Beyond Worst-Case Analysis in Combinatorial Optimization

Colin White

January 10, 2018

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
`crwhite@cs.cmu.edu`

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Traditionally, the theory of algorithms has focused on the analysis of worst-case instances. While this has led to beautiful results and a thorough understanding of a wide variety of problems, there are many problems for which worst-case analysis is not useful for empirical or real-world instances. A rapidly developing line of research, the so-called *beyond worst-case analysis* of algorithms (BWCA), considers the design and analysis of problems using more realistic models or using natural structural properties. The goals of BWCA are to design realistic models and new theoretical algorithms which perform well in practice, as well as to explain why preexisting algorithms that do well in practice perform better than worst-case analysis suggests. In other words, the goal of BWCA is to bridge the gap between theory and practice. In this thesis, we propose to continue this line of work by making contributions in several areas. Specifically, we focus on three main problems and models.

- Clustering is one problem that has benefitted greatly from BWCA. Using the *perturbation resilience* assumption proposed by Bilu and Linial (2011), we design robust, efficient algorithms that output near-optimal clusterings on the stable parts of the data. We consider center-based objectives such as k-median and k-means, as well as symmetric and asymmetric k-center.
- Clustering has many varied applications in practice, and an algorithm may have drastically different performance between two applications. To reconcile this fact, we consider the *algorithm configuration* problem, in which a clustering application is represented by a distribution over problem instances, and the goal is to find the best algorithm for the (unknown) distribution. This model was first studied theoretically by Gupta and Roughgarden (2016). We define rich, infinite classes of agglomerative clustering algorithms with dynamic programming, and then show tight bounds on the pseudo-dimension of these classes, which leads to computational- and sample-efficient algorithm configuration.
- As datasets become larger and larger, distributed learning becomes more prevalent. Motivated by the fact that similar datapoints often belong to the same or similar classes, we propose data-dependent dispatching that takes advantage of such structure. We provide new dispatching algorithms which cast the problem as clustering with important balance and fault-tolerance conditions. Finally, we provide general and robust distributed clustering algorithms under worst-case and beyond worst-case analysis.

Contents

Chapter 1

Introduction

Traditionally, the theory of algorithms has focused on the analysis of worst-case instances. This approach has led to many elegant algorithms, as well as a thorough understanding of many problems. Nevertheless, there are many problems for which worst-case analysis is not useful for empirical or real world instances. Perhaps the most famous example is linear programming - from a worst-case perspective, the simplex algorithm is far worse than the ellipsoid method because the former runs in exponential time as opposed to polynomial time. However, the simplex algorithm is the method of choice for practitioners, since it runs much faster than the ellipsoid method on real world instances. Indeed, the worst-case instances in theory seem to be delicate constructions that break under tiny perturbations. These particular instances dominate worst-case analysis but do not show up in practice. Spielman and Teng made this observation explicit by showing the simplex algorithm runs in linear time under "smoothed analysis" [72], in which an adversary chooses an arbitrary instance, and then the instance undergoes a small randomized perturbation.

The seminal work by Spielman and Teng was one of the first papers in a rapidly developing line of work in the algorithms community, the so-called *beyond worst-case analysis* of algorithms (BWCA), which considers the design and analysis of problem instances under natural structural properties and more practical setups. The goals of BWCA are to use realistic models to design new theoretical algorithms which perform well in practice, as well as to explain why some preexisting algorithms that do well in practice perform better than worst-case analysis suggests. In other words, the goal of BWCA is to bridge the gap between theory and practice.

In recent years, one of the problems which has benefited the most from BWCA is *clustering*. Clustering is a fundamental problem in combinatorial optimization with numerous real-life applications in areas from bioinformatics to computer vision to text analysis and so on. The underlying goal is to group a given set of points to maximize similarity inside a group and minimize similarity among groups. BWCA has also seen success in problems ranging from distributed learning to computing Nash equilibria to the famous traveling salesman problem.

In this thesis, our goal is to continue the strong line of BWCA. We focus on clustering problems, distributed learning, and traveling salesman. We accomplish this by defining new models and constructing new state-of-the-art algorithms. Specifically, we focus on these challenges:

Clustering under Perturbation Resilience. The popular notion of α-perturbation resilience, introduced by Bilu and Linial [24], informally states that the optimal solution does not change when the input distances are allowed to increase by up to a factor of α. This definition seeks to capture a phenomenon in practice: the optimal solution often "stands out", thereby the optimal solution does not change even when the input is slightly perturbed. Several recent papers have successfully applied perturbation resilience to clustering, culminating in an optimal algorithm for k-median, k-means, and k-center under 2-perturbation resilience [4].

In Chapter 2, we first show an algorithm for asymmetric k-center under 2-perturbation resilience. The algorithm is *locally robust*: it returns the optimal clusters in the areas of perturbation resilient data, and it never returns worse than an $O(\log^* n)$-approximation (which matches the worst-case approximation ratio for asymmetric k-center). Next, we study the relaxed definition, (α, ϵ)-perturbation resilience, in which an ϵ-fraction of the datapoints may switch clusters under an α-perturbation. We show locally robust algorithms for k-center and asymmetric k-center under this assumption. Finally, we give hardness results for clustering under perturbation resilience.

Algorithm Configuration. In Chapter 3, we consider a different type of BWCA where it is assumed that the algorithm designer has a specific task at hand, such as clustering protein sequences, or clustering documents in a database, which follows an unknown distribution over problem instances. We use the PAC-learning framework for algorithm configuration, recently defined by Gupta and Roughgarden [49]. We define rich, infinite classes of linkage-based algorithms with pruning, and prove tight bounds on the pseudo-dimension of these classes. We show how this implies computationally efficient algorithms to learn the best algorithm parameters for a specific application.

Distributed Learning. Chapter 4 introduces yet another take on BWCA. In distributed learning, a common, simple approach is to randomly distribute the data among m different machines. Motivated by the fact that data is "locally simple but globally complex", we show how to cluster and dispatch datapoints so that similar data winds up on the same machine. The key is to cast the dispatching as a clustering problem with the real-world constraints of fault-tolerance and balancedness which add nontrivial and novel challenges to overcome. Finally, we consider the distributed clustering problem. We construct general and robust algorithms for distributed clustering in worst-case and BWCA settings.

Future Work. In Chapter 5, we propose several concrete open questions and plans for attacking them. Answering these questions would continue the line of BWCA in multiple areas. Specifically, we propose open questions for improving the state of the art along multiple axes (running time, and level of stability assumed) for clustering and traveling salesman under perturbation resilience. We also propose to improve the running time of algorithm configuration to make the algorithms tractable in the real world. Finally, we propose to apply the algorithm configuration framework to distributed machine learning, to learn the best parameters for each application.

Chapter 2

Clustering under Perturbation Resilience

2.1 Introduction

Clustering is a fundamental problem in machine learning with applications in many areas including computer vision, text analysis, bioinformatics, and so on. The underlying goal is to group a given set of points to maximize similarity inside a group and dissimilarity among groups. A common approach to clustering is to set up an objective function and then approximately find the optimal solution according to the objective. Examples of these objective functions include k-means, k-median, and k-center, and more generally any ℓ_p objective, in which the goal is to find k centers to minimize the sum of the ℓ_p distances from each point to its closest center. These popular objective functions are provably NP-hard to optimize [46, 53, 56], and finding approximation algorithms to different clustering objectives and variants has attracted significant attention in the computer science community [6, 28, 30, 31, 33, 46, 63].

For a real world clustering application, there may be uncertainty in the data. For instance, if the clustering is over a road network weighted by driving times, the weights might change depending on rush-hour traffic. Ideally, we would like the optimal solution to not change under these small uncertanties in the distances. The popular notion of α-perturbation resilience, introduced by Bilu and Linial [24], makes this assumption explicit: it informally states that the optimal solution does not change when the input distances are allowed to increase by up to a factor of α. The typical approach in BWCA is to take this α-perturbation resilience assumption, and try to find the minimum value of α which admits an algorithm that finds the optimal solution in polynomial time.

In this chapter, we construct robust algorithms that output near-optimal clusters on the stable parts of the data, while still achieving the worst-case approximation ratio over the unstable parts of the data. We present algorithms that simultaneously output the optimal clusters from the stable regions of the data, while achieving state-of-the-art approximation ratios over the rest of the data. To answer this question, we define the notion of *local perturbation resilience*. This is the first definition that applies to an individual region of the data, rather than the dataset as a whole. Informally, a set of optimal clusters satisfies α-perturbation resilience if the clusters remain in the optimal solution under any α perturbation to the (entire) input. We show that every optimal cluster in a locally stable region will be returned by our algorithms.

The work from this chapter is based on the following papers: [16, 19].

2.2 Preliminaries

A clustering instance consists of a set S of n points, as well as a distance function $d : S \times S \to \mathbb{R}_{\geq 0}$. For a point $u \in S$ and a set $A \subseteq S$, we define $d(u, A) = \min_{v \in A} d(u, v)$. The k-median, k-means, and k-center objectives are to find a set of points $X = \{x_1, \ldots, x_k\} \subseteq S$ called *centers* to minimize $\sum_{v \in S} d(v, X)$, $\sum_{v \in S} d(v, X)^2$, and $\max_{v \in S} d(v, X)$, respectively. We denote $\text{Vor}_X(x) = \{v \in S \mid x = \text{argmin}_{y \in X} d(v, y)\}$, the Voronoi tile of $x \in X$ induced by X on the set of points S, and we denote

$\text{Vor}_X(X') = \bigcup_{x \in X'} \text{Vor}_X(x)$ for a subset $X' \subseteq X$. We refer to the Voronoi partition induced by X as a clustering. Throughout the chapter, we denote the clustering with minimum cost by $\mathcal{OPT} = \{C_1, \ldots, C_k\}$, and we denote the optimal centers by $c_1, \ldots, c_k$, where c_i is the center of C_i for all $1 \leq i \leq k$.

Some of the distance functions we study are metrics, and sometimes we use an *asymmetric* distance function. An asymmetric distance function satisfies all the properties of a metric space except for symmetry. In particular, an asymmetric distance function must satisfy the *directed* triangle inequality: for all $u, v, w \in S$, $d(u, w) \leq d(u, v) + d(v, w)$.

Now we formally define *perturbation resilience*, a notion introduced by [24]. d' is called an α-perturbation of the distance function d, if for all $u, v \in S$, $d(u, v) \leq d'(u, v) \leq \alpha d(u, v)$. (We only consider perturbations in which the distances increase because WLOG we can scale the distances to simulate decreasing distances.)

Definition 2.2.1. *Given a clustering instance (S, d), a subset $S' \subseteq S$ satisfies* α-perturbation resilience *(α-PR) if for any α-perturbation d' of d, for each i such that $S' \cap C_i \neq \emptyset$, the optimal clustering $\mathcal{C}'$ under d' is unique and contains C_i.*

Note that the optimal *centers* might change under an α-perturbation, but the optimal *clustering* must stay the same. All prior work has considered perturbation resilience with respect to the entire dataset ($S' = S$). Under that definition, it is assumed the optimal clustering is unique and does not change under *any* α-perturbation.

Next we define a natural strengthening of α-PR. Intuitively, given an α-PR region S', the subset $S'' \subseteq S'$ satisfies *α-strong perturbation resilience* if it contains no points on the boundary of S'. We define this notion specifically for the k-center clustering objective. Here and throughout the chapter, we denote the optimal k-center radius by r^*.

Definition 2.2.2. *Given a k-center clustering instance (S, d), a subset S'' satisfies* α-strong perturbation resilience (α-SPR) *if there exists $S' \supseteq S''$ which is α-PR, and furthermore, for each $u \in S''$ if $d(u, v) \leq r^*$ then $v \in S'$ (any point within distance r^* of a point in S'' must be α-PR).*

Fact 2.2.3. *Given a k-center clustering instance (S, d) with optimal clustering $\mathcal{C}$, there exists an α-SPR subset S' which is laminar with respect to $\mathcal{C}$, and for all α-SPR sets S'', $S'' \subseteq S'$.*

One consequence of Fact 2.2.3 is that we can characterize the unique maximal α-PR and α-SPR regions as a subset of the optimal clusters. Throughout the chapter, we will often state a cluster C_i is an α-PR cluster, or a center c_i is an α-PR center, meaning c_i is in the maximally stable region.

Next, we define a more robust version of α-PR that allows a small change in the optimal clusters when the distances are perturbed. We say that two clusters A and B are ϵ-close if they differ by only ϵn points, i.e., $|A \setminus B| + |B \setminus A| \leq \epsilon n$. We say that two clusterings $\mathcal{C}$ and $\mathcal{C}'$ are ϵ-close if $\min_\sigma \sum_{i=1}^{k} |C_i \setminus C'_{\sigma(i)}| \leq \epsilon n$ (where σ denotes a perturbation of $[k]$).

Definition 2.2.4. *Given a clustering instance (S, d) with optimal clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$, a subset $S' \subseteq S$ satisfies* (α, ϵ)-perturbation resilience ((α, ϵ)-PR) *if for any α-perturbation d' of d, for each i such that $S' \cap C_i \neq \emptyset$, the optimal clustering $\mathcal{C}'$ under d' contains a cluster C'_i which is ϵ-close to C_i.*

Similar to α-PR, this notion has only been studied for the case when $S' = S$ (which was first defined by [17]).

In all definitions thus far, we do not assume that the α-perturbations satisfy the triangle inequality. [4] recently studied the weaker definition in which the α-perturbations must satisfy the triangle inequality, called *metric perturbation resilience*. All of our definitions can be generalized accordingly, and some of our results hold under this weaker assumption.

2.3 Results and Techniques

Asymmetric k-center The asymmetric k-center problem admits an $O(\log^* n)$ approximation algorithm due to Vishwanathan [75], which is tight [35]. We give a modification to the algorithm which further ensures we output all optimal 2-SPR clusters.

Theorem 2.3.1. *Given an asymmetric k-center clustering instance (S, d) of size n, there exists an efficient algorithm that returns each 2-SPR cluster exactly, and furthermore, the overall clustering is an $O(\log^* n)$-approximation. This statement hold for metric perturbation resilience as well.*

We combine the tools of Vishwanathan with the perturbation resilience assumption to prove this two-part guarantee. Specifically, we use the notion of a *center-capturing vertex (CCV)*, which is used in the first phase of the approximation algorithm to pull out supersets of clusters. A CCV c has the property that it is distance r^* to any point which is distance r^* to c. We can determine wheter a point is a CCV in polynomial time, and any CCV must be distance $2r^*$ to its entire optimal cluster, by the triangle inequality. The first phase Vishwanathan's algorithm iteratively finds an arbitrary CCV, and then pulls out all points distance $2r^*$ from the CCV. When there are no more CCV's, the next phase is an iterative greedy procedure, which crucially requires that there are no CCV's present.

We show that each 2-PR center is itself a CCV. We might hope that running Vishwanathan's algorithm will find all 2-PR centers and pull out their corresponding clusters. However, a single CCV might mark two 2-PR centers in the same iteration. This was not an issue for Vishwanathan's approximation guarantee, but it would relinquish our requirement of outputting each 2-SPR cluster exactly. The key challenge is thus carefully specifying which nearby points get marked by each CCV c chosen by the algorithm: mark too many points and we risk marking a different 2-PR center; mark too few points and we might not fully cover an optial cluster, sacrificing the $O(\log^* n)$ approximation guarantee.

We fix this problem by showing an extra structural property for 2-PR centers which we call *CCV-proximity*. Intuitively, a point c satisfying CCV-proximity must be closer than other CCVs to each point v such that $d(v, c) \leq r^*$. Then we show that a novel procedure for marking points will give our guarantees: c marks a point v if there exists u such that $d(u, c)$ and $d(u, v)$ are both $\leq r^*$.

Robust perturbation resilience Now we consider (α, ϵ)-PR. For k-center, we show that any 2- approximation algorithm will return the optimal $(3, \epsilon)$-PR clusters, assuming a mild lower bound on optimal cluster sizes. We also impose two sensible conditions on the 2-approximation. *(1)* for every edge $d(u, v) \leq 2r^*$ in the 2-approximation, $\exists w$ s.t. $d(u, w)$ and $d(w, v)$ are $\leq r^*$, and *(2)* there cannot be multiple clusters outputted in the 2-approximation that can be combined into one cluster with the same radius. Both of these properties are easily satisfied using quick pre- or post-processing steps. [1]

Theorem 2.3.2. *Given a k-center clustering instance (S, d) such that all optimal clusters are size $> 2\epsilon n$ and there are at least three $(3, \epsilon)$-PR clusters, then any 2-approximate solution satisfying conditions* (1) *and* (2) *must contain all optimal 2-PR clusters and $(3, \epsilon)$-SPR clusters.*

Our main structural result is that two points from different $(3, \epsilon)$-PR clusters cannot be within distance r^* of each other (from this, Theorem 2.3.2 is a corollary). The proof of the structural result consists of two parts. The first part is to show that if two points from different PR clusters are close together, then *all* points in the clustering instance must be near each other, in some sense. For this analysis, we use the idea of a *cluster-capturing center* [16] to show that if two points from different clusters are very close together, then there must exist a set of $k + 2$ points C, any k of which can become optimal centers under a suitable 3-perturbation.

The second part of the proof consists of showing that most points from three PR clusters must be reasonably far from one another; therefore, we achieve the final result by contradiction. We develop novel machinery to convert the guarantees of local perturbation resilience into global structure that leads to a contradiction when such a set C exists. We crucially show that for each PR cluster, there exists an ordering over C corresponding to their relative strength as a cluster center. Then we derive conditional dependencies

[1] For condition *(1)*, before running the algorithm, remove all edges of distance $> r^*$, and then take the metric completion of the resulting graph. For condition *(2)*, given the radius $\hat{r}$ of the outputted solution, for each $v \in S$, check if the ball of radius $\hat{r}$ around v captures multiple clusters. If so, combine them.

between the set of orderings which lead to a contradiction.

Approximation algorithms under local perturbation resilience For k-center, we show that *any* α-approximation algorithm for k-center will always return the clusters satisfying α-metric perturbation resilience.

Theorem 2.3.3. *Given an asymmetric k-center clustering instance (S, d) and an α-approximate clustering $\mathcal{C}$, each α-PR cluster is contained in $\mathcal{C}$, even under the weaker metric perturbation resilience condition.*

The proof works by creating an α-perturbation d' by increasing all distances by α, except the distances between each point and its center in the α-approximate solution. This ensures that the approximation is optimal under d'. However, d' may be highly non-metric, so our challenge is arguing that the proof still goes through after taking the metric completion of d'.

For k-median, we show that local search returns the optimal 3-perturbation resilient clusters. Intuitively, the approximation guarantee must be true locally as well as globally. This builds off recent work by Cohen-Addad and Schwiegelshohn [36], who showed that local search returns the optimal clustering under a stronger version of $(3 + 2\epsilon)$-PR.

Theorem 2.3.4. *Given a k-median instance (S, d), running local search with search size $\frac{1}{\epsilon}$ returns a clustering that contains every $(3 + 2\epsilon)$-PR cluster, and it gives a $(3 + 2\epsilon)$-approximation overall.*

Lower bounds Finally, we show hardness results for clustering under various levels of stability. We start with hardness for k-center under $(2 - \epsilon)$-approximation stability (a stronger condition than perturbation resilience). Because approximation stability is stronger than perturbation resilience, this result implies k-center under $(2-\epsilon)$-perturbation resilience is hard as well. Similarly, symmetric k-center is a special case of asymmetric k-center, so we get the same hardness results for asymmetric k-center. This proves that Theorem 2.3.1 is tight.

Theorem 2.3.5. *There is no polynomial-time algorithm for finding the optimal k-center clustering under $(2 - \epsilon)$-approximation stability, even when assuming all optimal clusters are size $\geq \frac{n}{2k}$, unless $NP = RP$.*

We show a reduction from a special case of Dominating Set which we call Unambiguous-Balanced-Perfect Dominating Set. A reduction from Perfect Dominating Set (Dominating Set with the additional constraint that for all dominating sets of size $\leq k$, each vertex is hit by exactly one dominator) to the problem of clustering under $(2 - \epsilon)$-center proximity was shown in [23] (α-center proximity is the property that for all $p \in C_i$ and $j \neq i$, $\alpha d(c_i, p) < d(c_j, p)$, and it follows from α-perturbation resilience). Our contribution is to show that Perfect Dominating Set remains hard under two additional conditions. First, in the case of a YES instance, each dominator must hit at least $\frac{n}{2k}$ vertices (which translates to clusters having size at least $\frac{n}{2k}$ as well). Second, we are promised that there is at most one dominating set of size $\leq k$ (which is required for establishing approximation stability for the resulting clustering instance).

Now we provide APX-hardness lower bounds under (α, ϵ)-perturbation resilience. We show hardness of approximation for k-median, k-means, and k-center, even when it is guaranteed the clustering satisfies (α, ϵ)-perturbation resilience for any $\alpha \geq 1$ and $\epsilon > 0$. In fact, the result holds for the stronger notion of (α, ϵ)-approximation stability. This generalizes prior hardness results in BWCA [14, 16].

Theorem 2.3.6. *Given $\alpha \geq 1$, $\epsilon > 0$, it is NP-hard to approximate k-center to 2, k-median to 1.73, or k-means to 1.0013, even when it is guaranteed the instance satisfies (α, ϵ)-approximation stability.*

The hardness is based on a reduction from the general clustering instances. We create a new instance of size $n + n/\epsilon$ which contains the original instance, plus n/ϵ singleton points which are very far from one another. We show that in order to approximate the new instance (which is stable), we must solve the original instance.

2.4 Related Work

The first constant-factor approximation algorithm for k-median was given by Charikar et al. [30], and the current best approximation ratio is 2.675 by Byrka et al. [28]. Jain et al. proved k-median is NP-hard to approximate to a factor better than 1.73 [53]. For k-center, Gonzalez showed a tight 2-approximation algorithm [46]. For k-means, the best approximation ratio was recently lowered to 6.357 by Ahmadian et al. [2]. Euclidean k-means was shown to be APX-hard by Awasthi et al. [11], and the constant was recently improved to 1.0013 [56].

Perturbation resilience was introduced by Bilu and Linial [24], who showed algorithms that outputted the optimal solution for max cut under $\Omega(\sqrt{n})$-perturbation resilience (this was later improved by Makarychev et al. [64]). The study of clustering under perturbation resilience was initiated by Awasthi et al. [10], who provided an optimal algorithm for center-based clustering objectives (which includes k-median, k-means, and k-center clustering, as well as other objectives) under 3-perturbation resilience. This result was improved by Balcan et al. [17], who showed an algorithm for center-based clustering under $(1+\sqrt{2})$-perturbation resilience. They also gave a near-optimal algorithm for k-median under $(2+\sqrt{3},\epsilon)$-perturbation resilience when the optimal clusters are not too small. Balcan et al. [16] constructed algorithms for k-center and asymmetric k-center under 2-perturbation resilience and $(3,\epsilon)$-perturbation resilience, and they showed no polynomial-time algorithm can solve k-center under $(2-\epsilon)$-approximation stability (a notion that is stronger than perturbation resilience) unless $NP = RP$. Angelidakis et al. [4] gave algorithms for center-based clustering under 2-perturbation resilience, and defined a more general notion of metric perturbation resilience. Cohen-Addad and Schweigelshohn [36] showed that local search outputs the optimal k-median solution when the data satisfies a stronger variant of 3-perturbation resilience, in which the optimal centers are not allowed to change. Perturbation resilience has also been applied to other problems, such as minimum multiway cut, the traveling salesman problem, and finding Nash equilibria [14, 64, 67].

There are many other works that show positive results for different natural notions of stability in various settings [5, 9, 50, 51, 54, 55].

Chapter 3

Algorithm Configuration

3.1 Introduction

NP-hard problems arise in a variety of diverse and oftentimes unrelated application domains. For example, clustering is a widely-studied NP-hard problem in unsupervised machine learning, used to group protein sequences by function, organize documents in databases by subject, and choose the best locations for fire stations in a city. Although the underlying objective is the same, a "typical problem instance" in one setting may be significantly different from that in another, causing approximation algorithms to have inconsistent performance across the different application domains.

In this chapter, we study how to characterize which algorithms are best for which contexts, a task often referred to in the AI literature as *algorithm configuration*. This line of work allows researchers to compare algorithms according to an application-specific metric, such as expected performance over their problem domain, rather than a worst-case analysis. If worst-case instances occur infrequently in the application domain, then a worst-case algorithm comparison could be uninformative and misleading. We approach application-specific algorithm configuration via a learning-theoretic framework wherein an application domain is modeled as a distribution over problem instances. We then fix an infinite class of approximation algorithms for that problem and design computationally efficient and sample efficient algorithms which learn the approximation algorithm with the best performance over the distribution, and therefore an algorithm with high performance in the specific application domain. Gupta and Roughgarden [49] introduced this learning framework to the theory community, but it has been the primary model for algorithm configuration and portfolio selection in the artificial intelligence community for decades [70] and has led to breakthroughs in diverse fields including combinatorial auctions [57], scientific computing [39], vehicle routing [29], and SAT [78]. In this framework, we study agglomerative clustering algorithms followed by a dynamic programming step to extract a good clustering. These techniques are widely used in machine learning and across many scientific disciplines for data analysis.

The work from this chapter is based on the following paper: [18].

3.2 Preliminaries

Problem description. We fix a computational problem, such as k-means clustering, and assume that there exists an unknown, application-specific distribution $\mathcal{D}$ over a set of problem instances Π. We denote an upper bound on the size of the problem instances in the support of $\mathcal{D}$ by n. For example, the support of $\mathcal{D}$ might be a set of social networks over n individuals, and the researcher's goal is to choose an algorithm with which to perform a series of clustering analyses. Next, we fix a class of algorithms $\mathcal{A}$. Given a cost function $\texttt{cost} : \mathcal{A} \times \Pi \to [0, H]$, the learner's goal is to find an algorithm $h \in \mathcal{A}$ that approximately optimizes the expected cost with respect to the distribution $\mathcal{D}$, as formalized below.

Definition 3.2.1 ([49])**.** *A learning algorithm L* (ϵ, δ)-learns the algorithm class $\mathcal{A}$ with respect to the cost function $\texttt{cost}$ *if, for every distribution $\mathcal{D}$ over Π, with probability at least $1-\delta$ over the choice of a sample $\mathcal{S} \sim \mathcal{D}^m$, L outputs an algorithm $\hat{h} \in \mathcal{A}$ such that* $\mathbb{E}_{x\sim\mathcal{D}}\left[\texttt{cost}\left(\hat{h}, x\right)\right] - \min_{h\in\mathcal{A}}\left\{\mathbb{E}_{x\sim\mathcal{D}}[\texttt{cost}(h,x)]\right\} < \epsilon$. *We require that the number of samples be polynomial in n, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$, where n is an upper bound on the size of the problem instances in the support of $\mathcal{D}$. Further, we say that L is* computationally efficient *if its running time is also polynomial in n, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$.*

We derive our guarantees by analyzing the pseudo-dimension of the algorithm classes we study. Consider a class of algorithms $\mathcal{A}$ and a class of problem instances $\mathcal{X}$. Let the cost function $\texttt{cost}(h, x)$ denote the abstract cost of running an algorithm $h \in \mathcal{A}$ on a problem instance $x \in \mathcal{X}$. Similarly, define the function class $\mathcal{H}_{\mathcal{A}} = \{\texttt{cost}(h,\cdot) : \mathcal{X} \to \mathbb{R} | h \in \mathcal{A}\}$. Then, recall that a finite subset of problem instances $S = \{x_1, x_2, \dots x_m\}$ is shattered by the function class $\mathcal{H}$, if there exist real-valued witnesses $r_1, \dots, r_m$ such that for all subsets $T \subseteq S$, there exists a function $\texttt{cost}\,(h_T, \cdot) \in \mathcal{H}$, or in other words, an algorithm $h_T \in \mathcal{A}$ such that $\texttt{cost}\,(h_T, x_i) \le r_i$ if and only if $i \in T$. Then, we can define the pseudo-dimension of the algorithm class $\mathcal{A}$ to be the pseudo-dimension $Pdim(\mathcal{H})$ of $\mathcal{H}$ i.e., the cardinality of the largest subset of $\mathcal{X}$ shattered by $\mathcal{H}$.

By bounding $Pdim(\mathcal{H})$, clearly we can derive sample complexity guarantees in the context of algorithm classes [41]: for every distribution $\mathcal{D}$ over $\mathcal{X}$, every $\epsilon > 0$, and every $\delta \in (0, 1]$, $m \ge c\left(\frac{H}{\epsilon}\right)^2\left(Pdim(\mathcal{H}) + \log\frac{1}{\delta}\right)$ for a suitable constant c (independent of all other parameters), then with probability at least $1-\delta$ over m samples $x_1, \dots, x_m \sim \mathcal{D}$,

$$\left|\frac{1}{m}\sum_{i=1}^{m}\texttt{cost}(h, x_i) - \mathbb{E}_{x\sim\mathcal{D}}[\texttt{cost}(h,x)]\right| < \epsilon$$

for every algorithm $h \in \mathcal{A}$. Therefore, if a learning algorithm receives as input a sufficiently large set of samples and returns the algorithm which performs best on that sample, we can be guaranteed that this algorithm is close to optimal with respect to the underlying distribution.

Clustering As in the previous chapter, a clustering instance $\mathcal{V} = (V, d)$ consists of a set V of n points and a distance metric $d : V \times V \to \mathbb{R}_{\ge 0}$ specifying all pairwise distances between these points. The overall goal of clustering is to partition the points into groups such that distances within each group are minimized and distances between each group are maximized. The learner's main goal is to minimize an abstract cost function such as the cluster purity or the clustering objective function, which is the case in many clustering applications such as clustering biological data [42, 66]. Clustering is typically performed using an objective function Φ, such as k-means, k-median, k-center, or the distance to the ground truth clustering. Formally, an objective function Φ takes as input a set of points $\mathbf{c} = \{c_1, \dots, c_k\} \subseteq V$ which we call centers, as well as a partition $\mathcal{C} = \{C_1, \dots, C_k\}$ of V which we call a clustering. We define the rich class of clustering objectives $\Phi^{(p)}(\mathcal{C}, \mathbf{c}) = \sum_{i=1}^{k}(\sum_{q\in C_i} d(q, c_i)^p)^{1/p}$ for $p \in [1, \infty) \cup \{\infty\}$. The k-means, k-median, and k-center objective functions are $\Phi^{(2)}$, $\Phi^{(1)}$, and $\Phi^{(\infty)}$, respectively.[1]

Agglomerative Algorithms with Pruning We study infinite classes of two-step clustering algorithms consisting of a linkage-based step and a dynamic programming step. First, the algorithm runs one of an infinite number of linkage-based routines to construct a hierarchical tree of clusters. Next, the algorithm runs a dynamic programming procedure to find the pruning of this tree that minimizes one of an infinite number of clustering objectives. For example, if the clustering objective is the k-means objective, then the dynamic programming step will return the optimal k-means pruning of the cluster tree. These types of algorithms are prevalent in practice [7, 71, 77] and enjoy strong theoretical guarantees in a variety of settings

[1]There have been several papers that provide theoretical guarantees for clustering under this family of objective functions for other values of p. For instance, see Gupta and Tangwongsan's work [48] which provides a $O(p)$ approximation algorithm when $p < \log n$ and Bateni et al.'s work [21] which studies distributed clustering algorithms.

[10, 16, 17, 47]. Examples of these algorithms include the popular *single-*, *complete-*, and *average-linkage* algorithms with dynamic programming. An agglomerative clustering algorithm with dynamic programming is defined by two functions: a merge function and a pruning function. A merge function $\xi(A, B) \to \mathbb{R}_{\geq 0}$ defines a distance between two sets of points $A, B \subseteq V$. The algorithm builds a *cluster tree* $\mathcal{T}$ by starting with n singleton leaf nodes, and iteratively merging the two sets with minimum distance until there is a single node remaining, consisting of the set V. The children of any node T in this tree correspond to the two sets of points that were merged to form T during the sequence of merges. Common choices for the merge function ξ include $\min_{a \in A, b \in B} d(a, b)$ (single linkage), $\frac{1}{|A| \cdot |B|} \sum_{a \in A, b \in B} d(a, b)$ (average linkage) and $\max_{a \in A, b \in B} d(a, b)$ (complete linkage).

A pruning function Ψ takes as input a k'*-pruning* of any subtree of $\mathcal{T}$ and returns a score $\mathbb{R}_{\geq 0}$ for that pruning. A k'-pruning for a subtree T is a partition of the points contained in T's root into k' clusters such that each cluster is an internal node of T. Pruning functions may be similar to objective functions, though the input is a subtree. The k-means, -median, and -center objectives are standard pruning functions. The algorithm returns the k-pruning of the tree $\mathcal{T}$ that is optimal according to Ψ, which can be found in polynomial time using dynamic programming.

We now define three infinite families of merge functions and provide sample complexity bounds for these families with any fixed but arbitrary pruning function. The families $\mathcal{A}_1$ and $\mathcal{A}_3$ consist of merge functions $\xi(A, B)$ that depend on the minimum and maximum of all pairwise distances between A and B. The second family, denoted by $\mathcal{A}_2$, is a richer class which depends on all pairwise distances. All classes are parameterized by a single value α.

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

$$\mathcal{A}_2 = \left\{ \left(\frac{1}{|A||B|} \sum_{u \in A, v \in B} (d(u, v))^\alpha \right)^{1/\alpha} \,\middle|\, \alpha \in \mathbb{R} \cup \{\infty, -\infty\} \right\},$$

$$\mathcal{A}_3 = \left\{ \alpha \min_{u \in A, v \in B} d(u, v) + (1 - \alpha) \max_{u \in A, v \in B} d(u, v) \,\middle|\, \alpha \in [0, 1] \right\}.$$

For $i \in \{1, 2, 3\}$, we define $\mathcal{A}_i(\alpha)$ as the merge function in $\mathcal{A}_i$ defined by α. $\mathcal{A}_1$ and $\mathcal{A}_3$ define spectra of merge functions ranging from single-linkage ($\mathcal{A}_1(-\infty)$ and $\mathcal{A}_3(1)$) to complete-linkage ($\mathcal{A}_1(\infty)$ and $\mathcal{A}_3(0)$). $\mathcal{A}_2$ defines a richer spectrum which includes average-linkage in addition to single- and complete-linkage. Given a pruning function Ψ, we denote $(\mathcal{A}_i(\alpha), \Psi)$ as the algorithm which builds a cluster tree using $\mathcal{A}_i(\alpha)$, and then prunes the tree according to Ψ. To reduce notation, when Ψ is clear from context, we often refer to the algorithm $(\mathcal{A}_i(\alpha), \Psi)$ as $\mathcal{A}_i(\alpha)$ and the set of algorithms $\{(\mathcal{A}_i(\alpha), \Psi) \mid \alpha \in \mathbb{R} \cup \{-\infty, \infty\}\}$ as $\mathcal{A}_i$. For example, when the cost function is $\Phi^{(p)}$, then we always set Ψ to minimize the $\Phi^{(p)}$ objective, so the pruning function is clear from context.

3.3 Results and Techniques

Recall that for a given class of merge functions and a `cost` function (a generic clustering objective Φ), our goal is to learn a near-optimal value of α in expectation over an unknown distribution of clustering instances. Our first result is showing that for $i \in \{1, 2, 3\}$, given any α, there exists a distribution over clustering instances for which $\mathcal{A}_i(\alpha)$ is the best algorithm in $\mathcal{A}_i$. This means the optimal value of α depends on the underlying, unknown distribution, and must be learned. Let $\mathbb{V}$ denote the set of all clustering instances over at most n points. With a slight abuse of notation, we will use $\Phi_{(\mathcal{A}_i(\alpha), \Psi)}(\mathcal{V})$ to denote the abstract cost of the clustering produced by $(\mathcal{A}_i(\alpha), \Psi)$ on the instance $\mathcal{V}$.

Theorem 3.3.1. *For $i \in \{1,2,3\}$ and a permissible value of α for $\mathcal{A}_i$, there exists a distribution $\mathcal{D}$ over clustering instances $\mathbb{V}$ such that $\mathbb{E}_{\mathcal{V}\sim\mathcal{D}}\left[\Phi^{(p)}_{\mathcal{A}_i(\alpha)}(\mathcal{V})\right] < \mathbb{E}_{\mathcal{V}\sim\mathcal{D}}\left[\Phi^{(p)}_{\mathcal{A}_i(\alpha')}(\mathcal{V})\right]$ for all permissible values of $\alpha' \neq \alpha$ for $\mathcal{A}_i$.*

Pseudo-dimension upper bounds Now for an arbitrary objective function Φ and arbitrary pruning function Ψ, we analyze the complexity of the classes

$$\mathcal{H}_{\mathcal{A}_1\times\{\Psi\},\Phi} = \left\{ \Phi_{(\mathcal{A}_1(\alpha),\Psi)} : \mathbb{V} \to \mathbb{R}_{\geq 0} \;\middle|\; \alpha \in \mathbb{R}\cup\{\infty,-\infty\}\right\},$$
$$\mathcal{H}_{\mathcal{A}_2\times\{\Psi\},\Phi} = \left\{ \Phi_{(\mathcal{A}_2(\alpha),\Psi)} : \mathbb{V} \to \mathbb{R}_{\geq 0} \;\middle|\; \alpha \in \mathbb{R}\cup\{\infty,-\infty\}\right\}, \text{ and}$$
$$\mathcal{H}_{\mathcal{A}_3\times\{\Psi\},\Phi} = \left\{ \Phi_{(\mathcal{A}_3(\alpha),\Psi)} : \mathbb{V} \to \mathbb{R}_{\geq 0} \;\middle|\; \alpha \in [0,1]\right\}.$$

We drop the subscript Φ from $\mathcal{H}_{\mathcal{A}_i\times\{\Psi\},\Phi}$ when the objective function is clear from context. Furthermore, in our analysis we will often fix a tuple $\mathcal{V} = (V,d)$ and use the notation $\Phi_{\mathcal{A}_i,\mathcal{V}}(\alpha)$ to analyze how $\Phi_{\mathcal{A}_i(\alpha)}(\mathcal{V})$ changes as a function of α. We start with $\mathcal{A}_1$ and $\mathcal{A}_3$.

Theorem 3.3.2. *For all objective functions $\Phi^{(p)}$, Pdim$\left(\mathcal{H}_{\mathcal{A}_1,\Phi^{(p)}}\right) = \Theta(\log n)$ and Pdim$\left(\mathcal{H}_{\mathcal{A}_3,\Phi^{(p)}}\right) = \Theta(\log n)$. For all other objective functions Φ [2] and all pruning functions Ψ, Pdim$\left(\mathcal{H}_{\mathcal{A}_1\times\{\Psi\},\Phi}\right) = O(\log n)$ and Pdim$\left(\mathcal{H}_{\mathcal{A}_3\times\{\Psi\},\Phi}\right) = O(\log n)$.*

Note that for $\alpha \neq \alpha'$, the clustering returned by $\mathcal{A}_1(\alpha)$ and the associated cost are both identical to that of $\mathcal{A}_1(\alpha')$ if both the algorithms construct the same merge tree. We upper bound the pseudo-dimension by showing that for any pairs of subsets of V, (A,B) and (X,Y), that could potentially merge, there exist eight points $p,p' \in A$, $q,q' \in B$, $x,x' \in X$, and $y,y' \in Y$ such that the decision of which pair to merge depends on the sign of $((d(p,q))^\alpha + d(p',q')^\alpha)^{1/\alpha} - ((d(x,y))^\alpha + d(x',y')^\alpha)^{1/\alpha}$. Using a consequence of Rolle's Theorem, we can show that the sign of the above expression as a function of α flips at most four times across $\mathbb{R}$. Since each merge decision is defined by eight points, iterating over all (A,B) and (X,Y) it follows that we can identify all $O(n^8)$ unique 8-tuples of points which correspond to a value of α at which some decision flips. This means we can divide $\mathbb{R}\cup\{-\infty,\infty\}$ into $O(n^8)$ intervals over each of which the merge tree, and therefore the output of $\Phi_{\mathcal{A}_1,\mathcal{V}}(\alpha)$, is fixed.

The upper bound on the pseudo-dimension implies a computationally efficient and sample efficient learning algorithm for $\mathcal{A}_i$ for $i \in \{1,3\}$. First, we know that $m = O\left((H/\epsilon)^2\left(\log n + \log\frac{1}{\delta}\right)\right)$ samples are sufficient to (ϵ,δ)-learn the optimal algorithm in $\mathcal{A}_i$. Next, as a consequence of the theorem, the range of feasible values of α can be partitioned into $O(mn^8)$ intervals, such that the output of $\mathcal{A}_i(\alpha)$ is fixed over the entire set of samples on a given interval. Moreover, these intervals are easy to compute. Therefore, a learning algorithm can iterate over the set of intervals, and for each interval I, choose an arbitrary $\alpha \in I$ and compute the average cost of $\mathcal{A}_i(\alpha)$ evaluated on the samples. The algorithm then outputs the α that minimizes the average cost.

Theorem 3.3.3. *Let Φ be a clustering objective and let Ψ be a pruning function computable in polynomial time. Given an input sample of size $m = O\left(\left(\frac{H}{\epsilon}\right)^2\left(\log n + \log\frac{1}{\delta}\right)\right)$, and a value $i \in \{1,3\}$, there exists an algorithm which (ϵ,δ)-learns the class $\mathcal{A}_i \times \{\Psi\}$ with respect to the cost function Φ and it is computationally efficient.*

Now we turn to $\mathcal{A}_2$. We obtain the following bounds on the pseudo-dimension.

Theorem 3.3.4. *For objective functions $\Phi^{(p)}$, Pdim$(\mathcal{H}_{\mathcal{A}_2,\Phi^{(p)}}) = \Theta(n)$. For all other objective functions Φ and all pruning functions Ψ, Pdim$\left(\mathcal{H}_{\mathcal{A}_2\times\{\Psi\},\Phi}\right) = O(n)$.*

Similar to the previous families of algorithms, this theorem implies that there exists an algorithm which (ϵ,δ)-learns the optimal algorithm in $\mathcal{A}_2$, using $m = O\left((H/\epsilon)^2\left(n + \log\frac{1}{\delta}\right)\right)$ samples. However, the

[2] Recall that although k-means, k-median, and k-center are the most popular choices, the algorithm designer can use other objective functions such as the distance to the ground truth clustering.

algorithm is not computationally efficient.

Key challenges. One of the key challenges in analyzing the pseudo-dimension of our clustering algorithm classes is that we must develop deep insights into how changes to an algorithm's parameters affect the solution the algorithm returns on an arbitrary input. For example, the cost function could be the k-means or k-median objective function, or even the distance to some ground-truth clustering. As we range over algorithm parameters, we alter the merge step by tuning an intricate measurement of the overall similarity of two point sets and we alter the pruning step by adjusting the way in which the combinatorially complex cluster tree is pruned. The cost of the returned clustering may vary unpredictably.

In this way, our analysis requires more care than standard complexity derivations commonly found in machine learning contexts. Typically, for well-understood function classes used in machine learning, such as linear separators or other smooth curves in Euclidean spaces, there is a simple mapping from the parameters of a specific hypothesis to its prediction on a given example and a close connection between the distance in the parameter space between two parameter vectors and the distance in function space between their associated hypotheses. Roughly speaking, it is necessary to understand this connection in order to determine how many significantly different hypotheses there are over the full range of parameters. Due to the inherent complexity of the classes we consider, connecting the parameter space to the space of approximation algorithms and their associated costs requires a much more delicate analysis. Indeed, the key technical part of our work involves understanding this connection from a learning-theoretic perspective. In fact, the structure we discover in our pseudo-dimension analysis is what allowed us to develop computationally efficient meta-algorithms for algorithm configuration.

3.4 Related Work

Algorithm configuration Application-specific algorithm configuration has been used in the artifical intelligence community for decades [70]. For example, Leyton-Brown et al. used algorithm configuration in designing combinatorial auctions [57], Demmel et al. used this technique for scientific computing [39], Caseau et al. studied the problem for vehicle routing [29], and Xu et al. used this model to design a portfolio-based algorithm for SAT [78]. Gupta and Roughgarden were the first to theoretically introduce algorithm configuration as a formal learning framework [49]. They gave pseudo-dimension bounds for families of greedy heuristics, and showed how to learn the best greedy algorithm for knapsack, maximum-weight independent set, and machine scheduling.

Agglomerative clustering with dynamic programming Agglomerative clustering algorithms with dynamic programming are prevalent in practice due to their simplicity and efficiency. For example, agglomerative algorithms with dynamic programming have been used to cluster business listings maintained by Google, and a newsgroup documents data set [7], software clustering [71], and bioinformatic datasets [77]. These families of algorithms have strong theoretical guarantees in a variety of settings. For example, Awasthi et al. show that complete-linkage returns the optimal clustering under $(2+\sqrt{3})$-perturbation resilience for any center-based objective [10], Balcan et al. show that single-linkage returns a near-optimal clustering for k-center under 2-perturbation resilience or $(3, \epsilon)$-perturbation resilience [16], and Balcan et al. show that a robust linkage procedure with a preprocessing step can be used to cluster stable min-sum and k-median instances [17]. Grosswendt and Roeglin show that complete-linkage yields a constant approximation for k-center for any metric that is induced by a norm, for constant dimension.

Chapter 4

Distributed Machine Learning

4.1 Introduction

In this chapter, we study distributed models of machine learning. Distributed computation is playing a major role in modern large-scale machine learning practice with a lot of work in this direction in the last few years [13, 15, 58, 61, 80, 81]. In the first model, the high-level form is where massive amounts of data are collected centrally, and for space and efficiency reasons this data must be dispatched to distributed machines in order to perform some machine learning task [58, 81]. In the second model, the data is inherently distributed, for example, hospitals may keep records of their patients locally, but may want to cluster the entire spread of patients across all hospitals. In this setting, it is assumed that data is partitioned arbitrarily across all machines.

When data is dispatched to distributed machines, past work has focused on performing the dispatching randomly [80, 81]. Random dispatching has the advantage that it is clean to analyze theoretically. Motivated by the fact that in practice, similar data points tend to have the same or similar classification, and more generally, classification rules of high accuracy tend to be "locally simple but globally complex" [74], we propose a new paradigm for performing *data-dependent dispatching* that takes advantage of such structure by sending similar datapoints to similar machines. For example, a *globally* accurate classification rule may be complicated, but each machine can accurately classify its *local* region with a simple classifier. We introduce and analyze dispatching techniques that partition a set of points such that similar examples end up on the same machine/worker, while satisfying key constraints present in a real world distributed system including balancedness and fault-tolerance. Such techniques can then be used within a simple, but highly efficient distributed system that first partitions a small initial segment of data into a number of sets equal to the number of machines. Then each machine locally and independently applies a learning algorithm, with no communication between workers at training. At the prediction time, we use a super-fast sublinear algorithm for directing new data points to the most appropriate machine. In our framework, a central machine starts by clustering a small sample of data into roughly equal-sized clusters, where the number of clusters is equal to the number of available machines. Next, we extend this clustering into an efficient dispatch rule that can be applied to new points. This dispatch rule is used to send the remaining training data to the appropriate machines and to direct new points at prediction time. In this way, similar datapoints wind up on the same machine. To perform the initial clustering used for dispatch, we use classic clustering objectives (k-means, k-median, and k-center).

As mentioned in previous chapters, clustering is a fundamental problem in machine learning with a diverse set of applications. As datasets become larger, sequential algorithms designed to run on a single machine are no longer feasible. Additionally, in may cases data is naturally spread out among multiple locations. For example, hospitals may keep records of their patients locally, but may want to cluster the entire spread of patients across all hospitals in order to do better data analysis and inference. Therefore, distributed clustering algorithms have gained popularity over the past few years [20, 22, 65]. In the distributed setting, it is assumed that the data is partitioned arbitrarily across m machines, and the goal is to find a clustering which approximates the optimal solution over the entire dataset while minimizing communication among machines. Recent work in the theoretical machine learning community establishes guarantees on the clusterings produced in distributed settings for certain problems [20, 22, 65]. For example, Malkomes et al. provide distributed algorithms for k-center and k-center with outliers [65], and Bateni et al. introduce distributed algorithms for capacitated k-clustering under any ℓ_p objective [22]. A key algorithmic idea common among both of these works is the following: each machine locally constructs an approximate size $\tilde{O}(k)$ summary of its data; the summaries are collected on a central machine which then runs a sequential clustering algorithm. A natural question that arises is whether there is a unifying theory for all distributed clustering variants. In the current work, we answer this question by providing a general distributed algorithm for clustering under any ℓ_p objective with or without outliers, and with or without capacity constraints, thereby generalizing and improving over recent results. We also provide distributed algorithms outputting nearly optimal clusterings when the data satisfies the $(1+\alpha,\epsilon)$-approximation stability condition. We complement our upper bounds by giving tight lower bounds with respect to the level of communication needed to find approximate clusterings.

The work from this chapter is based on the following papers: [8, 40].

4.2 Preliminaries

Distributed computing. We use a common, general theoretical framework for distributed computing called the *coordinator model*. There are m machines, and machine 1 is designated as the coordinator. Each machine can send messages back and forth with machine 1. This model is very similar to the *message-passing model*, also known as the *point-to-point* model, in which any pair of machines can send messages back and forth. In fact, the two models are equivalent up to small factors in the communication complexity [26]. We assume the data is arbitrarily partitioned across the m machines, and it is the coordinator's job to output the answer. Most of our algorithms can be applied to the mapreduce framework with a constant number of rounds. For more details, see [22, 65].

Communication Complexity. One of our main goals when designing distributed algorithms is to minimize the communication. For an input X and a protocol Π, the *communication cost* is the total number of bits in the messages sent to and from the coordinator. When designing algorithms, we wish to minimize the *communication complexity*, or the maximum communication cost over all possible inputs X. When proving a lower bound for a problem $\mathcal{A}$, we define the *δ-error communication complexity* as the minimum communication complexity of any randomized protocol Π, such that for all inputs X, the probability that Π outputs an incorrect answer is at most δ. For brevity, we use communication complexity to mean .99-error communication complexity.

Approximation Stability. We consider clustering under a natural property of real-world instances called *approximation stability* (which we briefly mentioned in Chapter 2). Intuitively, a clustering instance satisfies this assumption if all clusterings close in *value* to $\mathcal{OPT}$ are also close in terms of the clusters themselves. This is a desirable property when running an approximation algorithm, since in many applications, the k-means or k-median costs are proxies for the final goal of recovering a clustering that is close to the desired "target" clustering. Approximation stability makes this assumption explicit. First we define two clusterings $\mathcal{C}$ and $\mathcal{C}'$ as *ϵ-close*, if only an ϵ-fraction of the input points are clustered differently in the two clusterings, formally, there exists a permutation σ of $[k]$ such that $\sum_{i=1}^{k}|C_i \setminus C'_{\sigma(i)}| \leq \epsilon n$.

Definition 4.2.1. *A clustering instance (V,d) satisfies $(1+\alpha,\epsilon)$*-approximation stability *if all clusterings $\mathcal{C}$ with cost$(\mathcal{C}) \leq (1+\alpha)\cdot\mathcal{OPT}$ are ϵ-close to $\mathcal{C}$.*

4.3 Results and Techniques

Structure of balanced clustering It is well-known that solving standard clustering objectives optimally are NP-hard (even without the capacity generalizations) [52]. In fact, with the addition of the lower bound, the value of the optimal clustering objective $\mathcal{OPT}$ as a function of k behaves erratically. In uncapacitated clustering and clustering with upper bounds only, given a problem instance, the cost of the optimal solution always decreases as k increases. This is easy to see: given a set of optimal centers, if we add another center v, at the very least v is now distance 0 from a center, which decreases the cost. However, when there are lower bounds on the cluster sizes, there are simple examples in which the value of the optimal solution as a function of k contains a local minimum. For instance, the star graph has this property. A more subtle question is whether there exists a clustering instance with a local *maximum*. In fact, by constructing an intricate clustering instance, we are able to show that *any number* of local maxima may exist.

Theorem 4.3.1. *For all $m \in \mathbb{N}$, there exists a balanced clustering instance in which the k-center, k-median, and k-means objectives contain m local maxima, even for soft capacities.*

We prove Theorem 4.3.1 by constructing a clustering instance where the objective value for all even values of k between $10m$ and $12m$ is low and the objective value for all odd values of k between $10m$ and $12m$ is high. The m odd values will be the local maxima. We will set the lower bound $n\ell$ to be the product of all the even integers between $10m$ and $12m$. The construction involves creating a distinct set of "good" centers, X_k, for each even value of k between $10m$ and $12m$. Let X be the union of these sets. The set X_k contains k points which will be the optimal centers for a k-clustering in our instance. Then we will add an additional set of points, Y, and add edges from Y to the centers in X that satisfy certain properties. Specifically, we show that for each even value of k there is a clustering where the cost of each point in Y is small, any set including a full X_k and a point from a different $X_{k'}$ cannot achieve small cost without violating the capacities, and any set without a full X_k cannot achieve low cost. We satisfy this property by adding "backbone" points (to achieve the first property), and "dispersion" points (to satisfy the last property).

Algorithm for balanced clustering with fault-tolerance In light of these difficulties, one might ask whether any approximation algorithm exists for this problem. We answer affirmatively, by extending previous work [59] to fit our more challenging constrained optimization problem. Our algorithm returns a clustering whose cost is at most a constant factor multiple of the optimal solution, while violating the capacity constraints by a small constant factor. We also add *fault tolerance*, or replication, constraints: the goal of k-clustering with p-replication is to create k clusters with a low objective, such that each point appears in p clusters. In our distributed application, this will serve as a safeguard if a machine breaks down and loses its datapoints.

Theorem 4.3.2. *There exists an algorithm which returns a constant factor approximate solution for the balanced k-clustering with p-replication problem for $p > 1$, where the upper capacity constraints are violated by at most a factor of $\frac{p+2}{p}$, and each point can be assigned to each center at most twice.*

At a high level, our algorithm proceeds by first solving a linear program, followed by careful rounding. The key insight is that p-replication helps to mitigate the capacity violation in the rounding phase. Together with a novel min-cost flow technique, this allows us to simultaneously handle upper and lower bounds on the cluster sizes.

The first step is to solve the standard LP relaxation for k-median clustering. The variables are as follows: for each $i \in V$, let y_i be an indicator for whether i is opened as a center. For $i, j \in V$, let x_{ij} be an indicator for whether point j is assigned to center i. It is well-known that the LP has an unbounded integrality gap, even when the capacities are violated by a factor of $2-\epsilon$ [59]. However, with fault tolerance, the integrality is only unbounded when the capacities are violated by a factor of $\frac{p}{p-1}$. Intuitively, this is because the p centers can 'share' this violation. Next, we partition the points into coarse clusters such that every point is within a constant factor of its LP cost to the closest center in the coarse clustering. We show each cluster has total fractional centers $\geq p/2$, which is crucially ≥ 1 for $p \geq 2$ under our model of fault tolerance. Therefore, we can aggregate openings within each coarse cluster using an iterative greedy procedure. We open at least one center per cluster, and the total cost is proportional to the LP cost, however, we may incur a factor $\frac{p+2}{p}$ increase to the capacity constraints. Finally, we round the x's (the assignment variables) by setting up a min cost flow problem. Using the Integral Flow Theorem, we are guaranteed there is an assignment with low cost.

Distributed clustering Now we consider a different setting in which the data is arbitrarily distributed across different machines at the start. Our goal is to output a quality clustering while minimizing the amount of communication between machines. We give a simple algorithmic framework, together with a careful analysis, to prove strong guarantees in various settings. Each machine performs a k-clustering on its own data, and the centers, along with the size of their corresponding clusters, are sent to a central machine, which then runs a weighted clustering algorithm on the mk centers. For the case of clustering with outliers, each machine runs a $(k+z)$-clustering, and the central machine runs a clustering algorithm that handles outliers.

Theorem 4.3.3. *Given a sequential (δ, α)-approximation algorithm $\mathcal{A}$ for balanced k-clustering with the ℓ_p objective with z outliers, and given a sequential (γ, β)-approximation algorithm $\mathcal{B}$ for k-clustering with the ℓ_p objective, then there exists a distributed algorithm for clustering in ℓ_p with z outliers, with communication cost $O(m(k+z)(d+\log n)\gamma)$. The number of centers opened is δk and the approximation ratio is $(2^{3p-1}\alpha^p\beta^p + 2^{2p-1}(\alpha^p+\beta^p))^{1/p}$. For k-median and k-center, this ratio simplifies to $4\alpha\beta + 2\alpha + 2\beta$.*

Setting $\mathcal{B}$ to be the $(\frac{8\log n}{\epsilon}, 1+\epsilon)$-bicriteria approximation algorithm for k-median [62], the approximation ratio becomes $6\alpha + 2 + \epsilon$. If instead we plug in the sequential approximation algorithm for k-median with z outliers [33], we obtain the first constant-factor approximation algorithm for k-median with outliers. Our proof of Theorem 4.3.3 reasons about the optimal clustering in certain settings where subsets of the outliers are removed, to ensure the constant approximation guarantee carries through to the final result.

Next, we improve over Theorem 4.3.3 if it is known up front that the data satisfies approximation stability. We give modified algorithms which leverage this structure to output a clustering very close to optimal, using no more communication than in Theorem 4.3.3. We focus on k-median, but the details for k-center and ℓ_p for $p < \log n$ are similar.

Theorem 4.3.4. *There exists an algorithm which outputs a set of centers defining a clustering that is $O(\epsilon(1+\frac{1}{\alpha}))$-close to $\mathcal{OPT}$ for k-median under $(1+\alpha, \epsilon)$-approximation stability with $O(mk(d+\log n))$ communication.*

We achieve this result with a two-phase algorithm for k-median under approximation stability. The high level structure of the algorithm is similar to the algorithm from Theorem 4.3.3: first each machine clusters its local point set, and sends the weighted centers to the coordinator. Then the coordinator runs a weighted clustering algorithm to output the solution. The difference lies in the algorithm that is run by the machines and the coordinator, which will now take advantage of the approximation stability assumption.

We also show that if the optimal clusters are not too small, the error of the outputted clustering can be pushed even lower.

Theorem 4.3.5. *There exists an algorithm which outputs a clustering that is $O(\epsilon)$-close to $\mathcal{OPT}$ for k-median under $(1+\alpha, \epsilon)$-approximation stability with $O(m^2kd + mk\log n)$ communication if each optimal cluster C_i has size $\Omega(\epsilon n(1+\frac{1}{\alpha}))$.*

4.4 Related Work

Distributed Machine Learning The most popular method of dispatch in distributed learning is random dispatch [80, 81]. This may not produce optimal results because each machine must learn a global model. Previous work has studied partitioning for distributed machine learning [12, 25, 38, 76, 79], but none simultaneously achieve load-balancing guarantees and approximation guarantees for k-median, k-means, or

k-center.

Previous work in theoretical computer science has considered capacitated clustering, or clustering with upper bounds [27, 37, 59, 60], and lower bounds [1, 3], but our algorithm is the first to solve a more general and challenging question of simultaneously handling upper and lower bounds on the cluster sizes, and p-replication.

Distributed Clustering. Balcan et al. showed a coreset construction for k-median and k-means, which leads to a clustering algorithm with $\tilde{O}(mkd)$ communication, and also studied more general graph topologies for distributed computing [20]. Bateni et al. indroduced a construction for *mapping coresets*, which admits a distributed clustering algorithm that can handle balance constraints with communication cost $\tilde{O}(mk)$ [22]. Malkomes et al. showed a distributed 13- and 4- approximation algorithm for k-center with and without outliers, respectively [65]. Chen et al. studied clustering under the broadcast model of distributed computing, and also proved a communication complexity lower bound of $\Omega(mk)$ for distributed clustering [32], building on a recent lower bound for set-disjointness in the message-passing model [26]. Garg et al. showed a communication complexity lower bound for computing the mean of d-dimensional points [44].

Chapter 5

Future Work

In this thesis, we propose to continue the thread of BWCA in multiple areas including perturbation resilience for clustering and the traveling salesman problem, as well as algorithm configuration and distributed machine learning.

5.1 Perturbation Resilience

We propose to make substantive improvements for what is understood about perturbation resilience. Specifically, we will study two problems: the traveling salesman problem, and clustering. Both of these problems have existing work under perturbation resilience with current open questions.

5.1.1 Traveling salesman problem

The traveling salesman problem (TSP) is a classic NP-hard problem. Despite its popularity, there is no better approximation algorithm than the Christofides algorithm from 1976 [34]. However, various improvements have been shown in special cases [45, 68, 69] (for example, unweighted graphs). Mihalak et al. studied the metric traveling salesman problem under stability, showing a simple greedy algorithm achieves the optimal solution for 1.8-stable instances [67]. Their technique is to use two natural classes of perturbations to rule out bad edges that could cause problems under the greedy algorithm. Their algorithm also trivially works under $(5/3)$-stable instances for the special case of unweighted graphs.

Question 5.1.1. *Does there exist an efficient algorithm for metric TSP under $(1.8-\epsilon)$-stability? Does there exist an efficient algorithm for metric TSP in unweighted graphs under $(5/3-\epsilon)$-stability?*

We believe that any improvement over the upper bound of 1.8 must utilize new classes of perturbations, so our work will focus on developing novel, interesting classes. Mihalak et al. use 'local' perturbations, in which a tour jumps from the optimal tour to a different city, and then immediately jumps back. To develop new ideas, we will consider non-local perturbations, such as tours that jump away from the optimal tour for long periods of time.

For the case of asymmetric TSP, a recent breakthrough by Svensson et al. lowered the approximation ratio from $O(\frac{\log n}{\log\log n})$ to $O(1)$ [73].

Question 5.1.2. *Does there exist an efficient algorithm to solve asymmetric TSP under $O(1)$-stability?*

Asymmetric TSP under stability has never been studied, so we propose to give the first results for this problem. A starting point is to show that stability yields a constant approximation ratio that substantially improves the worst-case approximation ratio of 11,000 [73]. Next, we will try to prove that stability can give a true optimal algorithm for asymmetric TSP. We have initial thoughts that the algorithm of Mihalak et al. works for asymmetric TSP as long as the following condition is satisfied: for all vertices x, y, z, $d(x,y) + d(x,z) > \beta \cdot d(y,z)$. This condition was used to find approximation algorithms for asymmetric TSP [43]. However, finding an unconditional algorithm will require new algorithmic ideas.

5.1.2 Improved algorithms for clustering

As explained in Chapter 2, there have been numerous papers in the last decade studying clustering under perturbation resilience. This line of research culminates in an upper bound that 2-perturbation resilience gives an efficient algorithm for all center-based clustering objectives. Currently, there are no hardness results even under $(1 + \Omega(1))$-perturbation resilience. We propose to make progress towards closing this gap. Specifically, we believe the upper bound can be improved.

Question 5.1.3. *Does there exist an efficient algorithm for center-based clustering under $(2-\epsilon)$-perturbation resilience?*

In preliminary work, we give algorithms for clustering under 1.618-perturbation resilience under a few assumptions using an iterative greedy procedure. The arguments rely on what we call 'optimal perturbations'. Currently, prior work uses perturbations in which only one inter-cluster distance is perturbed, with the majority of perturbed distances staying within one cluster. We define an 'optimal perturbation' as one which chooses a set of potential centers, and then perturbs every distance in a way to maximize the chance that these points become centers.

Next, we will find efficient algorithms for clustering under perturbation resilience. For example, the state-of-the-art min spanning tree + DP algorithm of Angelidakis, Makarychev, and Makarychev runs in time $O(n^2k)$ [4]. We have ideas to reduce the runtime down to linear or even sublinear (by sampling) using the idea of 'core points', where each core point can become an optimal center under a specific perturbation. We can also show each cluster has a constant fraction of core points. If we sample a dataset and maintain at least one core point per cluster, then we can show the algorithm of Angelidakis, Makarychev, and Makarychev [4] will still return the optimal clusters (on the subset).

5.2 Algorithm Configuration

Algorithm configuration has a rich line of empirical work in the last decade. On the theoretical side, there are only a few works, starting with the foundational work of Gupta and Roughgarden [49], and the follow-up of Balcan et al [18]. As described in Chapter 3, the algorithm configuration framework models a problem application as a distribution over problem instances, and the goal is to find the best algorithm from a specified infinite class of parameterized algorithms. In Chapter 3, we studied families of agglomerative clustering algorithms with pruning, showing the empirically best algorithm from a sample can be learned in polynomial time. The corresponding algorithm involves solving for $\binom{n}{8}$ values of α corresponding to "decision points", and then running the α-linkage algorithm (which takes time $\Theta(n^2 \log n)$) on all $\Theta(n^8)$ values of α. Clearly, this runtime is not ideal for real-world applications.

Question 5.2.1. *Can we perform scalable algorithm configuration for clustering with the class $\mathcal{A}_1$ in much less time than $O(n^{10})$, for worst-case or beyond worst-case instances?*

In preliminary work, we have designed an algorithm that only runs α-linkage for the true number of distinct α-intervals which give a fixed output. For instance, $O(n^8)$ is an upper bound based on the decision points of the algorithm, but for many clustering instances, the true number of unique values of α might be significantly smaller than n^8. Therefore, the remaining step is to prove that for large families of real-world clustering instances, this number is $o(n^8)$. First we will show that for clustering instances generated by sampling n points uniformly at random from the d-dimensional cube $[0, 1]^d$, the distinct α-intervals drops by at least a factor of n^2. Note that assuming uniform data is only a first step, since real-world clustering instances rarely look like a uniform distribution. Later we will weaken the distributional assumptions to capture large classes of realistic clustering instances. Intuitively, we show that if the data is uniform, then with high probability, each level of the α-linkage cluster tree contains nodes which are a multiplicative factor greater in size than the previous level. If this is the case, then each set X never merges with a set of size much larger or much smaller than X. This insight allows us to show a substantial reduction in the number of possible outcomes of the α-linkage algorithm.

5.3 Distributed Machine Learning

Chapter 4 outlines a data-dependent approach to distributed machine learning. Crucially, the dispatching step is cast as a clustering problem with the real-world constraints of fault tolerance and balancedness. While sometimes it is clear how to set the parameters from the application, sometimes the best choice of parameters is unknown. For instance, setting the lower capacity constraint too low may result in wasting computing resources, while setting it too high may result in decreased accuracy.

Question 5.3.1. *Can we use algorithm configuration to efficiently learn the best parameters for fault tolerance and balancedness?*

We propose using the framework from Chapter 3 to answer this question. In this case, the “best” parameters will depend on the specific needs and constraints of the application, so we will model the problem with a general cost function that may take into account accuracy of the data, balance of computing resources, robustness to computer failure, and so on.

Bibliography

[1] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. In *Proceedings of the twenty-fifth ACM symposium on Principles of database systems*, pages 153–162, 2006.

[2] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *CoRR*, abs/1612.07925, 2016.

[3] Sara Ahmadian and Chaitanya Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In *Proceedings of the Annual International Colloquium on Automata, Languages, and Programming (ICALP)*, 2016.

[4] Haris Angelidakis, Konstantin Makarychev, and Yury Makarychev. Algorithms for stable and perturbationresilient problems. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2017.

[5] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models - going beyond SVD. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 2012.

[6] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[7] Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 550–558, 2014.

[8] Pranjal Awasthi, Maria-Florina Balcan, and Colin White. General and robust communication-efficient algorithms for distributed clustering. *arXiv preprint arXiv:1703.00830*, 2017.

[9] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a ptas for k-median and k-means clustering. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 309–318, 2010.

[10] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1):49–54, 2012.

[11] Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. *Proceedings of the Annual Symposium on Computational Geometry (SOCG)*, 2015.

[12] Kevin Aydin, MohammadHossein Bateni, and Vahab Mirrokni. Distributed balanced partitioning via linear embedding. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 387–396, 2016.

[13] Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity, and privacy. In *Proceedings of the Annual Conference on Learning Theory (COLT)*,

2012.

[14] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *Journal of the ACM (JACM)*, 60(2):8, 2013.

[15] Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general communication topologies. In *Advances in Neural Information Processing Systems*, 2013.

[16] Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-center clustering under perturbation resilience. In *Proceedings of the Annual International Colloquium on Automata, Languages, and Programming (ICALP)*, 2016.

[17] Maria Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *SIAM Journal on Computing*, 45(1):102–155, 2016.

[18] Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Proceedings of the Annual Conference on Learning Theory (COLT)*, pages 213–274, 2017.

[19] Maria-Florina Balcan and Colin White. Clustering under local stability: Bridging the gap between worst-case and beyond worst-case analysis. *arXiv preprint arXiv:1705.07157*, 2017.

[20] Maria-Florina F Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general topologies. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1995–2003, 2013.

[21] Mohammadhossein Bateni, Aditya Bhaskara, Silvio Lattanzi, and Vahab Mirrokni. Distributed balanced clustering via mapping coresets. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2591–2599, 2014.

[22] Mohammadhossein Bateni, Aditya Bhaskara, Silvio Lattanzi, and Vahab Mirrokni. Distributed balanced clustering via mapping coresets. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2591–2599, 2014.

[23] Shalev Ben-David and Lev Reyzin. Data stability in clustering: A closer look. In *Algorithmic Learning Theory*, pages 184–198. Springer, 2012.

[24] Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(05):643–660, 2012.

[25] Florian Bourse, Marc Lelarge, and Milan Vojnovic. Balanced graph edge partition. In *Proceedings of the international conference on Knowledge discovery and data mining*, pages 1456–1465, 2014.

[26] Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. A tight bound for set disjointness in the message-passing model. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 668–677, 2013.

[27] Jarosław Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 722–736, 2015.

[28] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 737–756, 2015.

[29] 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 (CP)*, pages 144–158. Springer, 1999.

[30] Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation

algorithm for the k-median problem. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 1–10, 1999.

[31] Moses Charikar, Samir Khuller, David M Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 642–651, 2001.

[32] Jiecao Chen, He Sun, David Woodruff, and Qin Zhang. Communication-optimal distributed clustering. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3720–3728, 2016.

[33] Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 826–835, 2008.

[34] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.

[35] Julia Chuzhoy, Sudipto Guha, Eran Halperin, Sanjeev Khanna, Guy Kortsarz, Robert Krauthgamer, and Joseph Seffi Naor. Asymmetric k-center is log* n-hard to approximate. *Journal of the ACM (JACM)*, 52(4):538–551, 2005.

[36] Vincent Cohen-Addad and Chris Schwiegelshohn. On the local structure of stable clustering instances. *CoRR*, abs/1701.08423, 2017.

[37] Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. Lp rounding for k-centers with non-uniform hard capacities. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 273–282, 2012.

[38] Daniel Delling, Andrew V Goldberg, Ilya Razenshteyn, and Renato F Werneck. Graph partitioning with natural cuts. In *IEEE International Parallel and Distributed Processing Symposium*, pages 1135–1146, 2011.

[39] 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*, 93(2):293–312, 2005.

[40] Travis Dick, Mu Li, Venkata Krishna Pillutla, Colin White, Maria Florina Balcan, and Alex Smola. Data driven resource allocation for distributed learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[41] R.M Dudley. The sizes of compact subsets of Hilbert space and continuity of Gaussian processes. *Journal of Functional Analysis*, 1(3):290 – 330, 1967.

[42] Darya Filippova, Aashish Gadani, and Carl Kingsford. Coral: an integrated suite of visualizations for comparing clusterings. *BMC bioinformatics*, 13(1):276, 2012.

[43] Alan M Frieze, Giulia Galbiati, and Francesco Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

[44] Ankit Garg, Tengyu Ma, and Huy Nguyen. On communication cost of distributed statistical estimation and dimensionality. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2726–2734, 2014.

[45] Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2011.

[46] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[47] Anna Grosswendt and Heiko Roeglin. Improved analysis of complete linkage clustering. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, volume 23, pages 656–667, 2015.

[48] Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

[49] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. In *Proceedings of the Annual Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 123–134, 2016.

[50] Rishi Gupta, Tim Roughgarden, and C Seshadhri. Decompositions of triangle-dense graphs. In *Proceedings of the Annual Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 471–482, 2014.

[51] Moritz Hardt and Aaron Roth. Beyond worst-case analysis in private singular vector computation. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 331–340, 2013.

[52] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM (JACM)*, 50(6):795–824, 2003.

[53] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 731–740, 2002.

[54] Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 299–308, 2010.

[55] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time (1+ ε)-approximation algorithm for geometric k-means clustering in any dimensions. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 454–462, 2004.

[56] Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017.

[57] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM (JACM)*, 56(4):22, 2009.

[58] Mu Li, David G Andersen, Alex J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 19–27, 2014.

[59] Shanfei Li. An improved approximation algorithm for the hard uniform capacitated k-median problem. In *Proceedings of the International Workshop on Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX-RANDOM)*, pages 325–338, 2014.

[60] Shi Li. Approximating capacitated k-median with (1+ ε) k open facilities. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 786–796, 2016.

[61] Yingyu Liang, Maria-Florina F Balcan, Vandana Kanchanapally, and David Woodruff. Improved distributed principal component analysis. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3113–3121, 2014.

[62] Jyh-Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992.

[63] Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko, and Justin Ward. A bi-criteria approximation algorithm for k means. In *Proceedings of the International Workshop on Approximation,*

Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX-RANDOM), 2016.

[64] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Bilu-linial stable instances of max cut and minimum multiway cut. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 890–906, 2014.

[65] Gustavo Malkomes, Matt J Kusner, Wenlin Chen, Kilian Q Weinberger, and Benjamin Moseley. Fast distributed k-center clustering with outliers on massive data. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1063–1071, 2015.

[66] Marina Meilă. Comparing clusterings: an information based distance. *Journal of multivariate analysis*, 98(5):873–895, 2007.

[67] Matús Mihalák, Marcel Schöngens, Rastislav Srámek, and Peter Widmayer. On the complexity of the metric tsp under stability considerations. In *SOFSEM*, volume 6543, pages 382–393. Springer, 2011.

[68] Tobias Mömke and Ola Svensson. Removing and adding edges for the traveling salesman problem. *Journal of the ACM (JACM)*, 63(1):2, 2016.

[69] Marcin Mucha. \frac {13}{9}-approximation for graphic tsp. *Theory of computing systems*, 55(4):640–657, 2014.

[70] John R Rice. The algorithm selection problem. *Advances in computers*, 15:65–118, 1976.

[71] Mehreen Saeed, Onaiza Maqbool, Haroon Atique Babri, Syed Zahoor Hassan, and S Mansoor Sarwar. Software clustering techniques and the use of combined algorithm. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 301–306. IEEE, 2003.

[72] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

[73] Ola Svensson, Jakub Tarnawski, and László A Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. *arXiv preprint arXiv:1708.04215*, 2017.

[74] Vladimir N. Vapnik and Leon Bottou. Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 1993.

[75] Sundar Vishwanathan. An o(log*n) approximation algorithm for the asymmetric p-center problem. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, 1996.

[76] Kai Wei, Rishabh K Iyer, Shengjie Wang, Wenruo Bai, and Jeff A Bilmes. Mixed robust/average submodular partitioning: Fast algorithms, guarantees, and applications. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2233–2241, 2015.

[77] 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. *BMC bioinformatics*, 11(1):152, 2010.

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

[79] Y. You, J. Demmel, K. Czechowski, L. Song, and R. Vuduc. CA-SVM: Communication-avoiding support vector machines on clusters. In *IEEE International Parallel and Distributed Processing Symposium*, 2015.

[80] Yuchen Zhang, John Duchi, Michael Jordan, and Martin Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2013.

[81] Yuchen Zhang, John C. Duchi, and Martin Wainwright. Communication-efficient algorithms for statistical optimization. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2012.