending on whether any of the 4 distances correspond to the same entry in the distance matrix D. We give the argument for the case when all 4 distances are distinct entries and therefore independent. The remaining cases are similar and considered in Appendix C. Let $X = d_{rr'} - d_{ii'}$ and $Y = d_{jj'} - d_{ss'}$ so that $c = X/(X + Y)$. The variables X and Y are independent. Since X and Y are each the sum of κ-smooth random variables, Lemma 25 implies that they are each have κ-bounded densities. Using the fact that $|X + Y| \le 2B$, applying Lemma 27 implies that the ratio $c = X/(X+Y)$ has a $16(\kappa B)^2$ bounded density, as required. $\square$

Running Algorithm 1 using the semi-bandit feedback returned by Algorithm 3, we obtain the following:

Corollary 15. *Under the same conditions as Lemma 14, using Algorithm 1 to tune the parameter $\rho \in [0, 1]$ of Algorithm 3 under semi-bandit feedback has expected regret bounded by $O(n^4\sqrt{T\log(Tn\kappa B)})$.*

In Appendix C.2 we show how to extend these results to apply to the case of also learning a metric in addition to interpolating between single and complete linkage.

5 CONCLUSION

In this work, we provide the first online optimization algorithm for piecewise Lipschitz functions under semi-bandit feedback with regret bounds that depend on the dispersion of the loss functions. We also give general tools for verifying dispersion in applications with exponentially tighter bounds than prior work. Finally, we apply our results to two data-driven algorithm design problems. We obtain the first online data-driven algorithm design procedure for a family of linkage-based clustering algorithms, and an online data-driven algorithm design procedure for a greedy family of knapsack algorithms that is more efficient and has better regret bounds than prior work. A cornerstone of our results is that, for many data-driven algorithm design problems, semi-bandit feedback can be obtained as efficiently as bandit-feedback and is sufficient for our algorithms to achieve nearly the same regret bounds as under full-information feedback. Our results largely mitigate the tradeoff between computational efficiency and good regret bounds suffered by prior approaches, making online data-driven algorithm design practical.

Acknowledgements. This work was supported in part by NSF grants CCF-1535967, IIS-1618714, IIS-1901403, CCF-1910321, SES-1919453, DMS1700365, the Defense Advanced Research Projects Agency under cooperative agreement HR0011202000, an AWS Machine Learning Research Award, an Amazon Research Award, a Microsoft Research Faculty Fellowship, a Bloomberg Research Grant, and by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program.

References

[1] Anders Aamand, Piotr Indyk, and Ali Vakilian. (learned) frequency estimation algorithms under zipfian distribution, 2019.

[2] Noga Alon, Nicolò Cesa-Bianchi, Claudio Gentile, Shie Mannor, Yishay Mansour, and Ohad Shamir. Nonstochastic multi-armed bandits with graph-structured feedback. *SIAM J. Comput.*

[3] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 2014.

[4] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 2002.

[5] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. In *Information Processing Letters*, 2012.

[6] Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. In *ICML*, 2014.

[7] Maria-Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. In *SIAM Journal on Computing*, 2016.

[8] Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-center clustering under perturbation resilience. In *ICALP*, 2016.

[9] Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. *COLT*, 2017.

[10] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *ICML*, 2018.

[11] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *FOCS*, 2018.

[12] Maria-Florina Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized lloyd's families. In *NeurIPS*, 2018.

[13] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. A general theory of sample complexity for multi-item profit maximization. In *EC*, 2018.

[14] Shai Ben-David, David Pal, and Shai Shalev-Shwartz. Agnostic online learning. In *COLT*, 2009.

[15] Yves Caseau, François Laburthe, and Glenn Silverstein. A meta-heuristic factory for vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, 1999.

[16] Vincent Cohen-Addad and Varun Kanade. Online Optimization of Smoothed Piecewise Constant Functions. In *AISTATS*, 2017.

[17] Jim Demmel, Jack Dongarra, Victor Eijkhout, Erika Fuentes, Antoine Petitet, Rich Vuduc, R Clint Whaley, and Katherine Yelick. Self-adapting linear algebra algorithms and software. *Proceedings of the IEEE*, 2005.

[18] Jim Demmel, Jack Dongarra, Victor Eijkhout, Erika Fuentes, Antoine Petitet, Rich Vuduc, R Clint Whaley, and Katherine Yelick. Self-adapting linear algebra algorithms and software. *Proceedings of the IEEE*, 2005.

[19] Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Learning space partitions for nearest neighbor search, 2019.

[20] Anna Grosswendt and Heiko Roeglin. Improved analysis of complete linkage clustering. In *European Symposium of Algorithms*, 2015.

[21] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 2017.

[22] András György, Tamás Linder, and György Ottucsák. The shortest path problem under partial monitoring. In *COLT*, 2006.

[23] Eric Horvitz, Yongshao Ruan, Carla Gomez, Henry Kautz, Bart Selman, and Max Chickering. A bayesian approach to tackling hard computational problems. In *UAI*, 2001.

[24] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *ICLR*, 2018.

[25] Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-based low-rank approximations, 2019.

[26] Satyen Kale, Lev Reyzin, and Robert E. Shapire. Nonstochastic bandit slate problems. In *NeurIPS*, 2010.

[27] Robert Kleinberg, Kevin Leyton-Brown, and Brendan Lucier. Efficiency through procrastination: Approximately optimal algorithm configuration with runtime guarantees. In *IJCAI*, 2017.

[28] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *JACM*, 2009.

[29] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *JACM*, 2009.

[30] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. In *Machine Learning*, 1998.

[31] Shie Mannor and Ohad Shamir. From bandits to experts: On the value of side-observations. In *NeurIPS*, 2011.

[32] Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Stochastic, constrained, and smoothed adversaries. In *NeurIPS*, 2011.

[33] John R Rice. The algorithm selection problem. In *Advances in Computers*. Elsevier, 1976.

[34] Mehreen Saeed, Onaiza Maqbool, Haroon Atique Babri, Syed Zahoor Hassan, and S. Mansoor Sarwar. Software clustering techniques and the use of combined algorithm. In *European Conference on Software Maintenance and Reengineering*, 2003.

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

[36] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *JACM*, 2004.

[37] Gilles Stoltz. *Incomplete information and internal regret in prediction of individual sequences*. PhD thesis, Université Paris Sud-Paris XI, 2005.

[38] Gellért Weisz, András György, and Csaba Szepesvári. CapsAndRuns: An improved method for approximately optimal algorithm configuration. In *ICML 2018 AutoML Workshop*, 2018.

[39] Gellért Weisz, András György, and Csaba Szepesvári. Leapsandbounds: A method for approximately optimal algorithm configuration. In *ICML*, 2018.

[40] James R. White, Saket Navlakha, Niranjan Nagarajan, Mohammad-Reza Ghodsi, Carl Kingsford, and Mihai Pop. Alignment and clustering of phylogenetic markers—implications for microbial diversity studies. In *BCM Bioinformatics*, 2010.

[41] L. Xu, F. Hutter, H.H. Hoos, and K. Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *JAIR*, 2008.